Multiple Hypergraph Learning for Ephemeral Group Recommendation

Rui Zhao[1,2], Beihong Jin[1,2(✉)], Yimin Lv[1,2], Yiyuan Zheng[1,2], and Weijiang Lai[1,2]

[1] Key Laboratory of System Software (Chinese Academy of Sciences) and State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China

[2] University of Chinese Academy of Sciences, Beijing, China

Beihong@iscas.ac.cn

Abstract. Ephemeral Group Recommendation (EGR) refers to recommending items for a temporarily existing group, where the ephemeral group has little or no historical interactions with items while each group member has his/her own interaction history. We note that EGR not only faces the challenge of extremely sparse or nonexistent group-item interactions and also has its own special needs. EGR needs to seek the common preferences of the members instead of maximizing the personalized needs of individuals. In particular, group preferences may not necessarily be related to the timeliness and intensity of the member's individual behavior and preferences. Following this line of thought, we propose an EGR model named HL4EGR. Specifically, we adopt hypergraphs to model complex relationships among users, items, and groups, during which we weaken the timeliness and intensity of user behavior and preferences and augment training data by discovering implicit and explicit group-group similarities. Moreover, we design a cross-hypergraph contrastive learning strategy to align embeddings for the same group in different hypergraphs, which enables group preferences to reflect the common preferences of group members comprehensively. We conduct extensive experiments on three real-world datasets, and the experimental results show that our model HL4EGR outperforms state-of-the-art models.

Keywords: Recommender Systems · Group Recommendation · Hypergraph Neural Networks · Contrastive Learning

1 Introduction

In recent years, recommender systems have been offering personalized item recommendations on online services. With the increasing social activities of users, providing group recommendations on online services is poised to become a new and viable pathway to attract users and boost user engagement continuously.

The group recommendation is to recommend items of common interests (such as dining restaurants, travel destinations, and gathering venues) for a group of

A. Bifet et al. (Eds.): ECML PKDD 2024, LNAI 14949, pp. 89–105, 2024.
https://doi.org/10.1007/978-3-031-70378-2_6

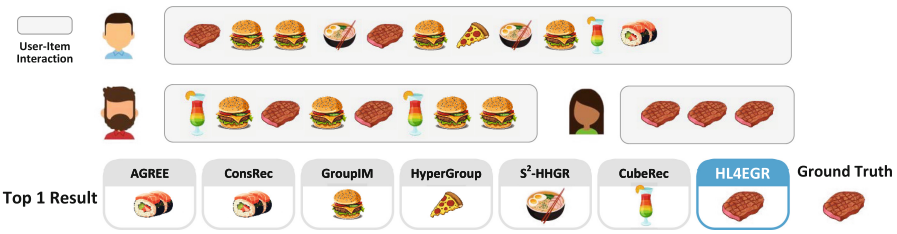

Fig. 1. Case study. The icon denotes the type of restaurant. The sequence of icons in a rectangle denotes the interaction history of the person on the left side.

members, often taking group preferences as a guideline [9,10,13,19]. Depending on the way a group is established, group recommendation can be divided into two categories: persistent group recommendation (PGR) and ephemeral group recommendation (EGR). The former is for a group with fixed members having long-term and extensive interactions between the group and items. The latter is for a temporarily formed group without fixed members, where the group has little or no historical interactions with items, making it impossible to directly learn from those interactions. We focus on EGR in this paper.

We note that the group recommendation, either PGR or EGR, should adhere to the norm of seeking common ground, which facilitates the smooth progress of collective activities. Specifically, the group recommendation models are supposed to treat the common preferences of all the members as the group preferences, and seek factors behind the consensus among group members. In particular, the model should treat the historical and current preferences of a group member equally, without weakening the role of the group member's historical preferences over time. This stems from real-life experience: if a member's historical preferences are the same as the current preferences of the other members in a group, even if the member's current preferences have deviated from his/her historical preferences, the member may still reach a compromise with the other members, accepting the item that is consistent with his/her historical preferences. Moreover, the model are expected to treat the strong preferences of one member and the weak preferences of another member equally, and should not be misled by the strong preferences exhibited by one or a few members.

Taking a real ephemeral group from the Yelp dataset as an example. As depicted in Fig. 1, the group consists of three persons. The person on the top has wide-ranging interests, where steak is his early preference. The person on the middle left has interest in steak and hamburger, and the person on the middle right has interest in steak only. Finally, this group actually visits a steakhouse, a restaurant that is acceptable to all three of them. Unfortunately, existing EGR models fail to obtain the correct result. For the case in the Fig. 1, GroupIM [14] and CubeRec [6] recommend a burger shop and a dessert shop, respectively. It seems that these two models are influenced by the strong personal preferences of two group members. S^2-HHGR [20] and HyperGroup [8] recommend a noodle shop and a pizza shop, respectively. These two models seem to ignore historical

preferences of members and be influenced by the user with wide-ranging preferences. In addition, two PGR models, i.e., AGREE [4] and ConsRec [16], also give incorrect results.

To realize "seeking common ground" in group recommendations and figure out a feasible solution to the inherent problem in EGR, that is, group-item interactions are extremely sparse or nonexistent, we propose a multi-hypergraph model named HL4EGR (Hypergraph Learning for Ephemeral Group Recommendation). The model employs hypergraphs to capture the relationship among users, items, and groups, and adopts a two-stage framework consisting of pre-training and fine-tuning. During the pre-training, we choose the hypergraph to model user-item interactions, where a hyperedge connects all the items that a user interacts with, thus equally treating historical interactions and current interactions. The item embeddings obtained by the pre-training are subsequently clustered to identify user preferences. At the stage of fine-tuning, we construct three hypergraphs to model user-group affiliations and two types of group-group similarities, respectively. Here, two types of similarities are given from two different perspectives, one explicit from the perspective of items interacted with by the members and another implicit from the perspective of common preferences of members. Both of them emphasize the commonality of member behavior or member preferences, and weaken the intensity of member behavior or preferences. Further, we maximize the agreement between contrastive views of groups by cross-hypergraph contrastive learning. Finally, we aggregate these group embeddings to generate group preferences and then perform the prediction for groups. For the case in Fig. 1, our model recommends a steakhouse that is in line with the ground truth. Our contributions are summarized as follows.

- We construct four hypergraphs and learn the complex relationships among users, items, and groups through hypergraph convolutions. Particularly, by means of hypergraphs, we weaken the timeliness and intensity of user behavior and preferences and captures their common preferences effectively, thus satisfying the intrinsic requirement of group recommendation.
- We highlight that identifying and leveraging similarities between groups provides a practicable way to cope with the absence of group-item interactions. Moreover, we take group self-discrimination as the self-supervised task, which offers auxiliary supervision signals via two views of a group w.r.t. explicit and implicit group-group similarities for reinforcing group representation learning.
- We conduct extensive experiments on three public datasets. The experimental results show that HL4EGR consistently outperforms the state-of-the-art models, showing relative gains of 8.92%-15.93% on Recall@50 and 13.37%-18.88% on NDCG@50, respectively.

2 Related Work

Early group recommendation adopts collaborative filtering to obtain the member's scores on items and then aggregates their scores to get group preferences by some hand-crafted heuristic rules. Customary aggregation methods include the

least misery [1], the average [3], and the maximum satisfaction strategy [2]. However, these predefined aggregation strategies lack the flexibility to achieve optimal performance in group recommendation. Subsequent work on group recommendation shifts towards how to efficiently aggregate the preference representations of all group members to the group preference. For example, multiple group recommendation models such as AGREE [4], SoAGREE [5] and MoSAN [15] propose different attention-based aggregation methods.

With the development of graph neural networks, the tripartite graph [12] has been employed to model users, items, and groups relationships and then learn group representations. Furthermore, hypergraphs are found to be more suitable for modeling groups because hyperedges in the hypergraph can connect two or more nodes and represent a more general topological relationship. Some models [8,10,16,20] apply the hypergraph to model groups and then employ Hypergraph Neural Networks (HNNs) [17] to generate group representations. For example, ConsRec [16] models users and items as nodes, groups as hyperedges, and learns group representations through HNNs. In addition, CubeRec [6] adaptively generates a hypercube representation for each group. However, these models do not discover the essence of user preferences playing a role in group recommendation scenarios. They do not treat strong and weak preferences equally, nor do they give equal weight to historical and current preferences.

Recently, the research on group recommendation [6,14,20] has attempted to incorporate self-supervised learning to alleviate the data sparsity problem. For example, for enhancing the user and group representations, GroupIM [14] proposes maximizing mutual information between members within a group and the group. S^2-HHGR [20] designs a double-scale node dropout strategy and performs node self-discrimination on different user representations. However, existing methods mainly focus on finding self-supervision signals in user-group relationships without considering group-group relationships. Besides, some studies rely on introducing additional information to improve the performance of group recommendations. For example, KGAG [7] introduces knowledge graphs into group recommendation. SIGR [18] and HyperGroup [8] introduce social relationships among users to learn group preferences influenced by social relationships.

Compared to existing work, our model employs multiple hypergraphs to model different relationships among users, items, and groups from multiple perspectives, using the prior about the role of user preferences for group recommendations as an inductive bias of the model. Moreover, our model captures self-supervision signals from the similarities between groups, and then learn more comprehensive group representations.

3 Methodology

3.1 Model Overview

Let $\mathcal{U}$, $\mathcal{V}$ and $\mathcal{G}$ denote the user set, item set, and ephemeral group set, respectively. An ephemeral group $g_k \in \mathcal{G}$ consists of $|g_k|$ users, i.e., $g_k = \{u_i^{g_k}\}_{i=1}^{|g_k|}$,

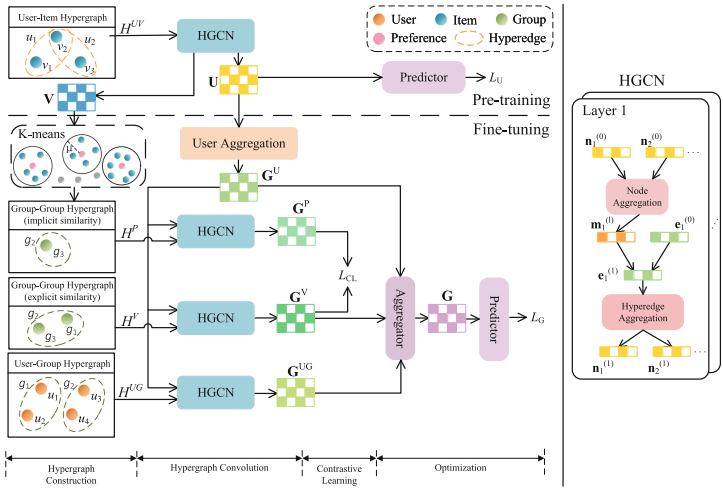

Fig. 2. Architecture of our HL4EGR.

where $u_i^{g_k} \in \mathcal{U}$. There are two types of observed interactions among users, items, and ephemeral groups, i.e., user-item interactions denoted as $\mathbf{X} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{V}|}$, and group-item interactions represented as $\mathbf{Y} \in \mathbb{R}^{|\mathcal{G}|\times|\mathcal{V}|}$, where the element y_{kj} of the matrix $\mathbf{Y}$ is equal to 1 if the group g_k has historical interactions with the item v_j, otherwise $y_{kj} = 0$.

Given an ephemeral group g_k, our task is to predict the item that the group g_k is most likely to be satisfied with.

For this task, we propose a multi-hypergraph model HL4EGR, whose architecture is shown in Fig. 2. We build four hypergraphs to model the user-item interactions, the user-group affiliations, and the explicit and implicit group-group similarities.

As shown in Fig. 2, the training of HL4EGR is divided into two stages, i.e., pre-training and fine-tuning.

In the first stage, we construct the user-item hypergraph H^{UV} and perform the convolution operation on H^{UV}, thus obtaining the user embeddings $\mathbf{U}$ and item embeddings $\mathbf{V}$. $\mathbf{U}$ is utilized for initializing the group embeddings used in the second stage and $\mathbf{V}$ is applied to characterize the user preferences by clustering.

In the second stage of training, i.e., fine-tuning, except for constructing the user-group hypergraph H^{UG}, we also construct two group-group hypergraphs H^V and H^P, portraying explicit and implicit similarities between groups, respectively. Then we perform the hypergraph convolution operations to obtain group embeddings. Furthermore, we adopt a cross-hypergraph contrastive learning strategy to align embeddings of the same group from both explicit and implicit perspectives, thus obtaining more comprehensive group preferences that are devoted to group recommendation.

3.2 Hypergraph Construction

User-Item Hypergraph . We define the user-item hypergraph as $H^{UV} = (\mathcal{V}, \mathcal{E}^{UV})$, where a node of H^{UV} is an item in $\mathcal{V}$, a hyperedge $e_i^{UV} \in \mathcal{E}^{UV}$, $i \in [1, |\mathcal{U}|]$ connects all the items that user u_i interacts with, and $|\mathcal{E}^{UV}| = |\mathcal{U}|$. As shown in Fig. 2, user u_1 has historical interactions with item v_1 and item v_2, thus we connect $\{v_1, v_2\}$ with a hyperedge. Such hyperedges eliminate temporal differentiations of interactions, treating historical interactions and current interactions equally.

User-Group Hypergraph . We define the user-group hypergraph as $H^{UG} = (\mathcal{U}, \mathcal{E}^{UG})$, where a node of H^{UG} is a user in $\mathcal{U}$, a hyperedge $e_k^{UG} \in \mathcal{E}^{UG}$, $k \in [1, |\mathcal{G}|]$ connects all the users in group g_k, and $|\mathcal{E}^{UG}| = |\mathcal{G}|$. As shown in Fig. 2, we connect the group member $\{u_1, u_2\}$ of the group g_1 with a hyperedge, which reflects the user-group affiliation.

Group-Group Hypergraphs . For alleviating the data sparsity issue, we construct two group-group hypergraphs, i.e., $H^V = (\mathcal{G}, \mathcal{E}^V)$ and $H^P = (\mathcal{G}, \mathcal{E}^P)$.

In hypergraph H^V, $\mathcal{G}$ is taken as the node set, and a hyperedge $e_k^V \in \mathcal{E}^V$, $k \in [1, |\mathcal{G}|]$ connects all such groups, provided that a member of that group and a member of group g_k interact with the same item. In other words, hypergraph H^V contains the explicit similarities between groups.

Complementary to H^V, hypergraph H^P implies the implicit similarities between groups, i.e., the preference similarities between groups. The group preference is essentially a collection of member preferences. Specifically, with the consideration of the interference of noisy behavior, we regard the items that all users have interacted with as the starting point to model the user's preferences, instead of capturing the user's preferences from a user's behavior. We perform K-means clustering on the item embeddings $\mathbf{V}$ obtained by the pre-training on the hypergraph H^{UV} and generate c clustering centers. Next, for each user, given an item that this user has interacted with, if the distance between the item embedding and the center of the category the item belongs to is less than μ, this center is considered to be a preference of this user. Subsequently, the preferences of group members are merged to form a set of group preferences. Then, we build a hyperedge $e_k^P \in \mathcal{E}^P$, $k \in [1, |\mathcal{G}|]$ to connect the groups that has common preferences with group g_k.

When building the group-group hypergraphs, we treat all the hyperedges equally (i.e., hyperedges with same weights), thus flattening the intensity of a user's individual behavior and preferences. This enables HL4EGR to more fairly learn the common preferences of users within the group, reducing the impact of the intensity of a user's personal behavior and preferences on the group-group similarity.

3.3 Hypergraph Convolution

In HL4EGR, we design a HyperGraph Convolutional Network (HGCN) to learn representations of nodes and hyperedges in a hypergraph. Without loss of generality, we formalize four hypergraphs uniformly as $H = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ denotes the node set and $\mathcal{E}$ denotes the hyperedge set. The learning process of the l-th layer of HGCN is as follows.

Firstly, we aggregate representations of all nodes connected by hyperedge e_k as follows.

$$\mathbf{m}_k^{(l)} = \mathrm{AGG}(\mathbf{n}_i^{(l-1)} | n_i \in e_k) \tag{1}$$

where $e_k \in \mathcal{E}$ denotes the k-th hyperedge, $\mathbf{n}_i^{(0)}$ is the initial embedding of node $n_i \in \mathcal{N}$, $\mathbf{n}_i^{(l-1)}$ is the embedding of the node n_i in the $(l-1)$-th layer, $\mathrm{AGG}(\cdot)$ denotes an aggregation function, realized as an average pooling function.

Then, we concatenate node aggregation representation $\mathbf{m}_k^{(l)}$ and hyperedge representation $\mathbf{e}_k^{(l-1)}$ to update the hyperedge representation as follows.

$$\mathbf{e}_k^{(l)} = \mathrm{CONCAT}(\mathbf{m}_k^{(l)}, \mathbf{e}_k^{(l-1)})\mathbf{W}^H \tag{2}$$

where $\mathbf{e}_k^{(0)}$ is the initial embedding of hyperedge e_k, $\mathbf{e}_k^{(l)}$ denotes the embedding of the hyperedge e_k in the l-th layer. $\mathbf{W}^H \in \mathbb{R}^{2d \times d}$ is a learnable matrix.

Moreover, node representations can be updated as follows.

$$\mathbf{n}_i^{(l)} = \mathrm{AGG}(\mathbf{e}_k^{(l)} | e_k \in \mathcal{E}_i) \tag{3}$$

where $\mathcal{E}_i$ represents the set of hyperedges connected to the node n_i.

Finally, we can obtain the embedding $\mathbf{n}_i$ of the node n_i, and the embedding $\mathbf{e}_k$ of the hyperedge e_k as follows.

$$\mathbf{n}_i = \sum_{l=1}^{L} \mathbf{n}_i^{(l)}, \quad \mathbf{e}_k = \sum_{l=1}^{L} \mathbf{e}_k^{(l)} \tag{4}$$

where L is the number of convolutional layers.

During pre-training, we first randomly initialize the representations of nodes and hyperedges in H^{UV} and feed them into an HGCN. Then, we iterate and optimize the HGCN by the cross entropy loss (L_U in Fig. 2). After pre-training, we obtain user embeddings $\mathbf{U} \in \mathbb{R}^{|\mathcal{U}| \times d}$ and item embeddings $\mathbf{V} \in \mathbb{R}^{|\mathcal{V}| \times d}$.

During fine-tuning, we first aggregate user embeddings $\mathbf{U}$ to generate group embeddings $\mathbf{G}^U \in \mathbb{R}^{|\mathcal{G}| \times d}$. In detail, taking the group g_k as an example, we leverage an attention mechanism to aggregate embeddings of users in the group g_k, thereby obtaining the initial group representation $\mathbf{g}_k^U = \mathbf{G}^U(k, :)$. This process can be formalized as follows.

$$\mathbf{g}_k^U = \sum_{u_i \in g_k} \alpha_i \mathbf{u}_i \tag{5}$$

$$\alpha_i = \frac{\exp(tanh(\mathbf{u}_i \mathbf{W}^{AGG} + b))}{\sum_{u_{i'} \in g_k} \exp(tanh(\mathbf{u}_{i'} \mathbf{W}^{AGG} + b))} \tag{6}$$

where $\mathbf{u}_i = \mathbf{U}(i,:)$ denotes the embedding of the user u_i obtained by pre-training, α_i is the attention weight w.r.t. the user u_i. $\mathbf{W}^{AGG} \in \mathbb{R}^d$ is a learnable vector and b is a bias.

Next, we use $\mathbf{G}^U$ to initialize hyperedges of H^{UG}, H^V and H^P, and nodes of H^V and H^P, and use user embeddings $\mathbf{U}$ obtained by the pre-training to initialize node representations on the hypergraph H^{UG}, and then feed them into corresponding HGCNs.

Finally, by performing the calculations over these three HGCNs, we obtain representations of hyperedges in three hypergraphs, denoted as $\mathbf{G}^{UG}$, $\mathbf{G}^V$, and $\mathbf{G}^P$, respectively. Given group g_k, its embeddings from three hypergraphs are $\mathbf{g}_k^{UG} = \mathbf{G}^{UG}(k,:)$, $\mathbf{g}_k^V = \mathbf{G}^V(k,:)$, and $\mathbf{g}_k^P = \mathbf{G}^P(k,:)$, respectively.

3.4 Cross-Hypergraph Contrastive Learning

To learn more comprehensive group preferences, we design a contrastive learning strategy on two group-group hypergraphs, i.e., the hypergraph H^V reflecting explicit similarities and the hypergraph H^P implying implicit similarities, aligning two embeddings of the same group in H^V and H^P. Concretely, we regard the representations w.r.t. the same group in two hypergraphs H^V and H^P as positive sample pairs. The representations w.r.t. different groups in the same batch in two hypergraphs H^V and H^P are considered as negative sample pairs. We take InfoNCE loss as the contrastive learning loss as follows.

$$L_{CL} = -\sum_{g_k \in \mathcal{G}} \log \frac{\exp(sim(\mathbf{g}_k^V, \mathbf{g}_k^P)/\tau)}{\exp(sim(\mathbf{g}_k^V, \mathbf{g}_k^P)/\tau) + N_V + N_P} \tag{7}$$

$$N_V = \sum_{g_{k'} \in \mathcal{G}_k^-} \exp(sim(\mathbf{g}_{k'}^V, \mathbf{g}_k^P)/\tau), \quad N_P = \sum_{g_{k'} \in \mathcal{G}_k^-} \exp(sim(\mathbf{g}_k^V, \mathbf{g}_{k'}^P)/\tau) \tag{8}$$

where $\mathbf{g}_k^V$ and $\mathbf{g}_k^P$ form a pair of positive samples, corresponding to the representations of the group g_k in the hypergraph H^V and H^P, respectively. $\mathcal{G}_k^-$ is the set of negative samples w.r.t. the group g_k, which is composed of other groups (i.e., $k' \neq k$) within the same batch. $sim(\cdot)$ function is adopted for calculating the similarity of a pair of vectors, which refers to the cosine similarity in this paper. τ is the temperature parameter.

3.5 Model Optimization

During pre-training, we predict the interaction probabilities $\hat{\mathbf{x}}_\mathbf{i} \in R^{|\mathcal{V}|}$ of user u_i on the item set $\mathcal{V}$ as follows.

$$\hat{\mathbf{x}}_\mathbf{i} = softmax(\mathbf{u}_i \mathbf{W}^{UV}) \tag{9}$$

where $\mathbf{u}_i = \mathbf{U}(i,:)$obtained from hypergraph H^{UV}, $\mathbf{W}^{UV} \in \mathbb{R}^{d \times |\mathcal{V}|}$ is a learnable matrix.

Then we calculated the cross entropy loss L_U as follows.

$$L_U = -\frac{1}{|\mathcal{U}|}\sum_{i=1}^{|\mathcal{U}|}\sum_{j=1}^{|\mathcal{V}|} x_{ij}\log \hat{x}_{ij} \tag{10}$$

where $\hat{x}_{ij}$ refers to the interaction probability of the user u_i w.r.t. the item v_j. x_{ij} is the ground truth of user-item interaction.

During fine-tuning, given group representations from different hypergraphs, we adopt an adaptive aggregation strategy to fuse different group embeddings, i.e., $\mathbf{g}_k^U$ obtained from Eq. 5, $\mathbf{g}_k^{UG}$ in the hypergraph H^{UG}, and $\mathbf{g}_k^V$ in the hypergraph H^V, to generate the group preference $\mathbf{g}_k$ for the group g_k as follows.

$$\mathbf{g}_k = \alpha\mathbf{g}_k^U + \beta\mathbf{g}_k^{UG} + \gamma\mathbf{g}_k^V \tag{11}$$

where $\alpha = \sigma(\mathbf{g}_k^U\mathbf{W}^U)$, $\beta = \sigma(\mathbf{g}_k^{UG}\mathbf{W}^{UG})$, and $\gamma = \sigma(\mathbf{g}_k^V\mathbf{W}^V)$. $\mathbf{W}^U$, $\mathbf{W}^{UG}$, and $\mathbf{W}^V \in \mathbb{R}^d$ are learnable matrices. σ is the sigmoid activation function.

We predict the interaction probabilities $\hat{\mathbf{y}}_k \in \mathbb{R}^{|\mathcal{V}|}$ of the group g_k on the item set $\mathcal{V}$ as follows.

$$\hat{\mathbf{y}}_k = softmax(\mathbf{g}_k\mathbf{W}^{GV}) \tag{12}$$

where $\mathbf{W}^{GV} \in \mathbb{R}^{d\times|\mathcal{V}|}$ is a learnable matrix.

Then, we adopt the cross entropy loss as the main loss, calculated as follows.

$$L_G = -\frac{1}{|\mathcal{G}|}\sum_{k=1}^{|\mathcal{G}|}\sum_{j=1}^{|\mathcal{V}|} y_{kj}\log \hat{y}_{kj} \tag{13}$$

where $\hat{y}_{kj}$ refers to the interaction probability of the group g_k w.r.t. the item v_j. y_{kj} is the ground truth.

We adopt a multi-task strategy to jointly optimize the main group recommendation task and the auxiliary contrastive learning task as follows.

$$L = L_G + \lambda L_{CL} \tag{14}$$

where λ is a hyperparameter.

3.6 Complexity Analysis

Space Complexity. In HL4EGR, the learnable parameters are mainly from embeddings of users, items, and groups. In addition, as for hypergraph convolutions, since we have four hypergraphs in two stages, each with L layers, the number of parameters is $4L \times 2d^2$. The number of parameters for two prediction layers in two stages is $2|\mathcal{V}|d$. Thus, the space complexity of HL4EGR is $O(Ld^2 + |\mathcal{U}|d + |\mathcal{V}|d + |\mathcal{G}|d)$.

Time Complexity. The computation amount of HL4EGR is mainly concentrated on the hypergraph convolutions. Let $|H|$ be the number of nonzero elements in the adjacency matrix of hypergraph H. The time complexity of each hypergraph convolution computation is $O(L \times (2|H|d + 2|\mathcal{E}|d^2))$, where $|\mathcal{E}|$ is the number of hyperedges. For hypergraphs H^{UV}, H^{UG}, H^V, and H^P, the numbers of hyperedges are $|\mathcal{U}|$, $|\mathcal{G}|$, $|\mathcal{G}|$, and $|\mathcal{G}|$, respectively. The total time complexity of HL4EGR is $O(Ld^2|\mathcal{G}| + Ld^2|\mathcal{U}| + Ld(|H^{UV}| + |H^{UG}| + |H^V| + |H^P|))$.

Table 1. Statistics of datasets.

Dataset	# Users	# Items	# Groups	# User-Item interactions	# Group-Item interactions	Avg. of Group-Item interactions	Avg. group size
Weeplaces	8,643	25,081	22,733	508,486	67,133	2.95	2.89
Yelp	34,504	22,611	24,103	482,273	26,883	1.18	4.45
Douban	25,377	21,696	72,962	2,817,094	118,022	1.62	5.23

4 Experiments

4.1 Experimental Settings

Datasets. We conduct experiments on three public datasets.

- **Weeplaces**. It records users' check-ins in location-based social networks. We extract check-ins from points of interest (POIs) in all major cities in the U.S. We follow the same operations as in GroupIM [14] for constructing user-POI interactions and group-POI interactions.
- **Yelp**. It records users' check-ins in local businesses (e.g., restaurants). We use the dataset published in [18], which includes users' check-ins on businesses located in Los Angeles, as well as groups' check-in information.
- **Douban**. It is also published in [18], recording the information of users organizing and participating in social activities. We filter out users and items with fewer than 10 interactions.

Table 1 lists the statistics of the three datasets. As shown in Table 1, the average of group-item interactions is less than 3, which manifests that we conduct experiments on ephemeral groups. We randomly split all the groups of each dataset into training, validation, and test sets with a ratio of 7:1:2. We ensure that each group can only appear in one of the three sets.

Baselines. We compare HL4EGR to the following baselines:

- Two PGR models: **AGREE**[1], which is a classical PGR model using an attention mechanism for member aggregation [4]. **ConsRec**[2], the state-of-the-art

[1] https://github.com/LianHaiMiao/Attentive-Group-Recommendation.
[2] https://github.com/FDUDSDE/WWW2023ConsRec.

model for PGR, which proposes an HNN to learn member-level aggregation and captures the group consensus on three views [16].

- Four EGR models: **GroupIM**[3], which maximizes user-group mutual information for group recommendation [14]. **HyperGroup**[4], which models groups as hyperedges to learn group representations [8]. **S²-HHGR**[5], which uses a hierarchical hypergraph and a node dropout strategy on the hypergraph to learn group preferences [20]. **CubeRec**[6], the state-of-the-art model for EGR, which utilizes the geometric expressiveness of hypercubes and hypercube intersection-based self-supervision to obtain the group representations [6].

Table 2. Overall performance. The values in bold and underlined are the best and second best results in each row.

Weeplaces

Metric	AGREE	ConsRec	GroupIM	HyperGroup	S²-HHGR	CubeRec	HL4EGR	Improv.(%)
R@20	0.1612	0.1949	<u>0.4132</u>	0.3926	0.3930	0.4119	**0.4776**	15.58
R@50	0.2927	0.3686	0.5688	0.5438	0.5337	<u>0.5719</u>	**0.6367**	11.33
N@20	0.0637	0.0731	<u>0.2218</u>	0.1720	0.1794	0.2162	**0.2549**	14.92
N@50	0.0913	0.1062	<u>0.2528</u>	0.1964	0.1958	0.2485	**0.2866**	13.37

Yelp

Metric	AGREE	ConsRec	GroupIM	HyperGroup	S²-HHGR	CubeRec	HL4EGR	Improv.(%)
R@20	0.1940	0.2236	0.5548	0.5247	0.5276	<u>0.5802</u>	**0.6647**	14.56
R@50	0.3388	0.3753	0.6603	0.6273	0.6158	<u>0.6995</u>	**0.7619**	8.92
N@20	0.0898	0.1025	0.3743	0.3408	0.3580	<u>0.3790</u>	**0.4503**	18.81
N@50	0.1168	0.1319	0.3953	0.3612	0.3719	<u>0.4027</u>	**0.4696**	16.61

Douban

Metric	AGREE	ConsRec	GroupIM	HyperGroup	S²-HHGR	CubeRec	HL4EGR	Improv.(%)
R@20	0.1236	0.1471	0.2184	0.2103	0.2014	<u>0.2244</u>	**0.2771**	23.48
R@50	0.2557	0.2913	0.3634	0.3574	0.3371	<u>0.3715</u>	**0.4307**	15.93
N@20	0.0455	0.0550	0.0886	0.0841	0.0838	<u>0.0926</u>	**0.1142**	23.32
N@50	0.0728	0.0841	0.1173	0.1131	0.1106	<u>0.1218</u>	**0.1448**	18.88

Implementation Details. We implement our model in PyTorch. In our model, the number of hypergraph convolutional layers L is set to 2 and temperature τ is set to 1. We tune the weight of contrastive learning loss λ, the number of clustering centers c, the threshold of distance to any clustering center μ for every dataset, finally setting λ to 0.3, μ to 0.2 for all datasets, c to 64 for Weeplaces, 128 for Yelp and Douban. We optimize the model via the Adam optimizer with

[3] https://github.com/CrowdDynamicsLab/GroupIM.
[4] https://github.com/FDUDSDE/WWW2023GroupRecBaselines.
[5] https://github.com/0411tony/HHGR.
[6] https://github.com/jinglong0407/CubeRec.

the learning rate 0.001. The implementation code has been released[7]. For the sake of fairness, we set the size of all embeddings d to 64, the batch size to 256 in all the experiments. For all baselines, the hyperparameters are set to values corresponding to best performance reported in their respective papers. Experiments are conducted on NVIDIA RTX3090 GPU with 24G memory.

Metrics. To evaluate the performance of recommending items to groups, we adopt two metrics, i.e., Recall@K and NDCG@K (R@K and N@K for short), where Recall focuses on whether the group actually chooses the recommended item, NDCG focuses on the ranking of the recommended items and K is set to either 20 or 50.

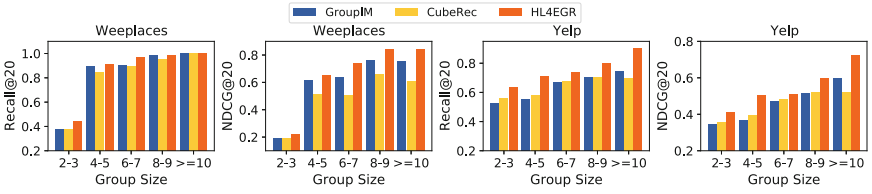

Fig. 3. Group recommendation performance on groups of different sizes.

4.2 Performance Comparison

Overall Performance. Table 2 lists the experimental results of our proposed model and compared models on the three datasets. From Table 2, we have the following observations.

- The PGR models are far inferior to the EGR models in all metrics. This is because PGR models depend on group-item interactions to learn group preferences; however, these interactions become extremely sparse or nonexistent in the context of ephemeral groups, ultimately leading to a decline in performance.
- Hypergraph-based models, i.e., ConsRec, HyperGroup, S^2-HHGR, and HL4EGR outperform the traditional attention-based model, i.e., AGREE, which demonstrates that the hypergraph structure excels in modeling user-group affiliations.
- Three EGR models equipped with self-supervised learning, i.e., GroupIM, CubeRec, and HL4EGR, outperform other models. This might be attributed to the fact that these EGR models can discover and utilize additional supervision signals, thus improving the quality of group embeddings. This shows the advantages of self-supervised learning in EGR.

[7] https://github.com/ZhaoRui-7/HL4EGR.

- Our HL4EGR outperforms all baselines on three datasets. Taking Recall@20 as an example, compared to the best baseline on each of the three datasets, HL4EGR shows improvements of 14.56% - 23.48%, averaging at 17.87%.

Performance on Groups of Different Sizes. We split the test set into five subsets by the range of the number of group members, i.e., 2-3, 4-5, 6-7, 8-9, and >=10 members. We choose GroupIM and CubeRec for comparison because they are the top-2 best baselines, and we conduct experiments on Weeplaces and Yelp datasets.

As shown in Fig. 3, HL4EGR outperforms GroupIM and CubeRec in almost all cases, except on groups of 10 or more members in Weeplaces, where all three models have the same Recall values (reaching the maximum value of 1). In particular, HL4EGR outperforms the other two models for the case of groups of 2-3 members, indicating that HL4EGR is more suitable for real-life group recommendations, where the size of groups shows the long-tail distribution. Meanwhile, compared to other two models, HL4EGR has more significant performance gains for groups of 10 or more members in Yelp. The reason behind might be that HL4EGR can correct group representations by treating all behavior and preferences of all members indiscriminately in terms of timeliness and intensity, thus capturing common preferences of groups more accurately, while the number of group members increases.

Table 3. Ablation study.

(a) Effect of multiple hypergraphs

Model	Weeplaces		Yelp	
	R@20	N@20	R@20	N@20
HL4EGR	**0.4776**	**0.2549**	**0.6647**	**0.4503**
(A) w/o H^{UV}	0.3674	0.1916	0.4929	0.3233
(B) w/o H^{P}	0.4674	0.2442	0.6542	0.4335
(C) w/o H^{V}	0.4585	0.2421	0.6377	0.4226
(D) w/o H^{UG}	0.4716	0.2498	0.6546	0.4384

(b) Effect of pre-training or weights

Model	Weeplaces		Yelp	
	R@20	N@20	R@20	N@20
(E)	0.3079	0.1651	0.4596	0.3093
(F)	0.4009	0.2201	0.5634	0.4052
(G)	0.4644	0.2419	0.6546	0.4401
(H)	0.4715	0.2501	0.6563	0.4372
(I)	0.4770	0.2500	0.6630	0.4466

4.3 Ablation Study

Effect of Multiple Hypergraphs. We design four variants to observe the effect of four hypergraphs in HL4EGR on the performance. Variant A deletes the HGCN on H^{UV} but also performs the pre-training, taking randomly initialized user and item embeddings as input and cross entropy loss, i.e., L_U as the optimization goal. Variant B removes the HGCN on H^P, which triggers a cascading removal of the contrastive learning module, since H^P is treated as a

source of self-supervision signals. Variant C removes the HGCN on H^V, which leads to the removal of contrastive learning module as well as the reduction of one source of the group preference. Variant D eliminates the HGCN on H^{UG}.

The experimental results on Weeplaces and Yelp are listed in Table 3(a). Compared to HL4EGR, all variants show different degrees of performance degradation, illustrating that each hypergraph is effective. Variant A shows the notable performance degradation, indicating that hypergraph H^{UV} is the underpinning of the whole model. The direct reason behind this is that the user and item embeddings derived from H^{UV} are subsequently used for the construction and learning of other hypergraphs, which imposes a great positive impact on performance. The performance decrease of variant B shows the effect of alleviating data sparsity and adjusting group embeddings via contrastive learning. Variant C has a significant decline in performance, while compared to variant D, which shows that group-group relationships play a more important role than inherent memberships of groups in group recommendation.

Effect of Pre-Training. To observe the impact of pre-training, we construct two extra variants of HL4EGR, namely variants E and F. Variant E removes the pre-training, thus collapsing into the scaled-down version of only containing H^{UG} and H^V and taking randomly initialized user embeddings as the input of fine-tuning. Variant F substitutes SASRec [11] for the HGCN on H^{UV}.

The performance of variants E and F is shown in the top half of Table 3(b). Variant E without pre-training shows the worst performance, indicating that the pre-training is indispensable.

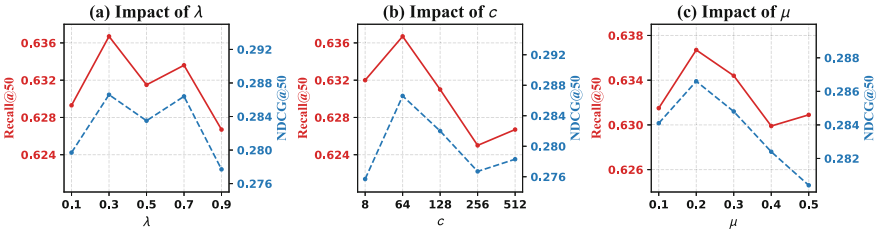

Fig. 4. Sensitivity analysis of hyperparameters λ, c and μ on Weeplaces dataset.

Variant F was originally anticipated to exhibit high performance because SASRec adopts a self-attentive mechanism that learns both long-term and short-term dependencies and produces high-quality user and item embeddings. However, experimental results show that variant F does not surpass the original HL4EGR. This observation reveals that the self-attention mechanism, which is good at capturing temporal dependencies embedded in sequences, does little to help eliciting the group preferences. Presumably the reason for this would be that group preferences are time-insensitive.

Effect of Hyperedge Weights. As mentioned in Sect. 3.2, when constructing group-group similarity hypergraphs in HL4EGR, the weights on the hyperedges are assigned to the same value, aiming to weaken the effect of the intensity of individual member behavior and preferences. To observe how weight values affect performance, we modify H^V and H^P by setting weights to the number of items interacted with by both members of two groups and the number of common preferences of two groups, respectively, and construct three variants of HL4EGR. Variant G introduces weights on H^V and H^P. Variant H introduces weights only on H^V while variant I does so only on H^P.

The experimental results are shown in the bottom half of Table 3(b). It can be seen that variants G, H and I have lower performance than HL4EGR. In particular, variant G has a significant performance degradation, which shows that simultaneously emphasizing the intensities of member behavior and preferences has a significant negative effect on group recommendation.

4.4 Hyperparameter Sensitivity Analysis

Impact of Contrastive Loss Weight λ. Figure 4(a) shows the results on Weeplaces dataset. This shows that appropriate contrastive learning loss can effectively normalize the group representations.

Impact of Number of Clustering Centers c. Figure 4(b) shows the results. HL4EGR achieves best performance on Weeplaces when $c = 64$. From Fig. 4(b), we believe that when c is very small, the model is unable to distinguish a user's different preferences, leading to false similarity when modeling group-group similarity. When c is too large, the model tends to identify the same preference of a user as multiple preferences, which fails to weaken the intensity of user's individual preference.

Impact of Distance Threshold μ. Figure 4(c) shows the results. HL4EGR achieves best performance when $\mu = 0.2$. We think that when μ is very small, items that indicate user preferences are filtered out; when μ becomes large, more items including noisy items are retained. Both cause performance reduction.

5 Conclusion

Ephemeral group recommendation is a challenging recommendation task, not only because group-item interactions are not enough to learn group preferences directly, but also because there are essential differences between group recommendation and personalized recommendation. This paper proposes a model HL4EGR that models the user-item interactions, user-group affiliations, and group-group similarities into multiple hypergraphs, reflecting the essence of the group recommendation. Meanwhile, HL4EGR also designs a contrastive learning

strategy on the hypergraphs, which enables HL4EGR to learn more comprehensive group preferences. The results of experiments on public datasets show that HL4EGR substantially improves the accuracy of ephemeral group recommendation results.

Acknowledgment. This work was supported by the National Natural Science Foundation of China under Grant No. 62072450.

References

1. Amer-Yahia, S., Roy, S.B., Chawlat, A., Das, G., Yu, C.: Group recommendation: semantics and efficiency. Proc. VLDB Endowment **2**(1), 754–765 (2009)
2. Baltrunas, L., Makcinskas, T., Ricci, F.: Group recommendations with rank aggregation and collaborative filtering. In: Proceedings of the Fourth ACM Conference on Recommender Systems, pp. 119–126 (2010)
3. Boratto, L., Carta, S.: State-of-the-art in group recommendation and new approaches for automatic identification of groups. In: Soro, A., Vargiu, E., Armano, G., Paddeu, G. (eds.) Information retrieval and mining in distributed environments, vol. 324, pp. 1–20. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-16089-9_1
4. Cao, D., He, X., Miao, L., An, Y., Yang, C., Hong, R.: Attentive group recommendation. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, pp. 645–654 (2018)
5. Cao, D., He, X., Miao, L., Xiao, G., Chen, H., Xu, J.: Social-enhanced attentive group recommendation. IEEE Trans. Knowl. Data Eng. **33**(3), 1195–1209 (2019)
6. Chen, T., Yin, H., Long, J., Nguyen, Q.V.H., Wang, Y., Wang, M.: Thinking inside the box: learning hypercube representations for group recommendation. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1664–1673 (2022)
7. Deng, Z., Li, C., Liu, S., Ali, W., Shao, J.: Knowledge-aware group representation learning for group recommendation. In: 2021 IEEE 37th International Conference on Data Engineering (ICDE), pp. 1571–1582. IEEE (2021)
8. Guo, L., Yin, H., Chen, T., Zhang, X., Zheng, K.: Hierarchical hyperedge embedding-based representation learning for group recommendation. ACM Trans. Inf. Syst. (TOIS) **40**(1), 1–27 (2021)
9. He, Z., Chow, C.Y., Zhang, J.D.: Game: learning graphical and attentive multi-view embeddings for occasional group recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 649–658 (2020)
10. Jia, R., Zhou, X., Dong, L., Pan, S.: Hypergraph convolutional network for group recommendation. In: 2021 IEEE International Conference on Data Mining (ICDM), pp. 260–269. IEEE (2021)
11. Kang, W.C., McAuley, J.: Self-attentive sequential recommendation. In: 2018 IEEE International Conference on Data Mining (ICDM), pp. 197–206. IEEE (2018)
12. Li, K., Wang, C.D., Lai, J.H., Yuan, H.: Self-supervised group graph collaborative filtering for group recommendation. In: Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, pp. 69–77 (2023)
13. Sajjadi Ghaemmaghami, S., Salehi-Abari, A.: Deepgroup: group recommendation with implicit feedback. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 3408–3412 (2021)

14. Sankar, A., Wu, Y., Wu, Y., Zhang, W., Yang, H., Sundaram, H.: GroupIM: a mutual information maximization framework for neural group recommendation. In: Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, pp. 1279–1288 (2020)
15. Vinh Tran, L., Nguyen Pham, T.A., Tay, Y., Liu, Y., Cong, G., Li, X.: Interact and decide: medley of sub-attention networks for effective group recommendation. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 255–264 (2019)
16. Wu, X., Xiong, Y., Zhang, Y., Jiao, Y., Zhang, J., Zhu, Y., Yu, P.S.: ConsRec: learning consensus behind interactions for group recommendation. In: Proceedings of the ACM Web Conference 2023, pp. 240–250 (2023)
17. Yadati, N., Nimishakavi, M., Yadav, P., Nitin, V., Louis, A., Talukdar, P.: HyperGCN: a new method for training graph convolutional networks on hypergraphs. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
18. Yin, H., Wang, Q., Zheng, K., Li, Z., Yang, J., Zhou, X.: Social influence-based group representation learning for group recommendation. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), pp. 566–577. IEEE (2019)
19. Yuan, Q., Cong, G., Lin, C.Y.: COM: a generative model for group recommendation. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 163–172 (2014)
20. Zhang, J., Gao, M., Yu, J., Guo, L., Li, J., Yin, H.: Double-scale self-supervised hypergraph learning for group recommendation. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 2557–2567 (2021)