Joint Multi-Grained Popularity-Aware Graph Convolution Collaborative Filtering for Recommendation

Kang Liu, Feng Xue, Xiangnan He, *Member, IEEE*, Dan Guo, *Member, IEEE*, and Richang Hong, *Member, IEEE*

***Abstract*— Graph convolution networks (GCNs), with their efficient ability to capture high-order connectivity in graphs, have been widely applied in recommender systems. Stacking multiple neighbor aggregation is the major operation in GCNs. It implicitly captures popularity features because the number of neighbor nodes reflects the popularity of a node. However, existing GCN-based methods ignore a universal problem: users' sensitivity to item popularity is differentiated, but the neighbor aggregations in GCNs actually fix this sensitivity through graph Laplacian normalization, leading to suboptimal personalization. In this work, we propose to model multigrained popularity features and jointly learn them together with high-order connectivity to match the differentiation of user preferences exhibited in popularity features. Specifically, we develop a Joint Multigrained Popularity-aware Graph Convolution Collaborative Filtering model, short for JMP-GCF, which uses a popularity-aware embedding generation to construct multigrained popularity features and uses the idea of joint learning to capture the signals within and between different granularities of popularity features that are relevant for modeling user preferences. In addition, we propose a multistage stacked training strategy to speed up model convergence. We conduct extensive experiments on three public datasets to show the state-of-the-art performance of JMP-GCF. The complete codes of JMP-GCF are released at https://github.com/kangliu1225/JMP-GCF.**

***Index Terms*— Collaborative filtering (CF), graph convolution networks (GCNs), high-order interactions, multigrained popularity.**

I. Introduction

PERSONALIZED recommendation methods have been deployed in many applications [1], [2] to solve information overload and refine user experience in online services. Collaborative filtering (CF) [3] is the mainstream algorithm for recommender systems because of its effectiveness and low computational overload. At its core is using historical user–item interactions to incorporate collaborative signals into the embedding process.

However, CF suffers from the problem of sparsity. One solution is to enhance embeddings with auxiliary information, such as reviews [4], images [5], [6], social networks [2], knowledge graph [7], [8], and demographic characteristics [9]; however, storing these additional data takes up too much memory, and most users resent the collection of personal information. Another solution is to consider neighbor information as an additional feature. For example, SVD++ [10] and FISM [11] aggregate first-order neighbors into user embeddings to enhance the modeling of user preferences. However, they suffer from two shortcomings: 1) the representation of item embedding is not enhanced and 2) high-order interactions are not captured.

The graph convolution network (GCN) [12]–[14] is an advanced deep learning technique for handling graph data that address the above two shortcomings. A common paradigm of GCN is to first process neighbor aggregation using a nonlinear neural network and then iteratively perform this process to capture high-order neighbor information. GC-MC [15] is an early effort that uses a first-order GCN to aggregate neighbor information for all nodes in the user–item graph, thus enhancing both user embedding and item embedding. PinSAGE [16] and NGCF [17] extend the depth of GCN to capture important high-order interactions and, hence, achieve better performance. Inspired by SGCN [18], LR-GCCF [19] and LightGCN [20] further simplify the structure of GCN by removing the redundant nonlinear network layers in graph convolution, which both speeds up the model training and significantly improves performance. In summary, a linear GCN (which excludes nonlinear network layers) significantly outperforms a traditional nonlinear GCN for embedding generation in CF scenarios.

In a user–item graph, the number of first-order neighbors directly reflects the popularity of the current node, and a linear GCN is essentially a process of iteratively performing neighbor aggregation. Therefore, a linear GCN implicitly incorporates popularity features into the node embedding. In addition, the normalization performed after neighbor aggregation is equivalent to scaling the popularity features with a fixed granularity. It is clear that linear GCN-based methods that enhance the node embedding by high-order neighbor aggregation benefit from this implicit capture of popularity

Manuscript received August 25, 2021; revised December 11, 2021 and January 22, 2022; accepted February 10, 2022. This work was supported in part by the Special Support Plan for Innovation and Entrepreneurship in Anhui Province and in part by the National Natural Science Foundation of China under Grant 61876058. *(Corresponding author: Feng Xue.)*

Kang Liu, Dan Guo, and Richang Hong are with the Key Laboratory of Knowledge Engineering With Big Data, the Intelligent Interconnected Systems Laboratory of Anhui Province, and the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, Anhui 230601, China (e-mail: kangliu1225@gmail.com; guodan@hfut.edu.cn; hongrc@hfut.edu.cn).

Feng Xue is with the Key Laboratory of Knowledge Engineering With Big Data, the Intelligent Interconnected Systems Laboratory of Anhui Province, and the School of Software, Hefei University of Technology, Hefei, Anhui 230601, China (e-mail: feng.xue@hfut.edu.cn).

Xiangnan He is with the School of Data Science, University of Science and Technology of China, Hefei, Anhui 230026, China (e-mail: xiangnanhe@gmail.com).

Digital Object Identifier 10.1109/TCSS.2022.3151822

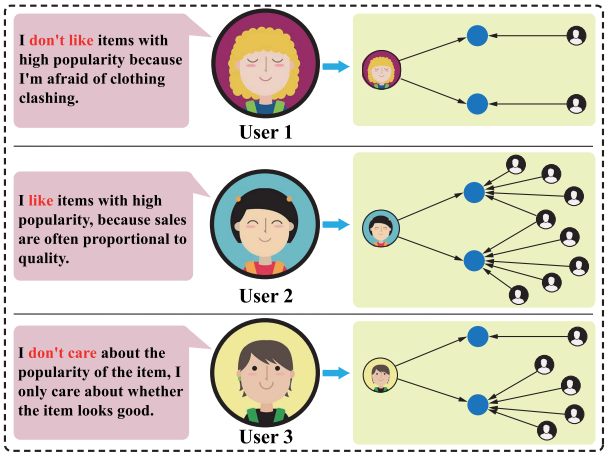

Fig. 1. Illustration of the differentiation in user preferences exhibited in item popularity. User1 prefers items with low popularity; hence, the items that she buys have few interactions in the right-hand interaction graph. User2 likes items with high popularity; hence, she buys items that have many interactions. User3 is insensitive to the popularity of items, so she buys items with both high and low popularity.

features. However, existing popularity-based methods [21]–[23] perform poorly in recommendation tasks, mainly because they do not fully use interaction data to jointly learn CF signals and popularity features and to achieve the complementation between them. Intuitively, a better solution is to use GCN as the base module to capture high-order CF signals and incorporate popularity features into the embedding generation.

Although neighbor aggregation can capture popularity features implicitly, **the sensitivity of user preferences to popularity is fixed, that is, neighbor aggregation captures popularity features at a fixed granularity.** This does not match the real situation, as shown in Fig. 1, in which the sensitivity of users to popularity shows great differentiation. Existing methods ignore this inevitable problem and, therefore, fail to achieve excellent personalization in respect of popularity. Thus, **capturing the distribution of users' preferences over popularity features with different granularities** is capable of modeling preferences better.

In this work, we propose a GCN-based model to achieve the above goal by jointly learning multigrained popularity features and high-order interactions. Specifically, we propose a simplified GCN block to capture high-order collaborative signals and construct multigrained popularity-aware embeddings and a layer selection mechanism to filter out the most expressive embeddings. To achieve the joint learning of popularity features and collaborative signals, we propose a separated Bayesian personalized ranking (BPR) loss to optimize the model and update parameters from the perspective of different layer semantics and different popularity granularities. In addition, we propose a multistage stacked training method to speed up convergence while facilitating the learning of popularity features. Finally, we conduct extensive experiments on three million-size datasets to verify the effectiveness of our proposed **J**oint **M**ultigrained **P**opularity-aware **G**raph Convolution **C**ollaborative **F**iltering (JMP-GCF) model.

The main contributions of our work are given as follows.

1) We highlight the critical importance of capturing multigrained popularity features to match the differentiation of user preferences exhibited in item popularity.
2) We propose JMP-GCF, a novel GCN-based recommendation framework, which jointly learns multigrained popularity features and high-order interactions to capture the signals related to modeling user preferences within and between different popularity granularities.
3) We conduct extensive experiments on three public datasets. Experimental results demonstrate the state-of-the-art performance of JMP-GCF.
4) To support the subsequent studies on JMP-GCF, we release the complete codes of JMP-GCF at https://github.com/kangliu1225/JMP-GCF.

II. Methodology

In this section, we present our proposed JMP-GCF model, whose general architecture is illustrated in Fig. 2. The model has four components: 1) the simplified graph convolution block for integrating multigrained popularity features into collaborative signals and generating embedding matrices; 2) the layer selection mechanism for filtering out optimal graph convolution layers; 3) the model prediction and optimizer layer to output predicted preference scores, and the separated BPR loss to optimize model parameters; and 4) the multistage stacked training strategy for speeding up convergence and facilitating the learning of multigrained popularity.

A. Simplified Graph Convolution Block

1) Embedding Generation: According to prior work [20], the network layers and nonlinear activation functions in traditional GCNs are redundant in CF scenarios. Therefore, we construct an embedding generation block that uses a simplified graph convolution to implement neighbor aggregation and message passing, named the simplified graph convolution block. Fig. 3 shows the detailed structure of this block. The first graph convolution operation, that is, the first yellow rectangle (marked Layer $= 1$) in Fig. 3, can be represented as follows:

$$\mathbf{E}^{(1)} = (\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})(\mathbf{D}+\mathbf{I})^{-\frac{1}{2}} \times [\mathbf{U}||\mathbf{I}] \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{(m+n)\times(m+n)}$ is the adjacency matrix, $\mathbf{D}$ is the degree matrix of $\mathbf{A}$, $\mathbf{I}$ is the identity matrix of $\mathbf{A}$, $\mathbf{U} \in \mathbb{R}^{m\times k}$ and $\mathbf{I} \in \mathbb{R}^{n\times k}$ are the randomly initialized user and item embedding matrices, respectively, m and n denote the number of users and items, respectively, k is the dimension length of the embedding, $||$ represents the matrix concatenation, and $\mathbf{E}^{(1)} \in \mathbb{R}^{(m+n)\times k}$ is the output of the first graph convolution layer.

Because the graph convolution operation is stackable in this block, assuming that the current layer is l, that is, the lth yellow rectangle, the output of this layer can be formulated as follows:

$$\mathbf{E}^{(l)} = (\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})(\mathbf{D}+\mathbf{I})^{-\frac{1}{2}} \times \mathbf{E}^{(l-1)} \tag{2}$$

where $\mathbf{E}^{(l)}$ and $\mathbf{E}^{(l-1)}$ denote the embedding matrices obtained at layers l and $l-1$, respectively. Note that the embedding matrices' output at each layer can be redivided into user and item embedding matrices. Setting the total number of graph

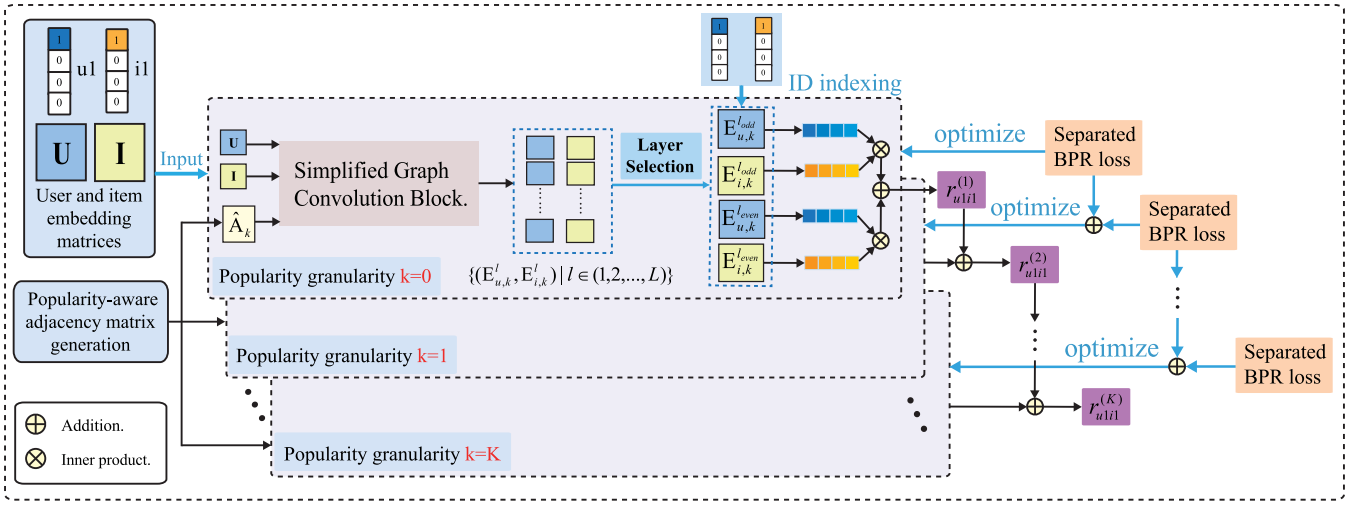

Fig. 2. Illustration of JMP-GCF, where k denotes the specific granularity in different simplified graph convolution blocks and K is the maximum popularity granularity. For a given user u1 and item i1, JMP-GCF is split into $K+1$ simplified graph convolution blocks to generate embeddings with different popularity granularities and optimal layers, and then, the corresponding preference scores between u1 and i1 are calculated, respectively. Finally, these embeddings are jointly optimized by using separated BPR loss so that multigrained popularity features, layer semantics, and high-order connectivities are learned.

convolution layers to L, the outputs of this block are given as follows:

$$\left\{\left(\mathbf{E}_u^{(l)}, \mathbf{E}_i^{(l)}\right) | l \in (1, 2, \ldots, L)\right\} \tag{3}$$

where $\mathbf{E}_u^{(l)}$ and $\mathbf{E}_i^{(l)}$ are the embedding matrices obtained at the lth layer of this block.

2) Popularity-Aware Embedding Generation: Intuitively, popularity is an important component of user preferences, and users have different sensitivities to popularity features. Therefore, the weights of nodes with high popularity (or degree) should be appropriately increased to improve their sensitivity to popularity. We can obtain a normalized adjacency matrix containing popularity features at different granularity levels by fine-tuning the exponent of the degree matrix as follows:

$$\hat{\mathbf{A}}_k = (\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})(\mathbf{D}+\mathbf{I})^{-\frac{1}{2}+kc} \tag{4}$$

where c is a constant, which is the smallest unit by which the granularity of popularity can vary (we set c to 0.1 in our experiments), and we set k to the granularity of popularity, which indicates how much the embeddings varies according to popularity. A greater value of kc represents a higher weight for the node popularity features and forces the model to recommend items with higher popularity to users. Specifically, the embedding generation process of (4) can be viewed as the simultaneous capture of the higher order CF signals and the popularity feature with granularity k. When $k = 0$, (4) is equivalent to the traditional graph convolution operation (cf. [12]), i.e., only the modeling of the CF signal is considered. When $k = 1$, the model starts to incorporate small granularity of popularity features and amplifies the value of the embeddings appropriately based on that granularity $k = 1$. When $k = 2$ or larger, the model incorporates a larger popularity granularity and scales up the embedding values more, which corresponds to the model being more sensitive to popularity. Since users have different sensitivities

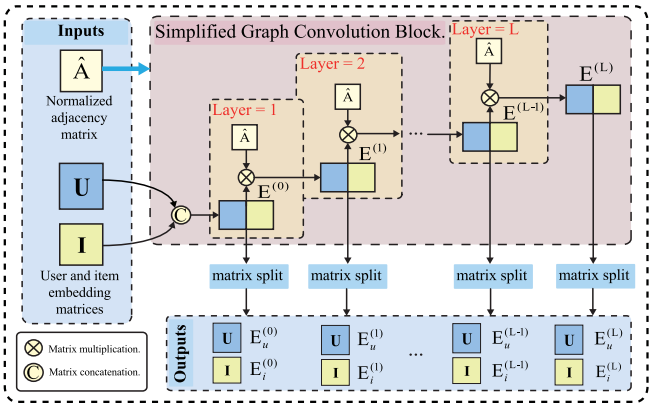

Fig. 3. Illustration of the simplified graph convolution block. Given a normalized adjacency matrix $\hat{\mathbf{A}}$, randomly initialized user, and item embedding matrices, $\mathbf{U}$ and $\mathbf{I}$, this block outputs the embedding matrices of users and items obtained at each graph convolution layer.

to popularity features, we set different popularity granularities k in embedding generation so that the recommendation results of the model can cover items with different popularity as much as possible, thus improving user satisfaction.

Inputting $\hat{\mathbf{A}}_k$ to the simplified graph convolution block as a normalized adjacency matrix, we obtain the embedding matrices at the lth graph convolution layer as follows:

$$\mathbf{E}_k^{(l)} = \hat{\mathbf{A}}_k \times \mathbf{E}_k^{(l-1)} \tag{5}$$

where $\mathbf{E}_k^{(0)} = [\mathbf{U}||\mathbf{I}]$, and $\mathbf{E}_k^{(l)}$ can be redivided into user and item embedding matrices. We obtain the following user and item embedding matrices at each layer:

$$\left\{\left(\mathbf{E}_{u,k}^{(l)}, \mathbf{E}_{i,k}^{(l)}\right) | l \in (1, 2, \ldots, L), k \in (0, 1, \ldots, K)\right\} \tag{6}$$

where K denotes the maximum popularity granularity. In our work, for simplicity, we set K to 2 to obtain three sets of user and item embedding matrices at each layer.

Having described how to incorporate the multigrained popularity features into the embedding generation, we next introduce a straightforward strategy that has the potential to better personalize the matching of user preferences and different popularity levels. Specifically, we assign corresponding weights λ_k to the user embedding matrices at different popularity granularities k. We formulate this process as follows:

$$\left\{\mathbf{E}_{u,k}^{(l)*} = \lambda_k \cdot \mathbf{E}_{u,k}^{(l)} | l \in (1, 2, \ldots, L), k \in (0, 1, \ldots, K)\right\} \quad (7)$$

where λ_k represents the user's preference level for different granularities of popularity, which can be obtained in two ways. One is to construct an attention network to calculate the sensitivity of user u to different popularity granularities. We leave it for future study. The other is that, in an online environment, users set the granularity value according to their own needs, thus controlling the popularity of the tweeted content. Specifically, all λ_k's are set to 1 by default, and if the user chooses a specific granularity value $k = 1$, we make $\lambda_1 = 10$ (or other value greater than 1), which is equivalent to that the system will be more inclined to generate recommendations based on that popularity granularity. In this work, we only briefly introduce this strategy without experimentally verifying it since it requires real-time user participation.

B. Layer Selection Mechanism

In this section, we propose a layer selection mechanism to ensure the expressiveness of embedding. This mechanism filters out the optimal graph convolution layer from the simplified graph convolution block.

In the simplified graph convolution block, we argue that the embedding output at layer l contains the complete information of the previous $l-1$ layers, that is, the embedding at a higher layer has a larger interaction space (the ratio of incorporated neighbor nodes to global nodes) and is, thus, more expressive. The following derivation proves the above assumption when $l = 3$ (for simplicity, normalization is not considered here):

$$\begin{aligned}\mathbf{E}^{(3)} &= (\mathbf{A}+\mathbf{I})\mathbf{E}^{(2)} = (\mathbf{A}+\mathbf{I})(\mathbf{A}+\mathbf{I})\mathbf{E}^{(1)} \\ &= \mathbf{A}(\mathbf{A}+\mathbf{I})\mathbf{E}^{(1)} + (\mathbf{A}+\mathbf{I})\mathbf{E}^{(1)} = \mathbf{A}(\mathbf{A}+\mathbf{I})\mathbf{E}^{(1)} + \mathbf{E}^{(2)}\end{aligned} \quad (8)$$

where $\mathbf{E}^{(3)}$, $\mathbf{E}^{(2)}$, and $\mathbf{E}^{(1)}$ denote the embedding matrices output at the third, second, and first layers, respectively.

It is worth mentioning that too high a layer tends to introduce substantial noise (that is, it contains many nodes that are completely unrelated to the current node). Therefore, it is important to choose the appropriate graph convolution layers to ensure sufficient interaction space and avoid contamination by noise nodes. This contrasts with the layer aggregation mechanism [17], [19], [20], which uses the embeddings of all the graph convolution layers.

In addition to considering the size of interaction space, we believe that the optimal graph convolution layers can sufficiently capture two layer semantics (heterogeneous and homogeneous nodes), which are interpreted in Fig. 4. Because these two layer semantics have different interaction space sizes in each layer, the positive effects of both semantics can be maximized if their interaction space is sufficiently large. For this reason, we obtain the optimal odd layer l_{odd} and even layer l_{even} output by Algorithm 1, which ensures that the interaction spaces of the two semantics are both sufficient.

Algorithm 1 Layer Selection

Input: Dataset $\mathbf{D}$; threshold parameter α; neighborhood function to compute the number of k-hop neighbors for current node u, $N^k(u)$.
Output: Optimal odd layer l_{odd}; optimal even layer l_{even}.

Compute the number of total users and items in $\mathbf{D}$, N_{user} and N_{item}, respectively.
Randomly sample 100 users into a set, H_{user}.
Let $l_{odd} \leftarrow 0$, $l_{odd} \leftarrow 0$, $k_1 \leftarrow 0$, $k_2 \leftarrow 0$.
while $l_{odd} = 0$ **do**
  Obtain next odd layer, $k_1 \leftarrow 2 \times k_1 + 1$.
  Initialize the total number of odd layer nodes, $S_{odd} \leftarrow 0$.
  for $u \in H_{user}$ **do**
    $S_{odd} \leftarrow S_{odd} + N^{k_1}(u)$
  end for
  if $(S_{odd}/N_{item})/100 >= \alpha$ **then**
    Obtain optimal odd layer, $l_{odd} \leftarrow k_1$
  end if
end while
while $l_{even} = 0$ **do**
  Obtain next even layer, $k_2 \leftarrow 2 \times k_2 + 2$.
  Initialize the total number of even layer nodes, $S_{even} \leftarrow 0$.
  for $u \in H_{user}$ **do**
    $S_{even} \leftarrow S_{even} + N^{k_2}(u)$
  end for
  if $(S_{even}/N_{user})/100 >= \alpha$ **then**
    Obtain optimal even layer, $l_{even} \leftarrow k_2$
  end if
end while
return l_{odd} and l_{even}

For a specific dataset, we can determine whether the current hop corresponds to a semantically sufficient graph convolution layer by calculating the interaction space size of the two semantics under each hop. In Algorithm 1, $S_{\text{odd}}/N_{\text{item}}$ denotes the interaction space size for the first semantic at hop k_1. Similarly, $S_{\text{even}}/N_{\text{user}}$ denotes the interaction space size for the other semantic at hop k_2. We empirically set the threshold parameter α to 50% for both semantics. Finally, the algorithm outputs l_{odd} and l_{even}, which are the odd and even layers for which the interaction space size first reaches α.

C. Model Prediction and Optimization

According to the simplified graph convolution block and layer selection mechanism, we obtain the user and item embedding matrices that contain sufficient layer semantics and multigrained popularity features as follows:

$$\left\{\left(\mathbf{E}_{u,k}^{(l)}, \mathbf{E}_{i,k}^{(l)}\right) | l \in (l_{\text{odd}}, l_{\text{even}}), k \in (1, 2, \ldots, K)\right\} \quad (9)$$

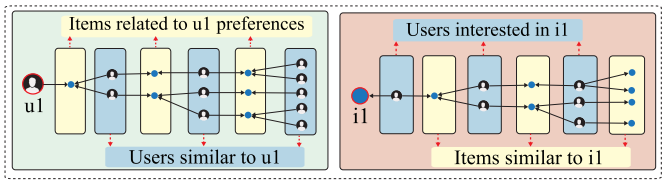

Fig. 4. Illustration of the relationship between the semantics of neighbor nodes and the number of hops where the neighbor nodes are located, and u_1 and i_1 are the target user and item, respectively.

where $\mathbf{E}_{u,k}$ and $\mathbf{E}_{i,k}$ denote the user and item embedding matrices at the popularity granularity of k, respectively.

The embedding representations of target users and items can be obtained directly by indexing the IDs (u_1 and i_1) in these matrices as follows:

$$\left\{\left(\mathbf{e}^{(l)}_{u_1,k}, \mathbf{e}^{(l)}_{i_1,k}\right) | l \in (l_{\text{odd}}, l_{\text{even}}), k \in (1, 2, \ldots, K)\right\} \tag{10}$$

where $\mathbf{e}_{u_1,k}$ and $\mathbf{e}_{i_1,k}$ denote the embedding representations for user u_1 and item i_1 at the popularity granularities of k, respectively.

We use a simple inner product operation to calculate the prediction scores at multiple popularity granularities and two layer semantics, and sum them to obtain the final prediction score between the target user and item. The specific prediction function is given as follows:

$$r_{u_1 i_1} = \sum_{k=0}^{K}\left[\left(\mathbf{e}^{l_{\text{odd}}}_{u_1,k}\right)^T \mathbf{e}^{l_{\text{odd}}}_{i_1,k} + \left(\mathbf{e}^{l_{\text{even}}}_{u_1,k}\right)^T \mathbf{e}^{l_{\text{even}}}_{i_1,k}\right]. \tag{11}$$

1) Separated BPR Loss: To facilitate the learning of layer semantics and popularity features for user preferences, we propose a separated BPR loss to optimize the model parameters as follows:

$$L = \sum_{(u,i,j)\in O}\sum_{k=0}^{K}\left\{-\ln\sigma\left[(\mathbf{e}^{l_{\text{odd}}}_{u,k})^T \mathbf{e}^{l_{\text{odd}}}_{i,k} - \left(\mathbf{e}^{l_{\text{odd}}}_{u,k}\right)^T \mathbf{e}^{l_{\text{odd}}}_{j,k}\right] - \ln\sigma \times\left[\left(\mathbf{e}^{l_{\text{even}}}_{u,k}\right)^T \mathbf{e}^{l_{\text{even}}}_{i,k} - \left(\mathbf{e}^{l_{\text{even}}}_{u,k}\right)^T \mathbf{e}^{l_{\text{even}}}_{j,k}\right]\right\} + \lambda||\mathbf{H}||_2^2 \tag{12}$$

where $O = \{(u, i, j) \mid i \in N_u, j \notin N_u\}$ denotes the training data, N_u denotes the observed item set for user u, and $\sigma(\cdot)$ is the sigmoid function. We apply L_2 regularization on $\mathbf{H}$ controlled by λ, where $\mathbf{H}$ denotes the embedding matrices in (9).

Without considering the popularity granularity (k), we let $d_{ui\leftarrow uj} = (\mathbf{e}_u)^T \mathbf{e}_i - (\mathbf{e}_u)^T \mathbf{e}_j$ and further derive the core term in (12) as follows:

$$-\ln\sigma\left(d^{l_{\text{odd}}}_{ui\leftarrow uj}\right) - \ln\sigma\left(d^{l_{\text{even}}}_{ui\leftarrow uj}\right) = -\ln\left[\frac{1}{1 + e^{-d^{l_{\text{odd}}}_{ui\leftarrow uj}} + e^{-d^{l_{\text{even}}}_{ui\leftarrow uj}} + e^{-\left(d^{l_{\text{odd}}}_{ui\leftarrow uj} + d^{l_{\text{even}}}_{ui\leftarrow uj}\right)}}\right]. \tag{13}$$

The above expanded formulas illustrate that the target of our proposed loss function is to essentially maximize $d^{l_{\text{odd}}}_{ui\leftarrow uj}$, $d^{l_{\text{even}}}_{ui\leftarrow uj}$, and $(d^{l_{\text{odd}}}_{ui\leftarrow uj} + d^{l_{\text{even}}}_{ui\leftarrow uj})$, that is, to maximize the distance between positive and negative samples when considering the two layer semantics both individually and jointly.

This derivation demonstrates that the separated BPR loss can explicitly capture signals related to modeling user preferences within and between different semantics. In addition, this joint learning of separated BPR loss also considers the effects of different granularities of popularity features simultaneously so that features within and interactions between different granularities are also successfully captured and incorporated into the process of modeling user preferences. We verify the effectiveness of separated BPR loss in Section III-C2.

D. Multistage Stacked Training

To effectively incorporate multigrained popularity features into the embedding generation and speed up the training convergence, we propose a multistage stacked training method, which divides the overall framework into multiple training phases depending on the granularity of popularity features, as shown in Fig. 2. In the following, we provide details of the model optimization for each training phase. It is worth noting that the model in each training phase can perform the recommendation task independently.

1) First Training Phase: We first consider the learning of coarse-grained popularity features because it can accelerate the convergence of model training (evidence in Section III-D). According to (4)–(6) and the layer selection mechanism, the embedding matrices with maximum popularity granularity are $\{(\mathbf{E}^{(l)}_{u,K}, \mathbf{E}^{(l)}_{i,K}) | l \in (l_{\text{odd}}, l_{\text{even}})\}$, where K is the maximum increase in the magnitude of popularity granularity. Using the ID indexing, we obtain the embedding corresponding to the user and item as $\{(\mathbf{e}^{(l)}_{u,K}, \mathbf{e}^{(l)}_{i,K}) | l \in (l_{\text{odd}}, l_{\text{even}})\}$. We use the following formulas to compute the preference score between the target user u_1 and item i_1:

$$r^{(1)}_{u1i1} = \left(\mathbf{e}^{l_{\text{odd}}}_{u1,K}\right)^T \mathbf{e}^{l_{\text{odd}}}_{i1,K} + \left(\mathbf{e}^{l_{\text{even}}}_{u1,K}\right)^T \mathbf{e}^{l_{\text{even}}}_{i1,K}. \tag{14}$$

In addition, we use the following separated BPR loss to optimize the model parameters:

$$L_1 = \sum_{(u,i,j)\in O} -\ln\sigma\left[\left(\mathbf{e}^{l_{\text{odd}}}_{u,K}\right)^T \mathbf{e}^{l_{\text{odd}}}_{i,K} - \left(\mathbf{e}^{l_{\text{odd}}}_{u,K}\right)^T \mathbf{e}^{l_{\text{odd}}}_{j,K}\right] - \ln\sigma\left[\left(\mathbf{e}^{l_{\text{even}}}_{u,K}\right)^T \mathbf{e}^{l_{\text{even}}}_{i,K} - \left(\mathbf{e}^{l_{\text{even}}}_{u,K}\right)^T \mathbf{e}^{l_{\text{even}}}_{j,K}\right] + \lambda||\mathbf{H}_1||_2^2 \tag{15}$$

where $\mathbf{H}_1 = \{(\mathbf{E}^{(l)}_{u,K}, \mathbf{E}^{(l)}_{i,K}) | l \in (l_{\text{odd}}, l_{\text{even}})\}$ denotes the trainable embedding representations in layers l_{odd} and l_{even} in the current training phase.

2) Stacked Training Phase: When the first training phase has sufficiently captured coarse-grained popularity features, we focus on learning finer-grained features in subsequent training and then stack these features on top of the previous training to achieve joint learning of multigrained features and collaborative signals. Assuming that the current training phase is p, we obtain the embedding matrices at the corresponding popularity granularity are $\{(\mathbf{E}^{(l)}_{u,k}, \mathbf{E}^{(l)}_{i,k}) | l \in (l_{\text{odd}}, l_{\text{even}})\}$, where $k = K - p + 1$. The embedding representations for users and items from these embedding matrices are $\{(\mathbf{e}^{(l)}_{u,k}, \mathbf{e}^{(l)}_{i,k}) | l \in (l_{\text{odd}}, l_{\text{even}})\}$. In addition, the prediction and objective functions of this phase can be seen as a stack on top of the previous training phase.

TABLE I
STATISTICS OF THE DATASETS

Dataset	User #	Item #	Interaction #	Density
Gowalla	29,858	40,981	1,027,370	0.00084
Yelp2018	31,668	38,048	1,561,406	0.00130
Amazon-Book	52,643	91,599	2,984,108	0.00062

The prediction function in the current training phase p is given as follows:

$$r_{u1i1}^{(p)} = \left(\mathbf{e}_{u1,k}^{l_{\text{odd}}}\right)^T \mathbf{e}_{i1,k}^{l_{\text{odd}}} + \left(\mathbf{e}_{u1,k}^{l_{\text{even}}}\right)^T \mathbf{e}_{i1,k}^{l_{\text{even}}} + r_{u1i1}^{(p-1)} \tag{16}$$

where $r_{u1i1}^{(p)}$ is the predicted score for user u_1 and item i_1 in the current training phase, and $r_{u1i1}^{(p-1)}$ is the predicted score in the $(p-1)$th training phase. We use the following separated BPR loss to optimize the model parameters for the current training phase:

$$\begin{aligned} L_p = \sum_{(u,i,j)\in O} & -\ln\sigma\left[\left(\mathbf{e}_{u,k}^{l_{\text{odd}}}\right)^T \mathbf{e}_{i,k}^{l_{\text{odd}}} - \left(\mathbf{e}_{u,k}^{l_{\text{odd}}}\right)^T \mathbf{e}_{j,k}^{l_{\text{odd}}}\right] \\ & -\ln\sigma\left[\left(\mathbf{e}_{u,k}^{l_{\text{even}}}\right)^T \mathbf{e}_{i,k}^{l_{\text{even}}} - \left(\mathbf{e}_{u,k}^{l_{\text{even}}}\right)^T \mathbf{e}_{j,k}^{l_{\text{even}}}\right] + \lambda||\mathbf{H}_p||_2^2 + L_{p-1} \end{aligned} \tag{17}$$

where L_p is the optimization target of the current training phase of the model, L_{p-1} is the optimization target of the $(p-1)$th training phase, and $\mathbf{H}_p = \{(\mathbf{E}_{u,k}^{(l)}, \mathbf{E}_{i,k}^{(l)})|l \in (l_{\text{odd}}, l_{\text{even}})\}$ denotes all trainable embedding representations in the current phase. It is worth noting that (17) and (12) are equivalent when the training phase p is $K+1$.

III. EXPERIMENT

We conduct experiments to evaluate our proposed JMP-GCF and further experimental analysis to verify the effectiveness of each component of JMP-GCF. We aim to answer the following questions.

1) *RQ1:* How does JMP-GCF perform compared with other state-of-the-art recommender methods?
2) *RQ2:* Are all components (layer selection, separated BPR loss, popularity integration, and multistage stacked training) of JMP-GCF helpful?
3) *RQ3:* How does the joint learning of multigrained popularity features and high-order interactions affect the JMP-GCF?

A. Experimental Settings

1) Datasets: For a fair comparison, we conduct experiments on the three public datasets, Gowalla [24], Yelp2018, and Amazon-Book [5], used in [17] and [20]. We present the details of these three datasets in Table I. We randomly sample 80% of the interaction data for each user as the training set and use the remaining 20% as the test set. Furthermore, we sample 10% of the interaction data from the training set as the validation set to tune the hyperparameters. It is worth mentioning that the datasets and dataset settings that we use are the same as those of the previous GCN-based works, i.e., LightGCN [20] and NGCF [17].

2) Evaluate Metrics: We select two protocols [17]: Recall@20 and NDCG@20 that are commonly used in recent works [17], [20], [25] to evaluate model performance. Specifically, we compute the average Recall@20 and NDCG@20 for each user in the test set. Note that, for each user, we used all the items that the user had not interacted with as negative samples.

3) Baselines: To demonstrate the effectiveness of our proposed JMP-GCF, we compare it with the following baselines.

1) *BPRMF [26]:* This is a classical matrix factorization (MF) method with BPR loss, which is widely used as a recommendation baseline.
2) *SVD++ [10]:* This is a variant of MF, which integrates the historical interactions into user embeddings. It can also be regarded as a one-layer GCN that only passes messages for user nodes.
3) *NeuMF [27]:* This is a neural CF method that uses nonlinear neural networks as an interaction function instead of a simple inner product of MF.
4) *GC-MC [15]:* This is a GCN-based encoder–decoder framework, which only uses one nonlinear graph convolution layer to capture first-order interactions for users and items.
5) *NGCF [17]:* This method adopts a GCN technique with multiple nonlinear graph convolution layers, and it concatenates the embeddings obtained at each layer as the final representations of users and items.
6) *LR-GCCF [19]:* This model fine-tunes the structure of NGCF by removing the nonlinear activation function in the graph convolution layer.
7) *LightGCN [20]:* This is a GCN-based CF model, which generates the embeddings with a simplified GCN and uses the summation of the embeddings obtained at each layer as the final representation.
8) *SGL [28]:* This is the latest GCN-based CF model, which employs node dropout, edge dropout, and random walk to construct different graph views and uses self-supervised contrastive learning to maximize the agreement between different views. In addition, SGL applies LightGCN as the base module for embedding generation.

4) Hyperparameter Settings: For all methods of comparison, we set the embedding size and batch size to 64 and 2048, respectively. We tune the learning rate in $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and search the coefficient of L_2 regularization in $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. Specifically, 10^{-3} is the optimal learning rate for the three datasets. 10^{-4}, 10^{-4}, and 10^{-5} are the optimal coefficients of L_2 regularization for the Gowalla, Yelp2018, and Amazon-Book, respectively. Besides, we use the Xavier initializer [29] to achieve the embedding initialization for all models.

B. Overall Comparison (RQ1)

Table II reports the performance of all the methods. We obtain the following findings.

1) BPRMF performs worst in all cases, which indicates that a simple inner product fails to capture complex

TABLE II
OVERALL PERFORMANCE COMPARISON

Method	Gowalla		Yelp2018		Amazon-Book	
	recall	ndcg	recall	ndcg	recall	ndcg
BPRMF	0.1291	0.1109	0.0433	0.0354	0.0250	0.0196
SVD++	0.1439	0.1297	0.0500	0.0412	0.0332	0.0251
NeuMF	0.1399	0.1212	0.0451	0.0363	0.0258	0.0200
GC-MC	0.1395	0.1204	0.0462	0.0379	0.0288	0.0224
NGCF	0.1569	0.1327	0.0579	0.0477	0.0337	0.0261
LR-GCCF	0.1701	0.1452	0.0604	0.0498	0.0375	0.0296
LightGCN	0.1830	0.1554	0.0649	0.0530	0.0411	0.0315
SGL	0.1810	0.1531	0.0675	0.0555	0.0478	0.0379
JMP-GCF	**0.1871**	**0.1583**	**0.0702**	**0.0577**	**0.0499**	**0.0388**
%Improv.	2.24%	1.87%	4.00%	3.96%	4.39%	2.37%

interactions between nodes in the user–item graph. NeuMF outperforms BPRMF across all datasets, which verifies that using nonlinear neural networks can distill complex and nonlinear connectivities between nodes.

2) GC-MC outperforms NeuMF on the Yelp2018 and Amazon-Book datasets, which verifies that using first-order GCN to incorporate one-hop neighbors can improve model performance. However, GC-MC is slightly worse than NeuMF on the Gowalla dataset, which might be because the nonlinear GCN is insufficient for capturing complex interactions.
3) Compared with NeuMF and GC-MC, SVD++ achieves higher Recall@20 and NDCG@20 on all datasets, which demonstrates that explicitly incorporating first-order neighbors into user embeddings is more effective at capturing complex interactions than nonlinear networks in NeuMF and GC-MC.
4) NGCF achieves a significant improvement over GC-MC and SVD++ on the three datasets, which indicates that explicitly aggregating high-hop neighbors can effectively capture important high-order interactions and further improve model performance.
5) LR-GCCF generally achieves better performance than NGCF in all cases. This might be due to the removal of the nonlinear activation function in LR-GCCF, which can improve the learning process of GCN-based methods.
6) LightGCN significantly outperforms LR-GCCF on all datasets, which illustrates that nonlinear network layers are redundant in terms of enhancing the capture of high-order collaborative signals.
7) SGL significantly outperforms LightGCN on the Yelp2018 and Amazon-Book datasets, which demonstrates the effectiveness of supplementing the recommendation task with self-supervised learning. Unexpectedly, SGL is slightly weaker than LightGCN on the Gowalla dataset, which may be due to the loss of important interaction information caused by SGL's utilization of node dropout and edge dropout to construct different graph views. However, the superior performance of SGL on Yelp2018 and Amazon-Book may be due to the larger number of spurious interactions in these two datasets, so the dropout and contrastive learning strategies can enhance the model robustness and, thus, alleviate the negative impact of spurious interactions on user preference modeling.
8) Our proposed JMP-GCF yields the best performance across all cases. Specifically, JMP-GCF outperforms LightGCN w.r.t. Recall@20 by 2.24%, 8.17%, and 21.41% on the Gowalla, Yelp2018, and Amazon-Book datasets, respectively. Although JMP-GCF does not improve much on the Gowalla dataset, it substantially outperforms other methods on the Yelp2018 and Amazon-Book datasets. This is probably because multigrained popularity features are incorporated in JMP-GCF, and the users in the Yelp2018 and Amazon-Book datasets are more sensitive to popularity than those in the Gowalla dataset. In addition, JMP-GCF outperforms SGL w.r.t. Recall@20 by 3.37%, 4.00%, and 4.39% on the Gowalla, Yelp2018, and Amazon-Book datasets, respectively. Such results further verify the superiority of incorporating multigrained popularity features.
9) Another observation is that the overall performance is best on the Gowalla dataset and worst on the Amazon-Book dataset. We believe that such results may be attributed to the interaction sparsity and application scenarios of the datasets. Specifically, according to Table I, the Amazon-book dataset has the highest sparsity, which leads to the lowest accuracy of recommendations. Although the Yelp2018 is denser than the Gowalla, the performance on the Yelp2018 is significantly weaker than that on the Gowalla. Such results may be attributed to the variability of the datasets in terms of application scenarios. Specifically, the application scenario of Yelp2018 is restaurant and bar recommendations. Intuitively, most people tend to dine nearby, i.e., the geographic location feature in the Yelp2018 dataset is closely related to the user interaction behavior, but that feature is not taken into count in recommendation generation. In contrast, Gowalla is a check-in dataset, in which the user interactions are relatively less correlated with geographic location. We think that, even though Gowalla is sparser than Yelp2018, this kind of scenario-related factors lead to better performance of Gowalla than that of Yelp2018 when only ID information is used.

To better verify the superiority of JMP-GCF over the traditional CF baseline SVD++ and the strongest GCN-based baseline LightGCN, we investigate their performance w.r.t. different sizes k of the recommendation lists. As shown in Fig. 5, the curve of JMP-GCF always lies above the curves of SVD++ and LightGCN, which further indicates the effectiveness of JMP-GCF. In addition, we find that a phenomenon is demonstrated on all three datasets, i.e., JMP-GCF achieves a greater improvement on NDCG@k compared to Recall@k, which indicates that JMP-GCF has a stronger advantage in ranking preference modeling.

C. Analysis of Model Components (RQ2)

The experiments in this section are designed to validate the effectiveness of each component of JMP-GCF. For simplicity,

TABLE III

PERFORMANCE OF RECALL@20 AND NDCG@20 ON THE GOWALLA, YELP2018, AND AMAZON-BOOK DATASETS W.R.T. DIFFERENT COMPONENTS IN JMP-GCF

Variants	Gowalla		Yelp2018		Amazon-Book	
	recall	ndcg	recall	ndcg	recall	ndcg
JMP-GCF/ $\{ms,mp,se,ls\}$	0.1800 (-3.9%)	0.1515 (-4.5%)	0.0650 (-8.0%)	0.0540 (-6.9%)	0.0400 (-24.7%)	0.0301 (-28.9%)
JMP-GCF/ $\{ms,mp,se\}$	0.1814 (-3.1%)	0.1528 (-3.6%)	0.0669 (-4.9%)	0.0553 (-4.3%)	0.0450 (-10.9%)	0.0350 (-10.9%)
JMP-GCF/ $\{ms,mp\}$	0.1831 (-2.2%)	0.1545 (-2.5%)	0.0674 (-4.2%)	0.0554 (-4.2%)	0.0461 (-8.2%)	0.0359 (-8.1%)
JMP-GCF/ $\{ms\}$	0.1842 (-1.6%)	0.1550 (-2.1%)	0.0689 (-1.9%)	0.0566 (-1.9%)	0.0489 (-2.0%)	0.0378 (-2.6%)
JMP-GCF	0.1871	0.1583	0.0702	0.0577	0.0499	0.0388

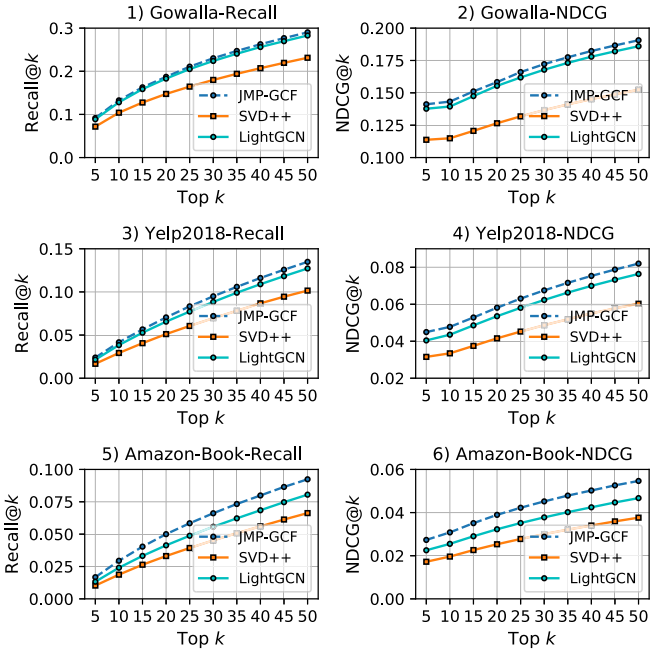

Fig. 5. Performance comparison of JMP-GCF, SVD++, and LightGCN w.r.t. different recommendation list sizes k on the Gowalla, Yelp2018, and Amazon-Book datasets. (a) Gowalla-Recall. (b) Gowalla-NDCG. (c) Yelp2018-Recall. (d) Yelp2018-NDCG. (e) Amazon-Book-Recall. (f) Amazon-Book-NDCG.

the symbol ms denotes the multistage stacked training method, mp denotes multigrained popularity features, se denotes separated BPR loss, and ls denotes the layer selection mechanism. These symbols are used to specify the following variants of JMP-GCF.

1) Effect of Using Layer Selection Mechanism: We set up two model variants: JMP-GCF/{ms, mp, se, ls}, which removes ms, mp, se, and ls from JMP-GCF; JMP-GCF/{ms, mp, se}, which additionally considers ls on top of JMP-GCF/{ms, mp, se, ls}, that is, it uses the layer selection mechanism to generate the final node representations. The performance of these two variants is reported in Table III, which shows that the variant considering ls achieves significant improvements over JMP-GCF/{ms, mp, se, ls} on the three datasets. This indicates that the approach of selecting optimal layers to ensure adequate capture of the layer semantics is effective.

2) Effect of Using Separated BPR Loss: We set up the model variant JMP-GCF/{ms, mp}, which uses se instead of the traditional BPR loss used in JMP-GCF/{ms, mp, se}, as the objective function to optimize the model parameters. The multigrained popularity features are not considered in this variant here; se only implements the joint learning of two layer semantics. The experimental results are recorded in Table III, which shows that JMP-GCF/{ms, mp} outperforms JMP-GCF/{ms, mp, se} across all cases. This indicates that the joint learning of two layer semantics is reasonable, and the separated BPR loss is helpful for improving model performance.

3) Effect of Using Popularity Integration: We set up a variant JMP-GCF/{ms} that incorporates multigrained popularity features based on JMP-GCF/{ms, mp}. Specifically, in accordance with (6), we set $k = 0$, $k = 1$, and $k = 2$ to construct the popularity features at the three granularities and use (11) to predict the preference scores and (12) to jointly learn the popularity features at these three granularities. The experimental results are recorded in Table III, showing that JMP-GCF/{ms} achieves a significant improvement over the variant JMP-GCF/{ms, mp} on all three datasets. This demonstrates that popularity features play an important role in modeling user preferences, and it proves that user preferences show differentiation in popularity features. In addition, we find that incorporating popularity features results in a more pronounced improvement in performance on the Yelp2018 and Amazon-Book datasets, probably because users in these two scenarios are more sensitive to popularity features.

4) Effect of Using Multistage Stacked Training: We compare the performance of the variant JMP-GCF/{ms} with that of JMP-GCF, which is equivalent to adding the multistage stacked training strategy to JMP-GCF/{ms}. Specifically, we use (15) and (17) to implement the stacked joint learning of multigrained popularity features. Table III shows that JMP-GCF outperforms JMP-GCF/{ms} in all cases, which demonstrates the effectiveness of multistage stacked training. It is worth mentioning that the main purpose of our design of multistage stacked training is to accelerate the model convergence, which is further verified in Section III-D.

5) Study on Selection of Graph Convolution Layers: Despite the effectiveness of the layer selection mechanism for improving model performance, as verified in Section III-C1, such a simple experimental comparison cannot verify that the layer selection mechanism can output the optimal graph convolution layers. For this reason, we conduct a set of comparison experiments on the Gowalla and Yelp2018 datasets with the number of graph convolution layers as an independent variable and using the JMP-GCF/{ms, mp, se, ls} as the base model. We set up the layer numbers in the range of {1,2,3,4,5}. Table IV records the results of performance when selecting different graph convolution layers. We have the following findings.

1) As the number of graph convolution layers increases, the model performance shows an increasing trend, and the model performs best in both datasets when the number of layers is 3, which illustrates the importance of capturing high-order interactions. Moreover, the model

TABLE IV

PERFORMANCE OF RECALL@20 AND NDCG@20 ON THE GOWALLA AND YELP2018 DATASETS W.R.T. THE SELECTION OF GRAPH CONVOLUTION LAYER (RESULTS ON THE AMAZON-BOOK DATASET HAVE THE SAME TREND, WHICH ARE REMOVED FOR SPACE)

Layers Selection	Gowalla		Yelp2018	
	recall	ndcg	recall	ndcg
(1)	0.1668(-8.8%)	0.1419(-7.7%)	0.0605(-10.6%)	0.0502(-10.2%)
(2)	0.1776(-2.1%)	0.1502(-1.7%)	0.0641(-4.4%)	0.0539(-2.6%)
(3)	0.1800(-0.8 %)	0.1515(-0.9%)	0.0664(-0.8%)	0.0550(-0.5%)
(4)	0.1786(-1.6%)	0.1502(-1.7%)	0.0643(-4.0%)	0.0541(-2.2%)
(5)	0.1681(-8.0%)	0.1418(-7.8%)	0.0590(-13.4%)	0.0492(-12.4%)
(1,2,3,4,5)	0.1730(-4.9%)	0.1467(-4.2%)	0.0585(-14.4%)	0.0502(-10.2%)
(2,3)	0.1809(-0.3%)	0.1524(-0.3%)	0.0663(-0.9%)	0.0548(-0.9%)
(3,4)	**0.1814**	**0.1528**	**0.0669**	**0.0553**
(4,5)	0.1798(-0.9%)	0.1511(-1.1%)	0.0637(-5.0%)	0.0525(-5.3%)

performance declines when a larger number of layers is used, which validates our point in Section II-B that too high a graph convolution layer tends to introduce noise data.

2) The model variant that concatenates the embeddings' output from the first five graph convolution layers as the node representations performs worse than the model variant that uses the combination of the third and fourth layers This suggests that the low-layer embeddings are unnecessary for generating the node representations in a GCN because the high-layer embeddings contain complete information from all the previous layers, which is consistent with the point made in Section II-B.
3) The model variant that uses the combination of graph convolution layers, such as (3, 4), outperforms the model variant using combinations (2, 3) and (4, 5). This indicates that the third and fourth layers are the optimal combination of graph convolution layers for the Gowalla and Yelp2018 datasets. The optimal odd and even layers output by the layer selection mechanism are also the third and fourth layers; such consistency further validates the effectiveness of the layer selection mechanism.

D. Analysis of Multigrained Popularity Features (RQ3)

The experiments in this section are designed to investigate the effect of popularity features at different granularities, as well as the method of fusing multigrained popularity features, on the GCN model. In particular, we use (6) to construct three types of popularity features with different granularities by setting k to 0,1, and 2, which corresponds to three model variants, JMP-GCF$_{k=0}$, JMP-GCF$_{k=1}$, and JMP-GCF$_{k=2}$, respectively. When $k = 2$, the embeddings contain more coarse-grained popularity features, that is, the model is more sensitive to popularity. In addition, we create a variant JMP-GCF$_{k=0,1,2}$, which uses (12) to accumulate these popularity features with different granularities to achieve the joint learning of popularity and high-order interactions. JMP-GCF is equivalent to additionally considering the multistage stacked training strategy on top of JMP-GCF$_{k=0,1,2}$ to accelerate the convergence of model training and enhance the learning process for multigrained popularity features. Table V records the detailed performance of these model variants on the three datasets within 2000 training epochs, and Fig. 6 shows the trend of model performance for these variants (due to space constraints, we only show the model performance for the first 1000 training epochs in Fig. 6). We have the following findings.

TABLE V

PERFORMANCE OF RECALL@20 AND NDCG@20 ON THE GOWALLA, YELP2018, AND AMAZON-BOOK DATASETS W.R.T. DIFFERENT GRANULARITIES OF POPULARITY FEATURES

Method	Gowalla		Yelp2018		Amazon-Book	
	recall	ndcg	recall	ndcg	recall	ndcg
JMP-GCF$_{k=0}$	0.1831	0.1545	0.0674	0.0554	0.0461	0.0359
JMP-GCF$_{k=1}$	0.1836	0.1551	0.0688	0.0564	0.0473	0.0370
JMP-GCF$_{k=2}$	0.1820	0.1539	0.0691	0.0570	0.0479	0.0372
JMP-GCF$_{k=0,1,2}$	0.1842	0.1550	0.0693	0.0571	0.0489	0.0378
JMP-GCF	**0.1871**	**0.1583**	**0.0702**	**0.0577**	**0.0499**	**0.0388**

1) For the scenarios that consider only a single popularity granularity (that is, $k = 0$, $k = 1$, or $k = 2$), JMP-GCF$_{k=1}$ achieves the best performance on the Gowalla dataset, whereas JMP-GCF$_{k=2}$ performs best on the Yelp2018 and Amazon-Book datasets. This illustrates that users in different scenarios have different sensitivities to popularity features. Fig. 6 shows that JMP-GCF$_{k=2}$ consistently converges most rapidly but then gradually decreases, whereas JMP-GCF$_{k=0}$ converges most slowly. This indicates that coarse-grained popularity features can accelerate model convergence but easily lead to overfitting.
2) The model variant JMP-GCF$_{k=0,1,2}$, which considers popularity features with three granularities, outperforms JMP-GCF$_{k=0}$, JMP-GCF$_{k=1}$, and JMP-GCF$_{k=2}$ on the three datasets. This illustrates the importance of incorporating multigrained popularity features for user preference modeling. Fig. 6 shows that, in terms of convergence speed, JMP-GCF$_{k=0,1,2}$ outperforms JMP-GCF$_{k=0}$ on the Gowalla dataset and outperforms JMP-GCF$_{k=0}$ and JMP-GCF$_{k=1}$ on the Yelp2018 and Amazon-Book datasets, and there is no risk of overfitting. This indicates that jointly learning multigrained popularity features both accelerates model convergence and avoids the problem of overfitting.
3) JMP-GCF achieves the best performance on the three datasets, which indicates that the multistage stacked training approach can better jointly learn multigrained popularity features. Fig. 5 shows that JMP-GCF has the fastest model convergence speed on all three datasets. It is worth mentioning that the transient drop in the trend curve of JMP-GCF at 300 epochs is caused by the stacking of untrained popularity features with a new granularity when the training phase is switched.

E. Hyperparameter Studies

In JMP-GCF, L_2 regularization coefficient λ is the most important hyperparameter, and we study here the

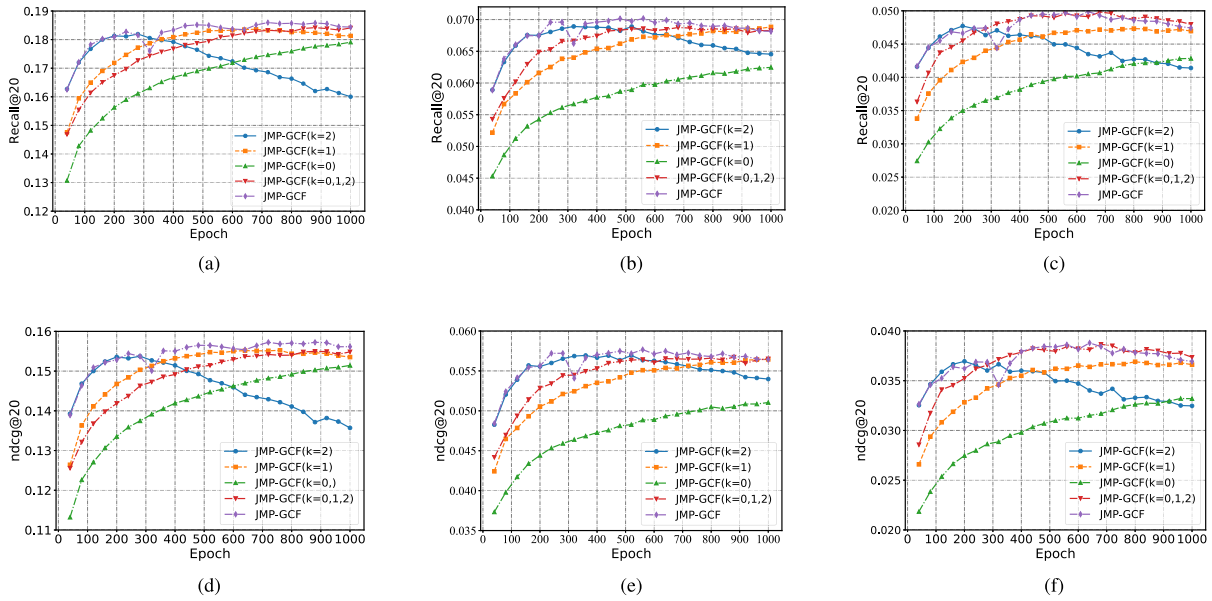

Fig. 6. Performance trends of Recall@20 and NDCG@20 within 1000 training epochs w.r.t. different granularities of popularity features on the Gowalla, Yelp2018, and Amazon-Book datasets. (a) Gowalla-Recall@20. (b) Yelp2018-Recall@20. (c) Amazon-book-Recall@20. (d) Gowalla-NDCG@20. (e) Yelp2018-NDCG@20. (f) Amazon-book-NDCG@20.

performance of JMP-GCF w.r.t. different λ's on the three datasets.

As shown in Fig. 7, JMP-GCF achieves the best performance on the Gowalla and Yelp2018 datasets when $\lambda = 10^{-4}$, and it performs best on the Amazon-Book dataset when $\lambda = 10^{-5}$. More specifically, we observe that, when λ is small ($\lambda = 10^{-3}$ or less), the model performance does not show significant degradation on the three datasets, which indicates that the JMP-GCF model is highly stable, i.e., it does not easily fall into the issue of overfitting. In contrast, when λ is set very large ($\lambda = 10^{-2}$ or larger), the model consistently performs poorly, which may be due to that too strong L_2 regularization limits the learning ability of the model, so, when applying JMP-GCF to a new dataset, we recommend not choosing too large L_2 regularization coefficient.

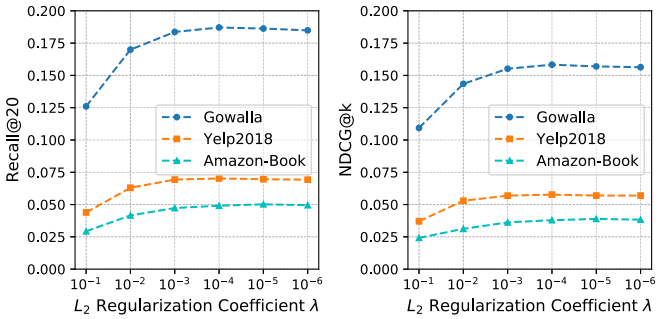

Fig. 7. Performance of JMP-GCF w.r.t. different L_2 regularization coefficients λ on the Gowalla, Yelp2018, and Amazon-Book datasets.

IV. Related Work

In this section, we briefly review traditional CF methods and GCN-based methods, which are most closely related to our work.

A. Traditional Collaborative Filtering Methods

A common assumption of CF methods [30], [31] is that similar users exhibit similar preferences to items that they commonly interacted with in the past. MF [26] is the pioneering work of CF, vectorizes users and items by mapping their ID information into embeddings, and reconstructs the interaction between users and items with the inner product of their embeddings. BiasesSVD [32] assumes that MF fails to accurately capture the differences in preferences among users and items, and introduces a specific bias on top of MF to compensate for embeddings' weaknesses in expressiveness. Distinct from merely using the ID information of users and items, another type of CF method uses historical interacted items to enhance the embedding generation process. Specifically, SVD++ [10] and FISM [11] integrate the embeddings of one-hop neighbor nodes into user representations. To explore high-order interactions and further improve model performance, HOSLIM [33] encodes high-hop neighbor nodes into the embeddings, but the time complexity is too high to handle the million-size datasets efficiently. In addition, many researchers believe that some auxiliary properties related to users (or items), such as age, gender, occupation [34], multimedia features [1], and knowledge graph [35], [36], are relevant to user preferences. To achieve this, SVDFeature [9] effectively integrates user- and item-related attributes into the framework of MF. VBPR [6] and CDL [37] use deep image feature to enhance item representations. In some recent efforts, deep neural networks have been applied to the CF algorithm because of their powerful feature learning capabilities. For example, NeuMF [27], JNCF [38], and DICF [39] use neural networks to learn nonlinear collaborative signals between users–item interactions.

B. Graph Convolution Network-Based Methods

GCN [12], [14] has received considerable attention in recent years and is becoming increasingly important in the research field of recommendation systems. GCN-based approaches redefine the traditional rating matrix (or interaction matrix) in CF as a user–item bipartite graph and enhance the quality of

recommendations by learning local graph structural features. The traditional GCN technique consists of two main components: aggregation of neighbor nodes and feature extraction by nonlinear neural networks. To the best of our knowledge, GC-MC [15] was the first model to apply GCN to recommendation tasks. Specifically, GC-MC incorporates one-hop neighbor nodes into the embedding representations of target nodes. PinSAGE [16] stacks multiple graph convolution layers to capture information from high-hop neighbor nodes. NGCF [17] is a recommendation framework that combines GCN and MF methods, and concatenates embeddings obtained at each graph convolution layer to capture semantic information at different layers.

A recently proposed assumption about GCN is that the network layers and the nonlinear activation function in GCN are redundant and easily lead to overfitting, which refers to SGCN [18]. Inspired by SGCN, LR-GCCF [19] improves upon NGCF by removing the nonlinear activation function from graph convolution layers and achieves promising improvements. Compared with LR-GCCF, LightGCN [20] removes the activation function and network layers simultaneously from traditional GCN and uses the summation of the embedding outputs from each graph convolution layer as the final representation. More recently, SGL [28] incorporates a contrastive learning module into the linear GCN and jointly learns them, achieving a significant improvement over LightGCN.

V. Conclusion

In this work, we develop a novel recommendation model JMP-GCF, which applies the idea of joint learning to incorporate multigrained popularity features, layer semantics, and high-order interactions into embedding generation. In addition, a multistage stacked training strategy is designed to speed up the model convergence of JMP-GCF. We conduct extensive experiments on three public datasets and achieved the state-of-the-art performance. Further experimental analysis verified the effectiveness and rationality of each component of JMP-GCF.

In the future, we plan to further improve JMP-GCF in three main directions. First, attention networks [40], [41] can be constructed to assign varying weights to different semantics and multigrained popularity features. Second, JMP-GCF can be extended to recommendation scenarios that consider the attribute information of users and items, and these attribute features can be further divided into multiple granularities and learned jointly through the JMP-GCF framework. Finally, we intend to integrate knowledge graph [25], [35] and causal inference [42] into JMP-GCF to further improve interpretability and performance.

References

[1] J. Fang, D. Grunberg, S. Lui, and Y. Wang, "Development of a music recommendation system for motivating exercise," in *Proc. Int. Conf. Orange Technol. (ICOT)*, Dec. 2017, pp. 83–86.

[2] L. Wu, P. Sun, Y. Fu, R. Hong, X. Wang, and M. Wang, "A neural influence diffusion model for social recommendation," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2019, pp. 235–244.

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web (WWW)*, 2001, pp. 285–295.

[4] D. Cao *et al.*, "Cross-platform app recommendation by jointly modeling ratings and texts," *ACM Trans. Inf. Syst.*, vol. 35, no. 4, pp. 1–27, Aug. 2017.

[5] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *Proc. 25th Int. Conf. World Wide Web*, 2016, pp. 507–517.

[6] R. He and J. J. McAuley, "VBPR: Visual Bayesian personalized ranking from implicit feedback," in *Proc. 13th AAAI Conf. Artif. Intell.*, Phoenix, AZ, USA, Feb. 2016, pp. 144–150.

[7] H. Wang, M. Zhao, X. Xie, W. Li, and M. Guo, "Knowledge graph convolutional networks for recommender systems," in *Proc. World Wide Web Conf. (WWW)*, San Francisco, CA, USA, 2019, pp. 3307–3313.

[8] H. Wang *et al.*, "RippleNet: Propagating user preferences on the knowledge graph for recommender systems," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 417–426.

[9] T. Chen, W. Zhang, Q. Lu, K. Chen, Z. Zheng, and Y. Yu, "SVDfeature: A toolkit for feature-based collaborative filtering," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 3619–3622, Dec. 2012.

[10] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2008, pp. 426–434.

[11] S. Kabbur, X. Ning, and G. Karypis, "FISM: Factored item similarity models for top-N recommender systems," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 659–667.

[12] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017, pp. 1–14.

[13] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, Apr./May 2018, pp. 1–12.

[14] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 1024–1034.

[15] R. van den Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," 2017, *arXiv:1706.02263*.

[16] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 974–983.

[17] 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.*, Jul. 2019, pp. 165–174.

[18] 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.

[19] L. Chen, L. Wu, R. Hong, K. Zhang, and M. Wang, "Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach," in *Proc. 34th AAAI Conf. Artif. Intell., AAAI 32nd Innov. Appl. Artif. Intell. Conf. (IAAI) 10th AAAI Symp. Educ. Adv. Artif. Intell. (EAAI)*, New York, NY, USA, Feb. 2020, pp. 27–34.

[20] 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. (SIGIR)*, Jul. 2020, pp. 639–648.

[21] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-N recommendation tasks," in *Proc. 4th ACM Conf. Recommender Syst. (RecSys)*, 2010, pp. 39–46.

[22] H. Steck, "Item popularity and recommendation accuracy," in *Proc. 5th ACM Conf. Recommender Syst. (RecSys)*, 2011, pp. 125–132.

[23] H. Li, J. Liu, B. Cao, M. Tang, X. Liu, and B. Li, "Integrating tag, topic, co-occurrence, and popularity to recommend web Apis for mashup creation," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, Jun. 2017, pp. 84–91.

[24] D. Liang, L. Charlin, J. McInerney, and D. M. Blei, "Modeling user exposure in recommendation," in *Proc. 25th Int. Conf. World Wide Web*, Apr. 2016, pp. 951–961.

[25] X. Wang, X. He, Y. Cao, M. Liu, and T. Chua, "KGAT: Knowledge graph attention network for recommendation," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, Anchorage, AK, USA, Aug. 2019, pp. 950–958.

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

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

[28] J. Wu *et al.*, "Self-supervised graph learning for recommendation," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2021, pp. 726–735.
[29] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.
[30] X. Ning and G. Karypis, "SLIM: Sparse linear methods for Top-N recommender systems," in *Proc. IEEE 11th Int. Conf. Data Mining*, Dec. 2011, pp. 497–506.
[31] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Adv. Artif. Intell.*, vol. 2009, Mar. 2009, Art. no. 421425.
[32] X. Jiang *et al.*, "Novel boosting frameworks to improve the performance of collaborative filtering," in *Proc. Asian Conf. Mach. Learn. (ACML)*, Canberra, ACT, Australia, Nov. 2013, pp. 87–99.
[33] E. Christakopoulou and G. Karypis, "HOSLIM: Higher-order sparse linear method for top-N recommender systems," in *Advances in Knowledge Discovery and Data Mining* (Lecture Notes in Computer Science), vol. 8444, V. S. Tseng, T. B. Ho, Z. H. Zhou, A. L. P. Chen, and H. Y. Kao, Eds. Cham, Switzerland: Springer, 2014, doi: 10.1007/978-3-319-06605-9_4.
[34] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for Youtube recommendations," in *Proc. 10th ACM Conf. Recommender Syst.*, Sep. 2016, pp. 191–198.
[35] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 353–362.
[36] X. Wang, Y. Ye, and A. Gupta, "Zero-shot recognition via semantic embeddings and knowledge graphs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6857–6866.
[37] C. Lei, D. Liu, W. Li, Z.-J. Zha, and H. Li, "Comparative deep learning of hybrid representations for image recommendations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 2545–2553.
[38] W. Chen, F. Cai, H. Chen, and M. D. Rijke, "Joint neural collaborative filtering for recommender systems," *ACM Trans. Inf. Syst.*, vol. 37, no. 4, pp. 1–30, Dec. 2019.
[39] F. Xue, X. He, X. Wang, J. Xu, K. Liu, and R. Hong, "Deep item-based collaborative filtering for top-N recommendation," *ACM Trans. Inf. Syst.*, vol. 37, no. 3, pp. 1–25, Jul. 2019.
[40] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst., Annu. Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 5998–6008.
[41] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T. Chua, "Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Tokyo, Japan, Aug. 2017, pp. 335–344.
[42] S. Bonner and F. Vasile, "Causal embeddings for recommendation," in *Proc. 12th ACM Conf. Recommender Syst.*, Sep. 2018, pp. 104–112.

Kang Liu received the B.S. degree from the China University of Geosciences, Wuhan, China, in 2017. He is currently pursuing the M.S. and Ph.D. degrees with the Hefei University of Technology, Hefei, China.

His research interests include recommendation systems, data mining, and multimedia analysis.

Feng Xue received the Ph.D. degree from the Department of Computer Science, Hefei University of Technology (HFUT), Hefei, China, in June 2006.

He is currently a Professor with HFUT. His current research interests are in artificial intelligence, multimedia analysis, and recommendation systems.

Xiangnan He (Member, IEEE) received the Ph.D. degree in computer science from the National University of Singapore (NUS), Singapore, in April 2016.

He is currently a Professor with the University of Science and Technology of China (USTC), Hefei, China. He has over 100 publications that appeared in top conferences, such as ACM Special Interest Group on Information Retrieval (SIGIR), International World Wide Web Conferences (WWW), and ACM Knowledge Discovery and Data Mining (KDD), and journals, including IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE), *ACM Transactions on Information Systems* (TOIS), and IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS (TNNLS). His research interests span information retrieval, data mining, and multimedia analytics.

Dr. He has served as an SPC/PC Member for several top conferences, including SIGIR, WWW, KDD, ACM Multimedia (MM), ACM International Conference on Web Search and Data Mining (WSDM), and International Conference on Machine Learning (ICML). His work received the Best Paper Award Honorable Mention in WWW 2018 and the Association for Computing Machinery (ACM) SIGIR 2016. He also serves as an Associate Editor for *ACM TOIS*, *Frontiers in Big Data*, *AI Open*, and so on. He has served as the PC Chair of IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS) 2019 and a Regular Reviewer for journals, including TKDE and TOIS.

Dan Guo (Member, IEEE) received the Ph.D. degree in system analysis and integration from the Huazhong University of Science and Technology, Wuhan, China, in 2010.

She is currently a Professor with the School of Computer and Information, Hefei University of Technology, Hefei, China. Her research interests include computer vision, machine learning, and intelligent multimedia content analysis.

Richang Hong (Member, IEEE) received the Ph.D. degree from the University of Science and Technology of China, Hefei, China, in 2008.

He was a Research Fellow with the School of Computing, National University of Singapore, Singapore, from 2008 to 2010. He is currently a Professor with the Hefei University of Technology, Hefei. He has coauthored more than 100 publications in the areas of his research interests, which includes multimedia content analysis and social media.

Dr. Hong is also a member of the Association for Computing Machinery (ACM) and an Executive Committee Member of the ACM SIGMM China Chapter. He was a recipient of the Best Paper Award in the ACM Multimedia 2010, the Best Paper Award in the ACM ICMR 2015, and the Honorable Mention of the IEEE TRANSACTIONS ON MULTIMEDIA Best Paper Award. He has served as the Technical Program Chair of the International Conference on Multimedia Modeling (MMM) 2016. He has served as an Associate Editor for the *IEEE Multimedia Magazine*, *Information Sciences*, and *Signal Processing* (Elsevier).