Hierarchical Hyperedge Embedding-Based Representation Learning for Group Recommendation

Hierarchical Hyperedge Embedding-based Representation Learning for Group Recommendation

LEI GUO, Shandong Normal University, China
HONGZHI YIN*, The University of Queensland, Australia
TONG CHEN, The University of Queensland, Australia
XIANGLIANG ZHANG, King Abdullah University of Science and Technology, Saudi Arabia
KAI ZHENG, University of Electronic Science and Technology of China, China

Group Recommendation (GR) aims to recommend items to a group of users. In this work, we study GR in a particular scenario, namely Occasional Group Recommendation (OGR), where groups are formed ad-hoc and users may just constitute a group for the first time, that is, the historical group-item interaction records are highly limited. Most state-of-the-art works have addressed the challenge by aggregating group members' personal preferences to learn the group representation. However, the representation learning for a group is most complex beyond the aggregation or fusion of group member representation, as the personal preferences and group preferences may be in different spaces and even orthogonal. In addition, the learned user representation is not accurate due to the sparsity of users' interaction data. Moreover, the group similarity in terms of common group members has been overlooked, which however has the great potential to improve the group representation learning. In this work, we focus on addressing the aforementioned challenges in group representation learning task, and devise a hierarchical hyperedge embedding-based group recommender, namely HyperGroup. Specifically, we propose to leverage the user-user interactions to alleviate the sparsity issue of user-item interactions, and design a Graph Neural Network (GNN)-based representation learning network to enhance the learning of individuals' preferences from their friends' preferences, which provides a solid foundation for learning groups' preferences. To exploit the group similarity (i.e., overlapping relationships among groups) to learn a more accurate group representation from highly limited group-item interactions, we connect all groups as a network of overlapping sets (a.k.a. hypergraph), and treat the task of group preference learning as embedding hyperedges (i.e., user sets/groups) in a hypergraph, where an inductive hyperedge embedding method is proposed. To further enhance the group-level preference modeling, we develop a joint training strategy to learn both user-item and group-item interactions in the same process. We conduct extensive experiments on two real-world datasets and the experimental results demonstrate the superiority of our proposed HyperGroup in comparison to the state-of-the-art baselines.

CCS Concepts: • **Information systems → Recommender systems**.

Additional Key Words and Phrases: Group Recommendation, Hyperedge Embedding, Representation Learning

*Corresponding Author.

Authors' addresses: Lei Guo, leiguo.cs@gmail.com, Shandong Normal University, Jinan, Shandong, China, 250358; Hongzhi Yin, The University of Queensland, Brisbane, Australia, h.yin1@uq.edu.au; Tong Chen, The University of Queensland, Brisbane, Australia, tong.chen@uq.edu.au; Xiangliang Zhang, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia, xiangliang.zhang@kaust.edu.sa; Kai Zheng, University of Electronic Science and Technology of China, Chengdu, China, zhengkai@uestc.edu.cn.

1046-8188/xxx/0-ART0 $15.00
https://doi.org/10.1145/0000000.000000

ACM Reference Format:
Lei Guo, Hongzhi Yin, Tong Chen, Xiangliang Zhang, and Kai Zheng. xxx. Hierarchical Hyperedge Embedding-based Representation Learning for Group Recommendation. *ACM Transactions on Information Systems* 0, 0, Article 0 (xxx), 27 pages. https://doi.org/10.1145/0000000.000000

1 INTRODUCTION

With the recent advances in social networking services like Meetup and Facebook Event [37, 52], it is increasingly convenient for people with similar backgrounds (e.g., occupations, hobbies, locations, etc.) to form social groups and participate in activities in groups [19, 67], such as group tours, class reunion, and family dinners. It is becoming essential to develop group recommender systems [9, 33, 61, 62] to provide groups with appropriate recommendations (e.g., recommend a restaurant or a concert).

Generally, groups can be divided into persistent groups and occasional groups [8, 25, 65] based on whether they have stable group members. Concretely, Persistent Groups (PGs) [31, 47, 55] (a.k.a. static groups) often have fixed group members and abundant group-item interactions. For this kind of groups, we can directly apply the recommendation methods designed for individual users [11, 24, 28, 58] by treating each group as a pseudo user, since there are sufficient persistent group-item interactions. Occasional Groups (OGs) [4, 25, 39, 68] (a.k.a. cold-start group) refer to groups that are casually formed by ad-hoc users. As such kinds of groups are commonly established for temporary events (such as ride-sharing or attending an academic conference), the historical group-item interaction records are highly limited and even unavailable. Thus, the representation learning for an Occasional Group (OG) is more challenging than that for a Persistent Group (PG). Moreover, as different group members have different social influences and contribute differently to the group decision, OGR is much more complicated than making recommendations to individual users, and static and predefined aggregation methods [1, 4, 7, 48] are incompetent for the high complexity of group decision-making. In this study, we focus on OGR, which is more challenging and also more general in real-world applications compared with Persistent Group Recommendation (PGR).

To address the above challenges, some advanced data-driven aggregation methods [8, 25, 65] have been proposed to learn the group representation. For example, Cao et al. [8] incorporated the attention mechanism to dynamically aggregate different group members' preference information. To investigate the impact of group members' social influence on the group decision, Yin et al. [65] proposed a social influence-based group recommender to improve the preference aggregation process. But they ignore that the group's final decision is usually reached through consensus among group members, and the social interactions are unexplored. To address this limitation, Guo et al. [25] treated the group representation learning process as a multiple voting step, and developed a stacked social self-attention network [53] to aggregate group members' preferences. To alleviate the sparsity issue of group-item interaction data, the users' individual activity data has been leveraged to complement the group recommendation task in their method.

However, as the personal preferences and group preferences may be in different spaces and even orthogonal, the representation learning for a group is most complex beyond the aggregation or fusion of group member representation. Moreover, existing works tend to isolate groups when developing preference aggregation strategies, and ignore the fact that driven by multifaceted interests, a user may belong to multiple groups meanwhile. That is, the similarity between groups in terms of common group members is overlooked, which is substantially helpful to enhance the group preference modeling. Take a toy case for example, suppose we have groups A = {Amy, Bob, Carl} and B = {Amy, Bob, Eric}. When making recommendations for group A, if we know Amy and Bob are also in B, we can leverage the preference information of group B to improve the inference of group A's preference. Intuitively, the more common members A and B share, the more similar

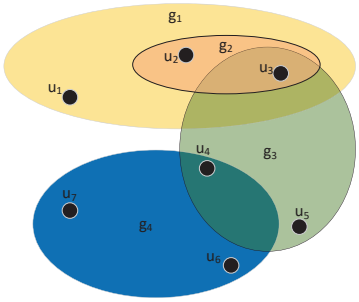

Fig. 1. Example of a hypergraph, where g_i denotes the i-th group/hyperedge that connects all the users within it. For example, users u_7, u_6 and u_4 are all connected by group/hyperedge g_4. The edge (a.k.a overlapping relationship) between two hyperedges/groups is established if they share at least one common group member, such as hyperedges/groups g_3 and g_4 share the same user u_4, then there is a connection between g_3 and g_4.

their group preferences will be, which provides another way to alleviate the data sparsity of group-item interaction data. Unfortunately, existing group recommendation methods fail to capture this important group-level similarity. In addition, these existing works have integrated the individual activity data to alleviate the sparsity issue of group-item interaction data, but they ignore a widely recognized fact that most of the user-item interaction data (i.e., individual activity data) are also extremely sparse.

To this end, instead of simply aggregating personal preferences as the group preference, we propose a hierarchical Hyperedge embedding-based Group recommender (HyperGroup) to learn group representation, which consists of Individual Preference Modeling (IPM), Hyperedge Representation Learning (HRL) and joint training components. More specifically, to address the sparsity issue of the user-item interactions, in our IPM component, we exploit the user-user interactions with the assumption that a user's preference can be indicated by his/her friends due to the social principle of homophily. Technically, we develop a GNN-based node embedding approach to learn group members' personal preferences that provide solid foundations for learning the preferences of OGs. To exploit the group similarity (i.e., overlapping relationships among groups) to enhance the process of learning groups' preferences, in our HRL component, we connect all groups as a network of overlapping sets (i.e., hypergraph, an example is shown in Fig. 1), and treat group representation learning as embedding hyperedges in a hypergraph, where a hyperedge is an edge that connects all users who belong to the same group. Then, an innovative inductive hyperedge embedding component is proposed to learn the representation of each hyperedge (i.e., group) by aggregating the representations of its incident hyperedges (i.e., groups that share at least one common member). Finally, a joint training strategy is developed to optimize the user-item and group-item recommendation tasks simultaneously, which provides an efficient way to accelerate and enhance the group-level preference modeling and learning.

Note that, the method that views a group as a hyperedge is different from the methods that directly treat groups as pseudo users, since they only depend on the highly limited group-item interaction records, while our method additionally considers both user-user and user-item interactions. Our method is also different from existing OGR studies that tend to model each group individually. HyperGroup exploits the group similarity in terms of common group members, thus making full use of the group-level collaborative preference signals to enhance the group preference learning.

The main contributions of this work are listed as follows:

- In this work, we 1) exploit the group similarity to improve the group representation learning and 2) introduce hyperedges to model groups.
- We propose HyperGroup, a novel GNN-based model for OGR tasks. Specifically, HyperGroup addresses the fundamental data sparsity problem in user-item interactions by leveraging friends' preferences from the social network, and further exploits group-level similarity with an innovative hyperedge embedding scheme for learning expressive group representations.
- We conduct extensive experiments on two real-world datasets and the experimental results demonstrate the superiority of our proposed method.

2 RELATED WORK

The group recommendation techniques that have been widely studied in the last years can be divided into two categories [41, 43]: recommendations to persistent groups and occasional groups. In a persistent group, it is assumed to have sufficient group feedback, while in an occasional group, the group feedback is highly limited and even not available. The group preference in the latter case must be learned on the basis of those of its members.

2.1 Persistent Group Recommendation

Due to persistent groups often have stable members and rich historical interactions [43, 65], previous studies [13, 31, 50] on this kind of group were mainly focused on treating groups as pseudo-users, and then adopt conventional personalized recommendation techniques [12, 14] for making group recommendations. For example, to estimate the rating that a group of members might give to an item, Chen et al. [13] proposed a genetic algorithm-based recommendation method by predicting the possible interactions among group members. But this method relies on known group preferences and is only implementable for persistent groups. Seko et al. [50] leveraged the entities that characterize groups (e.g., the tendency of content selection and the relationships among group members) to achieve high group recommendation accuracy. However, their method is also only applicable to predefined groups, such as couples and families, and requires a large amount of group behavior history. To relieve the vulnerability of data [5], Hu et al. [31] devised a deep architecture model via using high-level features that are learned from lower-level features to represent group preference. Similar to work [50], this method only focuses on pre-defined groups instead of occasional groups, and thus cannot be applied to our setting.

2.2 Occasional Group Recommendation

As existing works developed for occasional group recommendation mainly focus on investigating the strategies of aggregating individual preferences to conduct group recommendations, in the following we review these studies from two aspects: late aggregation methods and early aggregation methods.

2.2.1 Late Aggregation methods. The task of late aggregation methods [1, 16, 48, 63] is to aggregate the prediction scores (or recommendation list) of individual members as the score (or result) of the target group. That is, they first generate recommendation results for each group member, and then produce group recommendation via aggregating these individual results based on the static predefined aggregation strategies. In this category, three kinds of aggregation strategies are commonly utilized [1, 4, 43, 48], i.e., average satisfaction, least misery and maximum pleasure. For example, average satisfaction exploits the average score of all group members as the prediction score of the group by assuming each group member is equally important to the group decision making process. Least misery treats the minimum score of individual members as the prediction

of the group, where the least satisfied group member plays a key role in forming the group's final decision. However, these predefined aggregation strategies are heuristic and unstable (as shown in [17], none of them achieves the best performance on all the datasets), only sub-optimal recommendation results are reached.

2.2.2 Early Aggregation methods. Early aggregation methods are also known as preference aggregation-based methods [8, 39, 55, 66, 68] that aim at aggregating preferences of individual group members as the profile of a group. Compared with late aggregation methods, these methods first aggregate the preference (or representation) of group members, and then make group recommendations (or produce prediction scores) accordingly. For example, [21, 39, 68] studied group recommendation by developing probabilistic generative models, and aggregated both the group members' individual preferences and their impacts in the group to make recommendations. Although their works treat users differently by assuming they may have different contributions to the group, both models develop the same probability distribution for each user, which is infeasible in real-world cases. To address this problem, Cao et al. [8] proposed an attention-based neural network to aggregate individual representation (or profile) dynamically. Vinh Tran et al. [55] further captured the fine-grained interactions between group members via the sub-attention networks. But their work did not consider the sparsity issue of the group-item interactions, and has limited capability in dealing with occasional group recommendations. Yin et al. [65] incorporated the social influence to OGR, and proposed a social influence-enhanced group recommender. But the interactions among group members are ignored. To address the above challenges, Guo et al. [25] developed a group self-attention neural network to model the social influence and interactions among group members simultaneously. He et al. [30] modeled the interactions among groups, users, and items with an interaction graph, and then learned their multi-view embeddings. That is, learning embeddings of groups, users, and items from their interacted counterparts to improve recommendations for occasional groups. But this work [30] only considers user-group interactions, group-item interactions, and user-item interactions, while the group-group correlation (i.e., group similarity) and the social homophily in the user-user interaction network are not studied. Sankar et al. [49] proposed two data-driven strategies to investigate the preference covariance across individuals in the same group and the contextual relevance of users' individual preferences to each group. However, the captured preference covariance is different from our group-level similarity, which refers to the overlapping relationship among groups and can enhance the representation learning of groups by considering the groups that have common members with them.

Differences: Our hierarchical solution has significant differences from these existing studies. First, compared to social-interest based group recommendation methods (e.g., SIGR [65] and GroupSA [25]), we propose a different way to alleviate the sparsity issue of user-item interactions, that is, a GNN-based user embedding network is developed to enhance the learning of users' preferences from their neighbors. Second, compared to recent neural network-based group recommendation methods (e.g., AGREE [8], SIGR [65], GroupSA [25], GAME [30] and GroupIM [49]), we further investigate the overlooked group similarity to alleviate the group data sparsity by treating the task of learning groups' preferences as embedding hyperedges in a hypergraph, where a weighted preference aggregation strategy is developed to consider the overlapping relationships among groups as well as the specific users who connect the groups.

3 METHODOLOGIES

This section first gives an overview of our hierarchical group recommendation network, and then describes each of its components in detail.

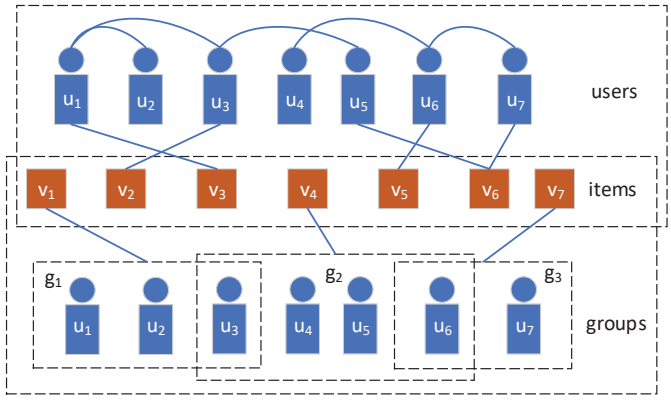

Fig. 2. Illustration of the input data of our OGR task, which includes user-item interactions, group-item interactions, user-user interactions and group-level overlapping relationships.

3.1 Preliminaries

In this work, we use bold lowercase letters (e.g., $\boldsymbol{x}$) to represent vectors (all vectors are in column forms if not specified), and employ bold capital letters (e.g., $\boldsymbol{X}$) to represent matrices. None-bold lowercase and capital letters (e.g., x and X) are utilized to represent scalars. Squiggle letters (e.g., $\mathcal{X}$) are used to denote sets.

Fig.2 illustrates the input data of our OGR task. Let $\mathcal{U} = \{u_1, u_2, ..., u_j, ...u_m\}$, $\mathcal{V} = \{v_1, v_2, ..., v_h, .., v_n\}$ and $\mathcal{G} = \{g_1, g_2, ..., g_t, ..., g_k\}$ be the sets of users, items and groups, and m, n and k denote the numbers of users, items and groups in the three sets respectively. The t-th group $g_t \in \mathcal{G}$ consists of a set of users $\mathcal{G}(t) = \{u_1, u_2, ..., u_j, ..., u_l\}$, where $u_j \in \mathcal{U}$, l is the size of g_t, and $\mathcal{G}(t)$ is the user set of g_t. Each user/group interacts with different items, which indicate their preferences. Besides, users can build social connections with others, and groups have overlapping relationships with others by sharing common group members. Totally, there are four kinds of interactions among $\mathcal{U}$, $\mathcal{V}$ and $\mathcal{G}$, namely user-user interactions, user-item interactions, group-item interactions and group-level overlapping relationships, which are respectively denoted by $\boldsymbol{R}^S = [r^S_{j,j'}]^{m\times m}$, $\boldsymbol{R}^U = [r^U_{j,h}]^{m\times n}$, $\boldsymbol{R}^G = [r^G_{t,h}]^{k\times n}$ and $\boldsymbol{R}^H = [r^H_{t,t}]^{k\times k}$ respectively. We use $r = 1$ to indicate observed interactions, and $r = 0$ for unobserved ones.

Take g_1 as an illustrated example (as shown in Fig.2). Suppose g_1 is a target group composed of three group members u_1, u_2 and u_3. Our goal is to generate a ranked list of items that g_1 is likely to interact with. As g_1 is formed occasionally, there are limited group-item interactions, and directly learning the representation of g_1 (i.e., the preference of g_1) is not feasible. Hence, we focus on designing GNN-based models to alleviate this sparsity issue by leveraging the user-item interactions (e.g., $(u_1, v_3), (u_2, v_1)$), social connections (e.g., $(u_1, u_2), (u_1, u_3)$) and similar groups that have common members with g_1 (i.e., g_2). The formal definition of the group recommendation task is as follows:

Input: Users $\mathcal{U}$, items $\mathcal{V}$, groups $\mathcal{G}$ as well as user-user, user-item, group-item interactions, and group-level overlapping relationships respectively denoted by $\boldsymbol{R}^S$, $\boldsymbol{R}^U$, $\boldsymbol{R}^G$ and $\boldsymbol{R}^H$.

Output: A function that maps an item to a real-valued score which indicates its probability of being consumed by the target group: $f_t : \mathcal{V} \rightarrow \mathbb{R}$.

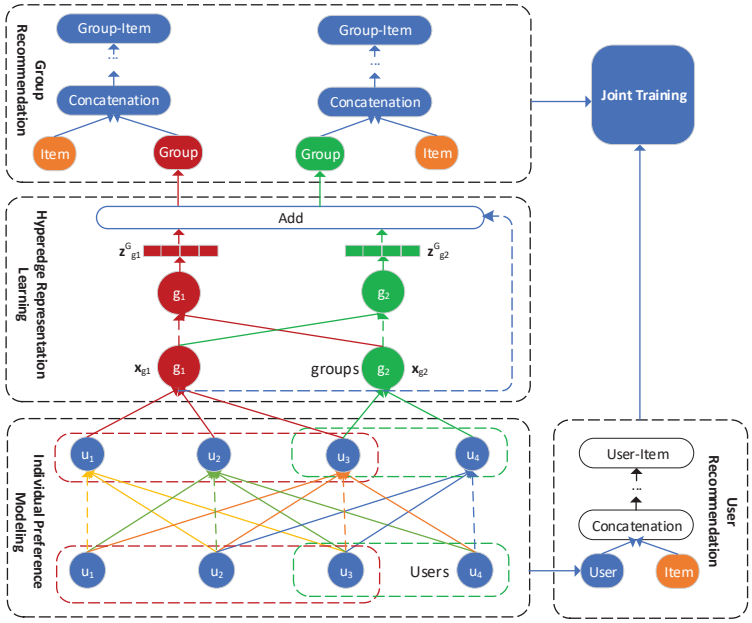

Fig. 3. Overview of the architecture of HyperGroup.

3.2 Overview of HyperGroup

In this work, we propose the recommender HyperGroup for OGR, which is designed to address the data sparsity issue in group representation learning problem with the power of a hierarchical graph neural network.

Motivation. Due to the sparsity nature of occasional groups, it is not straightforward to model the group preferences. The highly limited group-item and user-item interactions of OGs make the recommendation task most challenging. An effective way to alleviate this sparsity issue is to enhance the preferences of both users and groups by leveraging the preferences of their connected neighbors, which falls into the paradigm of GNN-based methods [32, 42, 60], that is, enhance users' preferences by exploiting their social connections, and enhance groups' preferences by exploiting groups that have common group members with them. Intuitively, two groups with group members tend to have similar overall preferences, and the more common members two groups share, the higher their similarity will be. In this work, we propose to model groups as hyperedges in a hypergraph rather than treating a group as a node to build a group graph, since not only the group-level overlapping relationship, but also the specific users shared by two groups have the great potential to enhance the group preference learning. In other words, we care about not only the number of the common group members between two groups, but also who the common members are. Based on this intuition, an innovative preference aggregation strategy for hypergraph is further developed (as shown in Eq.(3), see Section 3.4).

Fig. 3 shows the architecture of our HyperGroup, which consists of three components: IPM, HRL and joint model optimization. IPM is a GNN-based graph embedding module that is designed to exploit users' social connections, inspired by the social principle of homophily [34], to alleviate the sparsity issue of user-item interactions, where each group member's personal preference is enhanced by their friends' preferences and then is fed into the second component HRL to provide foundations for the group representation learning (see Section 3.3). HRL is developed to exploit

the group similarity based on common group members to learn a more accurate group representation by modeling groups as hyperedges, and the task of learning group representations is then transformed into the task of learning hyperedge embeddings (see Section 3.4). Finally, two joint training strategies are proposed to simultaneously optimize IPM and HRL in the same training process (see Section 3.5).

3.3 Individual Preference Modeling

Due to the rare interaction records of OGs, directly learning their preferences by treating them as pseudo users is infeasible. Therefore, motivated by recent studies [8, 25, 65], we devise a paradigm that learns each group member's individual preference and then aggregates them as our initial group representation. However, due to the sparsity of individual activity data, the learned individual preference may be not accurate. To address this challenge, motivated by the social principle of homophily [34], we develop a social-enhanced individual preference modeling method in the IPM component.

As social animals, users turn to their friends for recommendations [10, 18, 59], and also share many common preferences with friends. Thus, users' preferences can be summarized from both themselves and their direct neighbors (i.e., friends) in a social network [23, 40]. Specifically, to learn users' preferences from their social neighbors (denoted as $\boldsymbol{R}^S$), we treat users as nodes in a large social graph and develop a GNN-based node embedding module, where the preferences of their neighbors and their own are simultaneously considered. It is worth mentioning that, compared with the spectral graph convolutional network [36], we build our model upon an information aggregation-based network [26, 46], which is an inductive representation learning approach that bypasses the need for the entire graph's node adjacency matrix to operate, thus being space-efficient when handling large-scale datasets for recommendation.

Algorithm 1 Individual embedding generation algorithm

Require: User-user interactions $\boldsymbol{R}^S$; input features $\{\boldsymbol{x}_u, \forall u \in \mathcal{U}\}$; depth K; weight matrices $\boldsymbol{W}^i, \forall i \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions AGGEGATE_i, $\forall i \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : u \rightarrow 2^{\mathcal{U}}$

Ensure: Vector representations $\boldsymbol{z}_u$ for all $u \in \mathcal{U}$

$\boldsymbol{h}_u^0 \leftarrow \boldsymbol{x}_u, \forall u \in \mathcal{U}$;
for $i = 1...K$ **do**
  for $u \in \mathcal{U}$ **do**
    $\boldsymbol{h}^i_{\mathcal{N}(u)} \leftarrow \text{AGGREGATE}_i(\{\boldsymbol{h}^{i-1}_{u'}, \forall u' \in \mathcal{N}(u)\})$;
    $\boldsymbol{h}^i_u \leftarrow \sigma(\boldsymbol{W}^i \cdot concat(\boldsymbol{h}^{i-1}_u, \boldsymbol{h}^i_{\mathcal{N}(u)}))$;
  end for
  $\boldsymbol{h}^i_u \leftarrow \boldsymbol{h}^i_u / ||\boldsymbol{h}^i_u||_2, \forall u \in \mathcal{U}$;
end for
$\boldsymbol{z}_u \leftarrow \boldsymbol{h}^K_u, \forall u \in \mathcal{U}$;

Algorithm 1 shows the details of our GNN-based individual preference modeling component, which takes a social graph $\boldsymbol{R}^S$ with its node/user features $\boldsymbol{x}_u, u \in \mathcal{U}$ as the input. We adopt the node embedding method node2vec [22] to obtain the initialized node/user features, as it can well balance the embedding quality and computation cost. In the outer loop of this algorithm, each user $u \in \mathcal{U}$ first aggregates the representations of the nodes in its immediate neighborhood, $\{\boldsymbol{h}^{i-1}_{u'}, \forall u' \in \mathcal{N}(u)\}$, into a single vector $\boldsymbol{h}^{i-1}_{\mathcal{N}(u)}$, where $\mathcal{N}(u)$ is the sampled neighbors of u with a fixed size S, i denotes the i-th iteration or the i-th layer of GNN and $\boldsymbol{h}^i$ denotes a node's representation

at this iteration or layer. For $i = 0$, we let $\boldsymbol{h}^0_{\mathcal{N}(u)} = \boldsymbol{x}_u$. Note that a node's representation $\boldsymbol{h}^i_u$ at i-th iteration depends on both its own representation $\boldsymbol{h}^{i-1}_u$ and the aggregated neighborhood vector representation $\boldsymbol{h}^i_{\mathcal{N}(u)}$ generated at the i-1-th layer. We adopt the concatenation operation to combine them, followed by a fully connected layer with nonlinear activation function $\sigma(\cdot)$ and weight matrices $\boldsymbol{W}^i, \forall i \in \{1, ..., K\}$, which are used to propagate information between different layers. The final output at the K-th layer is denoted as $\boldsymbol{z}_u = \boldsymbol{h}^K_u, \forall u \in \mathcal{U}$ for convenience, which encodes u's preferences.

The aggregation function that aims at aggregating neighbor representations (denoted by AGGREGATE$(\cdot)$ in Algorithm 1) can be done by a variety of aggregator architectures [26] (e.g., mean aggregator, max-pooling aggregator and LSTM aggregator). In this work, we simply take the mean aggregator as our aggregation function, where the element-wise mean operation is applied to aggregate information across the neighbor set:

$$\text{AGGREGATE}_i(u) = \text{MEAN}(\boldsymbol{h}^i_{u'}, \forall u' \in \{\mathcal{N}(u)\}) \tag{1}$$

Then, these learned individual embeddings within an occasional group are further aggregated [38] to produce the group representation in the individual preference space. The embedding of group g is denoted as:

$$\boldsymbol{x}^G_g = \sum_{\forall u \in \mathcal{G}(g)} \alpha_u \boldsymbol{emb}_u, \tag{2}$$

where $\boldsymbol{emb}_u = \boldsymbol{z}_u + \boldsymbol{z}'_u$, $\boldsymbol{z}_u$ is the learned individual representation of group member u, $\boldsymbol{z}'_u$ is u's embedding in latent feature space, and α_u denotes the importance of user embedding. As this work is not focused on designing preference aggregation strategies, the simple average operation is utilized (i.e., $\alpha_u = 1/|\mathcal{G}(g)|$). The output of IPM ($\boldsymbol{x}^G_g$) is then passed to the HRL component as the initial representations of the corresponding group, which will be further optimized and learned in the higher-layer of our framework.

Compared to existing transductive individual preference learning methods that are based on matrix factorization [57, 64], IPM leverages node features to learn an embedding function that generalizes to unseen nodes, where the topological structure of each node's neighborhood and the distribution of node features in the neighborhood are simultaneously learned. Moreover, rather than training a distinct embedding vector for each node, we train K aggregation functions to aggregate feature information from the local neighborhood. Each aggregator function aggregates information from a different number of hops away from a given node [26, 54]. Note that, the number of hops is equal to the number of aggregators (both denoted as K).

3.4 Hyperedge Embedding-based Group Representation Learning

Simply aggregating group members' individual preferences as the preference of a group would miss the intrinsic group-level preferences which may be different from all individuals' preferences within the group. Moreover, as a user may belong to multiple groups at the same time, the groups that have common members should have similar group-level preferences. To quantify the similarity between two groups, the common users between two groups are further exploited to model the group-level similarity. What matters is not only the number of the common users, but also who the common users are, as different users have different influences on the group decision making. For example, suppose groups g_1, g_2 share a common user u_1, and groups g_2, g_3 share a common user u_2. If we know u_1 is more influential than u_2, we can infer that, the similarity between g_2 and g_1 is higher than that between g_2 and g_3. On this basis, let us further assume that g_2 has sufficient historical data and g_1 is a cold-start group, then g_2's preferences would provide important signals

Algorithm 2 Hyperedge embedding generation process

Require: Hypergraph $\mathcal{G}^G(\mathcal{U}, \mathcal{G})$; learned features $\{\boldsymbol{x}_g^G, \forall g \in \mathcal{G}\}$ from IPM; depth K; weight matrices $\boldsymbol{W}^i, \forall i \in \{1, 2, ..., K\}$; nonlinear activation function σ; aggregator functions AGGEGATE_i^G, $\forall i \in \{1, 2, ..., K\}$; neighborhood function $\mathcal{N} : g \rightarrow 2^{\mathcal{G}}$

Ensure: Hyperedge embeddings $\boldsymbol{z}_g^G$ for all $g \in \mathcal{G}$

$\boldsymbol{m}_g^0 \leftarrow \boldsymbol{x}_g^G, \forall g \in \mathcal{G}$;
for $i = 1...K$ **do**
  for $g \in \mathcal{G}$ **do**
    $\boldsymbol{m}_{\mathcal{N}(g)}^i \leftarrow \text{AGGREGATE}_i^G(\{\boldsymbol{m}_{g'}^{i-1} + \boldsymbol{l}_{g,g'}^i, \forall g' \in \mathcal{N}(g)\})$;
    $\boldsymbol{m}_g^i \leftarrow \sigma(\boldsymbol{W}^i \cdot concat(\boldsymbol{m}_g^{i-1}, \boldsymbol{m}_{\mathcal{N}(g)}^i))$;
  end for
  $\boldsymbol{m}_g^i \leftarrow \boldsymbol{m}_g^i / ||\boldsymbol{m}_g^i||_2, \forall g \in \mathcal{G}$;
end for
$\boldsymbol{z}_g^G \leftarrow \boldsymbol{m}_g^K, \forall g \in \mathcal{G}$;

for g_1's preferences, thus further alleviate the data sparsity issue and produce more effective group representations for OGR.

To capture the group-level preferences and further exploit the group similarity to enhance group preference learning, we innovatively introduce a hypergraph [2, 6, 56] to model groups rather than treating a group as a node to build a group graph. That method would fail to capture who the common users are. In a hypergraph, each group is treated as a hyperedge that connects all users in that group, and two hyperedges are incident if they share at least one common member.

Then, the task of learning group representations is transformed into embedding hyperedges in a hypergraph. To integrate the group similarity based on common members in the learning process of hyperedge embedding, we devise a GNN-based hyperedge embedding model, called HRL. As OGs tend to be formed by chance [25, 65], HRL also adopts an inductive graph embedding method [3, 26] as its building block to generate embeddings for hyperedges, where a weighted feature aggregation scheme is proposed to account for the similarity between two groups.

Formally, given k groups $\mathcal{G} = \{g_1, g_2, ..., g_t, ..., g_k\}$ defined over the user set $\mathcal{U}$, where $g_t = \mathcal{G}(t) = \{u_1, u_2, ..., u_j, ...u_l\}$ consists of a set of users, we first construct a hypergraph $\mathcal{G}^G = (\mathcal{U}, \mathcal{G})$, where $\mathcal{G}$ is the collection of hyperedges/groups over the nodes/users $\mathcal{U}$. Let $\boldsymbol{H} \in \{0, 1\}^{|\mathcal{G}| \times |\mathcal{U}|}$ represent the incidence matrix of $\mathcal{G}^G$ with $\boldsymbol{H}(g, u) = 1$ if $u \in \mathcal{G}(g)$ else 0. The degree $d(u)$ of a vertex u is defined as the number of hyperedges associated with u, i.e., $d(u) = \sum_{g \in \mathcal{G}} \boldsymbol{H}(g, u)$. To generalize to an unobserved group, a hyperedge embedding generator will be learned and its basic idea is to aggregate feature information from its incident hyperedges as its embedding (i.e., its preference encoding).

Algorithm 2 describes the hyperedge embedding generation process, where the constructed hypergraph, $\mathcal{G}^G(\mathcal{U}, \mathcal{G})$, and the learned features $\boldsymbol{x}_g^G, \forall g \in \mathcal{G}$ from IPM are provided as input. In the higher-layer of our network, $\boldsymbol{x}_g^G, \forall g \in \mathcal{G}$ serves as the initial representation of group g, which will be further optimized. At each iteration or layer i, a hyperedge $g \in \mathcal{G}$ aggregates features from its immediate local neighbors (i.e., its incident hyperedges). Specifically, a hyperedge $g \in \mathcal{G}$ aggregates the embeddings of its incident hyperedges, $\{\boldsymbol{m}_{g'}^{i-1}, \forall g' \in \mathcal{N}(g)\}$, into a single vector $\boldsymbol{m}_{\mathcal{N}(g)}^i$ (the size of $\mathcal{N}(g)$ is also set as S), where $\boldsymbol{m}_{g'}^{i-1}$ denotes the embedding of g' generated at the previous the iteration/layer $i - 1$. After that, the aggregated neighborhood vector is concatenated with

the hyperedege's previous representation, $\boldsymbol{m}_g^{i-1}$, and followed by a fully connected layer with non-linear activation function $\sigma(\cdot)$ and weight matrices $\boldsymbol{W}^i, \forall i \in \{1, ..., K\}$, which are used to search depths. The representation obtained at the last layer/iteration $\boldsymbol{m}_g^K$ denotes the group g's preferences, which is defined as $\boldsymbol{z}_g^G = \boldsymbol{m}_g^K, \forall g \in \mathcal{G}$. We define our hyperedge aggregator AGGREGATE^G as follows:

$$\text{AGGREGATE}_i^G(g) = \sum_{\forall g' \in \mathcal{N}(g)} \alpha_{g,g'}(\boldsymbol{m}_{g'}^i + \boldsymbol{l}_{g,g'}^i) \tag{3}$$

where $\alpha_{g,g'}$ is the aggregation weight determined by the similarity between group g and g'.

In this work, we set $\alpha_{g,g'}$ to number of the common members between group g and g'. By doing this, we are able to give more attention to the groups sharing more common members with g. In Eq.(3), $\boldsymbol{l}_{g,g'}$ denotes the representation of the set of common members between group g and g', which is used to distinguish the specific members who are shared by these two groups. For simplicity, MEAN aggregator is employed to compute $\boldsymbol{l}_{g,g'}$:

$$\boldsymbol{l}_{g,g'}^i = \text{MEAN}(\boldsymbol{emb}_u, \forall u \in \{\mathcal{G}(g) \cap \mathcal{G}(g')\}). \tag{4}$$

To this end, our proposed representation aggregator not only considers the number of the shared common users, but also emphasises the importance of individuals who are the shared group members.

The resulted group representation $\boldsymbol{z}_g^G$ is then added with $\boldsymbol{x}_g^G$ by the residual operation. The final embedding of group g (denoted as $\boldsymbol{emb}_g^G$) can be represented as:

$$\boldsymbol{emb}_g^G = w\boldsymbol{z}_g^G + (1-w)\boldsymbol{x}_g^G, \tag{5}$$

where w is a hyper-parameter controlling the contributions of the two parts.

Note that, our solution is different from existing graph neural networks [3, 26] in two aspects. First, our group representation learning network is hierarchical, which first learns the group members' personal preferences in the lower-layer (i.e., IPM) and then infers groups' representations in the higher-layer (i.e., HRL). Second, to exploit and integrate the group similarity based on common members in HRL, the preference aggregation strategy (a.k.a hyperedge aggregator) that considers both the group-level preference and the personal preferences of common group members is devised.

3.5 Model Optimization

Given the embeddings of the target group and item (i.e., $\boldsymbol{emb}_g^G$ and $\boldsymbol{emb}_h^V$), we feed the concatenation of them into a Multi-Layer Perception (MLP) for preference prediction (as shown in Fig.1):

$$\begin{aligned}
&\boldsymbol{c}_1^G = [\boldsymbol{emb}_g^G \oplus \boldsymbol{emb}_h^V] \\
&\boldsymbol{c}_2^T = \sigma(\boldsymbol{W}_2 \cdot \boldsymbol{c}_1^G + \boldsymbol{b}_2) \\
&\cdots \\
&\hat{r}_{g,h}^G = \boldsymbol{w}^T \cdot \boldsymbol{c}_{k-1}^G
\end{aligned} \tag{6}$$

where $\boldsymbol{emb}_h^V$ is the item embedding in the latent space that is learned via optimizing the following loss function (Eq.(7)); $\boldsymbol{W}$ and $\boldsymbol{b}$ are the weight and bias of a feed-forward network; $\hat{r}_{g,h}^G$ is the predicted preference score of group g to item v_h.

Due to the implicit nature of the group-item interaction data, motivated by [25], a pairwise loss function [44] is employed:

$$L_G = \arg\min_{\Theta} \sum_{(g,v_h,v_{h'})\in\mathcal{D}_G} -\ln\sigma(\hat{r}^G_{g,h} - \hat{r}^G_{g,h'}) + \lambda||\Theta||^2 \tag{7}$$

where Θ represents the set of the model parameters. To learn from this implicit feedback, we reconstruct the group-item data by assuming that groups prefer observed item v_h over all other unobserved item v'_h. Then, the training data $\mathcal{D}_G : \mathcal{G} \times \mathcal{V} \times \mathcal{V}$ can be denoted as:

$$\mathcal{D}_G = \{(g, v_h, v_{h'}) | v_h \in \mathcal{V}^+_g \wedge v_{h'} \in \mathcal{V} \setminus \mathcal{V}^+_g\} \tag{8}$$

where $\mathcal{V}^+_g$ and $\mathcal{V} \setminus \mathcal{V}^+_g$ are the observed and unobserved item set w.r.t group g. In this work, we use N_x to denote the number of sampled negative items per positive item. The meaning of $(g, v_h, v_{h'}) \in \mathcal{D}_G$ is that group g prefers item v_h over $v_{h'}$.

As in OGs, the available group-item interactions are extremely sparse, the learned group representations (via optimizing Eq. (7)) are not sufficiently accurate or reliable. To further accelerate and enhance the group preference learning, we propose to leverage the user-item interaction data to optimize the group-item and user-item recommendation tasks simultaneously. As shown in Fig. 3, we propose to use another MLP to model the user-item interaction data. More specifically, given the embeddings of the target user and item, we first feed them into a MLP to calculate the personal preference score of a user to an item:

$$\begin{aligned}
\boldsymbol{c}^U_1 &= [\boldsymbol{emb}_u \oplus \boldsymbol{emb}^V_h] \\
\boldsymbol{c}^U_2 &= \sigma(\boldsymbol{W}_2 \cdot \boldsymbol{c}^U_1 + \boldsymbol{b}_2) \\
&\cdots \\
\hat{r}^U_{u,h} &= \boldsymbol{w}^T \cdot \boldsymbol{c}^U_{k-1}
\end{aligned} \tag{9}$$

where $\boldsymbol{emb}_u$ is the shared user representation/embedding that connects all user-user, user-item and group-item spaces and data. $\hat{r}^U_{u,h}$ is the predicted preference score of user u to item v_h. $\boldsymbol{emb}^V_h$ is another shared item embedding that bridges the group-item space and user-item space. As user-item interaction data is also implicit, the same pairwise loss function is utilized:

$$L_U = \arg\min_{\Theta} \sum_{(u,v_h,v_{h'})\in\mathcal{D}_U} -\ln\sigma(\hat{r}^U_{u,h} - \hat{r}^U_{u,h'}) + \lambda||\Theta||^2 \tag{10}$$

where $\mathcal{D}_U$ denotes the set of reconstructed user-item samples; $(u, v_h, v_{h'})$ represents user u prefers observed item v_h over unobserved item $v_{h'}$. Similar to Eq. (7), for every positive item, N_x negative items are randomly sampled. Technically, to integrate L_G with L_U, we develop two model optimization approaches Two-stage Training and Joint Training.

Two-stage Training. In this strategy, we first optimize L_U by the user-item interaction data to learn the representations of users and items in the user-item space, and then take item latent features as the latent vector of items in the group-item recommendation task (as shown in Algorithm 3). In the second stage, the parameters will be fine-tuned by optimizing L_G with the group-item interactions. In both of these two training stages, the Stochastic Gradient Descent (SGD) algorithm is adopted, and at each gradient step, a positive user-item sample (u, v_h) (or group-item example (g, v_h)) and N_x negative corresponding samples $(u, v_{h'})$ (or $(g, v_{h'})$) are randomly selected for training.

Joint Training. In this strategy, we jointly train L_G and L_U on all the group-item and user-item interactions (as shown in Algorithm 4), and the loss function is actually changed to the following

Algorithm 3 Two-state training method of HyperGroup

Require: $\mathcal{R}^S, \mathcal{R}^U, \mathcal{R}^G, \mathcal{R}^H$, number of positive samples of users M_u, number of positive samples of groups M_g, number of negative samples N_x;
Ensure: Parameter set $\Theta = \{\boldsymbol{emb}_g^G, \boldsymbol{emb}_h^V, \boldsymbol{emb}_u, \boldsymbol{W}, \boldsymbol{b}\}$;
1: **while** $iter \leq M_u$ **do**
2: Randomly draw (u, v_h) from $\mathcal{R}^U$;
3: Randomly sample N_x negative examples for u;
4: Update the model parameters w.r.t. Eq. (10);
5: $iter = iter + 1$;
6: **end while**
7: **while** $iter \leq M_g$ **do**
8: Randomly draw (g, v_h) from $\mathcal{R}^G$;
9: Randomly sample N_x negative examples for g;
10: Update the model parameters w.r.t. Eq. (7);
11: $iter = iter + 1$;
12: **end while**

Algorithm 4 Joint training method of HyperGroup

Require: $\mathcal{R}^S, \mathcal{R}^U, \mathcal{R}^G, \mathcal{R}^H$, number of positive samples of users M_u, number of positive samples of groups M_g, number of negative samples N_x;
Ensure: Parameter set $\Theta = \{\boldsymbol{emb}_g^G, \boldsymbol{emb}_h^V, \boldsymbol{emb}_u, \boldsymbol{W}, \boldsymbol{b}\}$;
1: **while** $iter \leq (M_u + M_g)$ **do**
2: Randomly draw (u, v_h) from $\mathcal{R}^U$ and sample N_x negative examples for u, and update the model parameters w.r.t. Eq. (10);
3: Randomly draw (g, v_h) from $\mathcal{R}^G$ and sample N_x negative examples for g, and update the model parameters w.r.t. Eq. (7);
4: $iter = iter + 1$;
5: **end while**

equation:

$$L(\boldsymbol{\Theta}) = L_G(\boldsymbol{\Theta}) + L_U(\boldsymbol{\Theta}). \tag{11}$$

All the parameters (denoted by $\boldsymbol{\Theta}$) are learned by a standard Stochastic Gradient Descent (SGD), and at each gradient step, we first randomly draw a positive user-item sample (u, v_h) and a positive group-item example (g, v_h)) from the user-item set and group-item set respectively, and then draw N_x negative corresponding samples $(u, v_{h'})$ and $(g, v_{h'})$ to update the gradients.

4 EXPERIMENTAL SETUP

In this section, we first introduce the research questions that we aim to answer in experiments, and then describe the datasets, evaluation methods and baselines utilized in this work.

4.1 Research Questions

We conduct extensive experiments on two real-world datasets to answer the following research questions.

RQ1 How does our proposed HyperGroup approach perform compared with state-of-the-art group recommendation methods?

Table 1. Statistics of the datasets.

Statistics	Yelp	Douban-Event
# Users	34,504	29,181
# Groups	24,103	17,826
# Items/Events	22,611	46,097
Avg. group size	4.45	4.84
Avg. # interactions per group	1.12	1.47
Avg. # friends per user	20.77	40.86
Avg. # interactions per user	13.98	25.22

RQ2 How do the three components of HyperGroup, i.e., Individual Preference Modeling (IPM), Hyperedge Representation Learning (HRL), and the joint training method contribute to the performance of HyperGroup? How do our proposed model optimization approaches perform on heterogeneous interaction data?
RQ3 How do the hyper-parameters affect the performance of HyperGroup?
RQ4 How is the training efficiency and scalability of HyperGroup when processing large-scale data?

4.2 Datasets

To evaluate the performance of our HyperGroup method, we conduct experiments on two large-scale real-world datasets Yelp[1] and Douban-Event[2] that are exclusively published for OGR by Yin et al. [65]. Yelp is a famous online social network that connects people with local businesses (e.g., restaurants and home services), where users can publish their reviews about these businesses and create social connections. The published dataset only focuses on the restaurants located in Los Angeles and every record in it contains a user, a timestamp and a business, which indicates the user visited the restaurant at that time. Douban-Event is one of the largest online event-based social networks in China that helps people publish and participate in social events. In this dataset, the user's event attendance list and friend list, as well as the event's time and venue were collected.

As the raw data of these two datasets does not contain any explicit group information, Yin et al. [65] extracted implicit groups by the following strategy: if a set of users who are connected in the social network attend the same event or visit the same restaurant at the same time, they are defined as the members of a group, and the group activities are the common activities of these users. The resulted Yelp data has 34,504 users, 24,103 groups, and 22,611 items for training and testing. For the Douban-Event data, to reduce the data size, we follow the data used in [25], which is generated by randomly keeping 29,181 users, 17,826 groups and 46,097 items. The statistics of these two datasets are shown in Table 1, from which we have the following observation, that is, compared with user-item interactions, the group-item interactions are much sparser. For example, in Yelp a user has 20.77 interactions on average, while a group has only 1.12 interactions. The second observation is that the user-item interaction data is also quite sparse. The densities of the user-item interaction matrices for Yelp and Douban-Event are 0.051% and 0.057%, respectively.

[1] www.yelp.com
[2] www.douban.com/location/world/

Note that, as the other two datasets CAMRa2011[3] and Movielens-Group [68] have either persistent groups or randomly generated groups, and none of them contain the social network information, they are not suitable to evaluate our solution. We do not conduct experiments on these two datasets.

4.3 Evaluation Protocols

We randomly split each dataset into training, validation and test sets with the ratio of 80%, 10% and 10% respectively. To fully evaluate our proposed method, we do not follow the evaluation protocol proposed in [8, 25], which only randomly selects 100 items that have never been interacted by the tested group as the candidate set to be ranked. Instead, we evaluate all the comparison methods by testing their ability to rank all items for each tested group, and report their performance in recommending Top-N items. The evaluation metrics Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) [27] are adopted in our experiments, where HR measures how many candidate items are ranked within the Top-N list, while NDCG accounts for the position of the hit by assigning higher score to hit at top positions.

More specifically, for each group-item interaction (g, v) in the test set, we first compute the ranking score for item v and all candidate items. And then, we pick N items with the highest ranking scores as the Top-N recommendation list. If item v appears in this list, we have a hit. Otherwise, we have a miss. The formal definition of HR [27] is written as follows:

$$\text{HR@}N = \frac{\#hit@N}{|\mathcal{D}_{test}|} \tag{12}$$

where $\#hit@N$ denotes the number of hits in the test set, and $|\mathcal{D}_{test}|$ is the total number of the test cases.

The metric NDCG [27] is defined as:

$$\text{NDCG@}N = Z_N \sum_{i=1}^{N} \frac{2^{r_i} - 1}{log_2(i+1)} \tag{13}$$

where Z_N is the normalizer[4] to ensure that the perfect ranking has a value of 1; r_i is the graded relevance of item at position i. We use the simple binary relevance in this work, that is, if the item at position i is the ground-truth item, $r_i = 1$; otherwise $r_i = 0$.

4.4 Baseline Methods

We compare HyperGroup with the following baseline methods.

- **Pop [15].** This is a popularity-based recommendation method, which recommends the most popular items in the training set.
- **NCF [29].** This method is developed for individual users. We utilize this method for OGR by treating groups as virtual users.
- **BPR-MF [44].** This is a traditional collaborative filtering-based method exploiting the pairwise loss as the optimization objection for recommending items to individual users. Same as NCF, we used it for OGR via assuming groups are virtual users.
- **PIT [39].** This is a probabilistic model devised for OGR, which extends the author topic model [45] proposed for document-authorship analysis by treating a group of users as the authors of a document and the interacted items as the words of the document. In this method,

[3]http://2011.camrachallenge.com/2011

[4]We set $Z_N = \log(2)$, as we use the binary relevance of item.

a personal impact parameter is introduced to model the representativeness of each member to a group.

- **COM [68].** This is another topic model-based approach proposed for OGR, but different from PIT that only considers group members' own topic preferences to select items, it considers both members' topic-dependent influences and group behaviors.
- **AGREE [8].** This is the first work that employs a neural attention network to learn the dynamic aggregation strategy for OGR.
- **SIGR [65].** This work develops a deep social influence learning framework to exploit both global and local social network structures to learn the social influence or weight of each group member in the group decision making. This is the first work to focus on the data sparsity issues of OGR.
- **GroupSA [25].** This is the state-of-the-art group recommendation method proposed for OGR, where the self-attention mechanism is utilized to learn the group preference aggregation strategies.
- **GroupIM [49].** This is another state-of-the-art group recommendation method developed for OGR, which leverages two data-driven strategies to investigate the preference covariance across individuals in the same group and the contextual relevance of users' individual preferences to each group. However, the captured preference covariance is different from our group-level similarity, which refers to the overlapping relationship among groups and can enhance groups' preferences via exploiting groups that have common group members with them.

4.5 Implementation Details

We implement HyperGroup based on Pytorch accelerated by NVIDIA RTX 2080 Ti GPU. In experiments, we first initialize the parameters using the Glorot initialization method [20], and then use the Adam optimizer [35] to optimize our loss function, where the mini-batch size is set to 256, and the initial learning rate is set to 0.0001. For hyper-parameters, the number of negative samples (N_x) per positive sample is searched within $\{1, 2, 3, 4, 5\}$; the dimensions of the network features, the embeddings of user, group and item are all set to 128; the number of latent layers is set as $K = 1$ for IPM and $K = 2$ for HRL; the number of sampled neighbors (denoted as S) for IPM and HRL are both searched within $\{1, 2, 3, 4, 5\}$; the hyper-parameter w that determines the importance of the residual connection is searched within [0.1-0.9] with a step size of 0.1. The details of tuning the hyper-parameters are shown in Section 6.2. To avoid over-fitting, the dropout regularization method [51] with drop ratio 0.1 is utilized for both datasets. If not specified, all the reported experimental results of our methods are achieved with a Two-stage Training strategy.

For the settings of baseline methods, we tune the following hyper-parameters that are reported as important factors in their publications to obtain optimal performance, and let the others as the default setting (both datasets are applied): 1) For NCF, we set the learning rate = [0.0005, 0.0001, 0.00005], negative samples = 3, and dropout ratio =0.1. 2) For BPR-MF, we set the factor number = 30, and sampled triples = $\sqrt{MaxUserID} \times 100$. 3) For PIT and COM, we tune the number of topics and achieve the best result when topic number = 250. 4) For AGREE, we set the learning rate = $[0.005, 0.001, 0.0005]$, and negative samples = 1. For fair comparisons, in all the ranking-based methods developed for OGR (i.e., AGREE, SIGR, and GroupSA), the number of negative samples per positive sample is set as 1 (as the setting in HyperGroup). 5) For SIGR, we set the importance controller $\eta = 0.5$ and $1/\rho_S^2 = 0.05$. 6) For GroupSA, we set the self-attention layer as 2, the number of items (or users) utilized in the item aggregation (or social aggregation) as 4. 7) For GroupIM, we set the layer size = 64, and negative users sampled per group = 5.

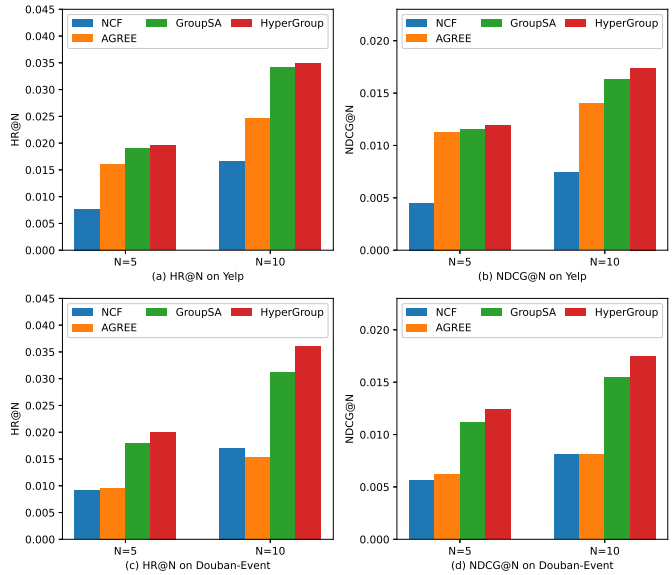

Fig. 4. Top-N recommendation performance on individual users.

Note that, all the baselines are trained end-to-end and the neural network-based methods (i.e., NCF, AGREE, SIGR, GroupSA, GroupIM and HyperGroup) are optimized with no pre-training.

Table 2. Top-N Recommendation performance on Yelp and Douban-Event via evaluating on all items.

Overall Performance Comparison								
Methods	Yelp				Douban-Event			
	N=5		N=10		N=5		N=10	
	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG
Pop	0.0117	0.0076	0.0201	0.0103	0.0031	0.0017	0.0046	0.0022
NCF	0.0110	0.0074	0.0193	0.0100	0.0041	0.0024	0.0061	0.0030
BPR-MF	0.0026	0.0056	0.0022	0.0078	0.0009	0.0017	0.0007	0.0023
PIT	0.0128	0.0076	0.0258	0.0117	0.0079	0.0043	0.0190	0.0075
COM	0.0481	0.0313	0.0812	0.0420	0.0103	0.0053	0.0214	0.0089
AGREE	0.0569	0.0389	0.0896	0.0495	0.0122	0.0073	0.0255	0.0116
SIGR	0.1085	0.0738	0.1499	0.0871	0.0200	0.0114	0.0345	0.0162
GroupSA	0.1211	0.0843	0.1680	0.0992	0.0212	0.0137	0.0382	0.0191
GroupIM	0.1312	0.1033	0.1493	0.1090	0.0511	0.0358	0.0669	0.0406
HyperGroup	**0.4827**	**0.3973**	**0.5598**	**0.4223**	**0.0608**	**0.0406**	**0.0914**	**0.0505**

5 EXPERIMENTAL RESULTS (RQ1)

The comparison results with the baseline methods are shown in Table 2, from which we can observe that: 1) HyperGroup significantly outperforms all the baselines on the two datasets (all the improvements are statistically significant with $p < 0.01$), which demonstrates the advantage of our hyperedge embedding-based solution. 2) NCF performs better than BPR-MF, but it can only

get similar or even worse results than Pop. This is because in OGR, groups are formed occasionally, and the observed group-item interactions are extremely sparse. In our Yelp and Douban-Event datasets, the average numbers of interactions per group are 1.12 and 1.47, respectively. This data sparsity issue makes it infeasible to treat a group as a virtual user and learn a group's interests only from her historical group-item interaction data (i.e., NCF and BPR-MF). 3) The performance of group recommendation methods (PIT, COM, AGREE, SIGR, GroupSA and HyperGroup) developed for OGR achieves superior performance over the recommendation methods proposed for individual users (i.e., NCF and Pop). This again demonstrates the complexity of the occasional group recommendation process, and simply view groups as virtual users cannot get satisfactory results. 4) Neural network-based group recommendation methods (i.e., AGREE, SIGR, GroupSA, GroupIM and HyperGroup) outperform topic model-based methods (PIT and COM), indicating the capability of neural networks in capturing group members' behaviour patterns, which can lead to a more accurate recommendation result. COM outperforms PIT because groups in the two datasets are loosely organized and there may not exist a representative member to make item selections for a group. 5) The methods that consider user's social influence (HyperGroup, GroupSA and SIGR) achieve better results than other baselines, demonstrating the benefits brought by exploiting the social influence.

To investigate the recommendation performance of our method in ranking items for individual users, we further compare HyperGroup with baselines that can make recommendations for individuals (i.e., NCF, AGREE and GroupSA) on the user-item recommendation task. The experimental results are reported in Fig. 4, from which we can find that HyperGroup also achieves the best performance, demonstrating the effectiveness and advantage of GNN in learning the representations of individual users, as well as the joint training method to mutually enhance group recommendation and individual recommendation.

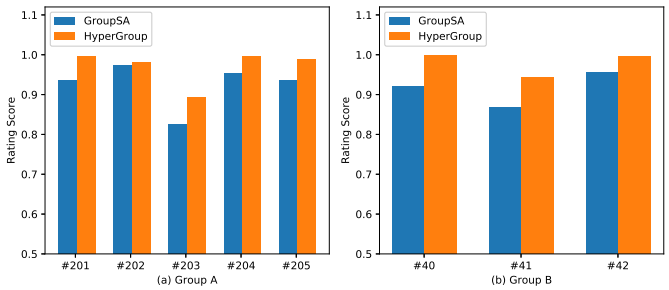

Fig. 5. Case studies: preference scores predicted by HyperGroup and GroupSA on Yelp.

Case Studies. Besides the above macro-level analysis, we also conduct case studies from a micro-level view via visualizing the rating scores for two randomly-chosen groups (A and B) from Yelp, which have interacted with items (#201, #202, #203, #204 and #205) and items (#40, #41, #42), respectively. To demonstrate the superiority of HyperGroup in predicting the preferences of groups to items, we compare it with the state-of-the-art group recommendation method GroupSA. The experimental results are shown in Fig. 5, from which we find that HyperGroup can make more accurate predictions than GroupSA, since for these ground-truth items, the predicted scores by HyperGroup are more close to the target value 1 than GroupSA. This result demonstrates the capability of HyperGroup in predicting groups' preferences and thus leads to a better recommendation result for occasional groups.

Table 3. Importance of components of HyperGroup.

Importance of Components of HyperGroup								
Methods	Yelp				Douban-Event			
	N=5		N=10		N=5		N=10	
	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG
AGREE	0.0569	0.0389	0.0896	0.0495	0.0122	0.0073	0.0255	0.0116
HGroup-SH	0.2159	0.1651	0.2806	0.1860	0.0309	0.0225	0.0478	0.0280
HGroup-S	0.2679	0.2089	0.3354	0.2305	0.0384	0.0286	0.0545	0.0337
HGroup-H	0.3543	0.2764	0.4339	0.3021	0.0514	0.0340	0.0789	0.0429
HyperGroup	**0.4827**	**0.3973**	**0.5598**	**0.4223**	**0.0608**	**0.0406**	**0.0914**	**0.0505**

Table 4. Importance of User-item Interaction Data.

Importance of User-Item Interaction Data								
Methods	Yelp				Douban-Event			
	N=5		N=10		N=5		N=10	
	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG
NCF	0.0110	0.0074	0.0193	0.0100	0.0041	0.0024	0.0061	0.0030
HGroup-U	0.0123	0.0081	0.0201	0.0106	0.0031	0.0019	0.0054	0.0026
HyperGroup	**0.4827**	**0.3973**	**0.5598**	**0.4223**	**0.0608**	**0.0406**	**0.0914**	**0.0505**

6 MODEL ANALYSIS

In this section, we first conduct ablation studies to investigate the impact of model components and hyper-parameters to HyperGroup. Then, the training efficiency of HyperGroup is further investigated.

6.1 Importance of Components (RQ2)

To investigate the importance of IPM and HRL, we first compare HyperGroup with its three variants:

- **HGroup-SH.** This is a simplified version of HyperGroup that replaces both the IPM and HRL components with more basic components. Specifically, it uses only a basic matrix factorization model without consideration of the social network information to learn individual preferences of group members, and then employs a vanilla attention-based preference aggregation strategy to learn groups' preferences. This is to study the effectiveness of these two components.
- **HGroup-S.** This is another variant of HyperGroup that removes the IPM component, that is, excluding the GNN-based individual preference learning mechanism from HyperGroup, and utilizes a basic matrix factorization model to learn users' personal preferences. This is to validate the importance of leveraging friends' preferences to enhance the individual's preference learning.
- **HGroup-H.** This variant removes the HRL component from HyperGroup, and utilizes an average aggregation-based strategy to learn the groups' representations by averaging the personal preferences of group members. This is to evaluate the effect of our hyperedge embedding-based group preference learning mechanism.

6.1.1 Importance of IPM. To evaluate our individual preference learning component, we compare HyperGroup with HGroup-SH and HGroup-S. The experimental results are reported in Table 3, from which we have the following observations: (1) HyperGroup significantly outperforms HGroup-SH and HGroup-S on both datasets, indicating the importance of the IPM and HRL components, and only considering one of them alone cannot get better results than combing them together. (2) HyperGroup performs better than HGroup-S, demonstrating the benefit of exploiting the social interests to alleviate the data sparsity of user-item interactions and the effectiveness of our GNN-based learning paradigm. This result also demonstrates that IPM is able to provide solid foundations for further learning group representations via hyperedge embedding techniques. (3) From Table 3, we also notice that all our variants of HyperGroup perform better than AGREE which only utilizes the attention mechanism to make group recommendation. This result demonstrates the effectiveness of our HyperGroup solution and the importance of exploiting social connections to alleviate the sparsity issue of users' interaction data as well as the benefit of investigating group similarity to enhance groups' preference learning.

6.1.2 Importance of HRL. To validate our HRL component, we further conduct another ablation study by comparing HyperGroup with HGroup-H. From the experimental results shown in Table 3, we can observe that: (1) HyperGroup outperforms HGroup-H demonstrating the power of our hierarchical neural network and the effectiveness of learning group representations by treating groups as hyperedges in a hypergraph. That is, modeling the group similarity in terms of common group members is helpful for learning a better group representation and thus leads to a better group recommendation. (2) We also notice that there is a bigger gap between HyperGroup and HGroup-S than that between HyperGroup and HGroup-H, which indicates that without effective individual preferences, we can only get sub-optimal group representations, and accurate personal preferences of group members can provide foundations for learning effective group representations.

6.1.3 Importance of the User-item Interactions. To validate the importance of integrating user-item interaction data to enhance the group preference learning process, we conduct experiments to compare HyperGroup with NCF and HGroup-U:

- **HGroup-U.** This is a variant of HyperGroup that does not integrate the user-item interaction data and only utilizes the group-item interaction data to train the model.

The experimental results are reported in Table 4. From the results we have the following observations: (1) HyperGroup consistently and significantly outperforms HGroup-U in both datasets, which validates the usefulness of user-item interaction data in enhancing the training of our group recommendation model, that is, leveraging the learned user and item representations from user-item interactions to provide solid foundations for group preference learning. (2) HGroup-U outperforms NCF, which demonstrates the capability of our method in learning group preferences, that is, the advantage of the GNN-based hyperedge embedding method. This result also demonstrates the importance of exploiting the group similarity in group representation learning. But we also notice that without the help of user-item interactions, the performance of HGroup-U has a big gap with HyperGroup, due to the extreme sparsity of group-item interaction data.

6.1.4 Comparison of Different Model Optimization Approaches. To evaluate the performance of our proposed model optimization strategies on heterogeneous data, we compare the following four model optimization strategies:

- **Group-ST.** This is the single training strategy that optimizes HyperGroup only on the group-item interactions, which is also known as HGroup-U.
- **Group-G.** This is the strategy that optimizes HyperGroup via only considering the user-item loss.

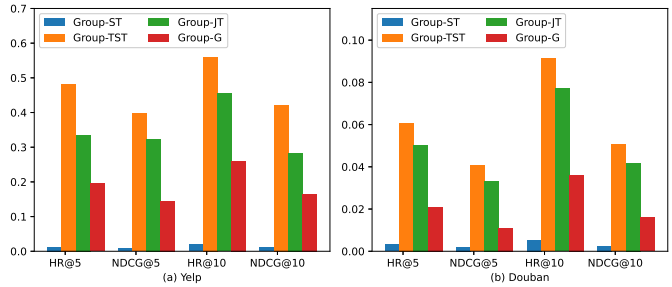

Fig. 6. Comparison of different model optimization approaches.

Table 5. Impact of parameter S.

S	HR@5	NDCG@5	HR@10	NDCG@10
1	0.4339	0.3531	0.5160	0.3796
2	0.4459	0.3612	0.5203	0.3853
3	0.4452	0.3670	0.5314	0.3950
4	0.4827	0.3973	0.5598	0.4223
5	0.4734	0.3873	0.5517	0.4126

Table 6. Impact of parameter N_x.

N_x	HR@5	NDCG@5	HR@10	NDCG@10
1	0.4827	0.3973	0.5598	0.4223
2	0.5582	0.4797	0.6254	0.5017
3	0.5862	0.4963	0.6600	0.5201
4	0.5845	0.5030	0.6440	0.5223
5	0.5842	0.4866	0.6594	0.5110

- **Group-TST.** This strategy integrates the user-item interactions and group-item interactions via a two-stage training method (as shown in Section 3.5).
- **Group-JT.** This is another optimization method that integrates the user-item interaction data via jointly training the group-item and user-item recommendation task simultaneously (as shown in Section 3.5).

From the experimental results reported in Fig. 6 we can observe that: (1) All joint training methods (i.e., Group-TST and Group-JT) significantly outperform the single training methods Group-ST and Group-G, demonstrating the significance of leveraging the user-item interaction data for training HyperGroup, and the ability of our joint model optimization method in addressing the heterogeneous data (i.e., the mixture of user-item and group-item interaction data). (2) Group-TST performs better than Group-JT, demonstrating the two-stage training method is more suitable to optimize our hierarchical group recommendation model, which first learns the embeddings of individual users in the lower layer of HyperGroup and then based on that learns group embeddings in the higher layer of our model.

Table 7. Impact of parameter w.

w	HR@5	NDCG@5	HR@10	NDCG@10
0.1	0.4452	0.3670	0.5314	0.3950
0.3	0.4736	0.3870	0.5505	0.4119
0.5	0.4827	0.3973	0.5598	0.4223
0.7	0.4554	0.3754	0.5307	0.3998
0.9	0.4541	0.3682	0.5323	0.3936

Table 8. Impact of Different Group Sizes (denoted by l).

l	HR@5	NDCG@5	HR@10	NDCG@10
$l < 3$	0.4709	0.3779	0.5467	0.4023
$3 \leq l \leq 7$	0.5529	0.4848	0.6020	0.5007
$7 < l$	0.4580	0.3770	0.5526	0.4073

Table 9. Top-N Recommendation Performance on Yelp (τ denotes the number of items visited by groups).

τ	HR@5	NDCG@5	HR@10	NDCG@10
$\tau \leq 3$	0.4668	0.3633	0.5576	0.3927
$\tau > 3$	0.4936	0.4164	0.5684	0.4408

6.2 Impact of Hyper-parameters (RQ3)

Tables 5-7 present the experimental results on tuning the hyper-parameters of HyperGroup. Due to similar results are achieved on Douban-event, only the results on Yelp are reported.

6.2.1 Impact of S. The hyper-parameter S refers to the number of neighbors sampled at each layer. A higher value of S indicates that there are more neighbors of users or groups that are aggregated in the corresponding aggregation functions. The recommendation performance with respect to S is shown in Table 5 (same values of S for both components are utilized in this result), from which we find diminishing returns for sampling large neighbors, and when the number of sampled neighbors surpass a certain value, the recommendation performance will even deteriorate, since more unrelated users or groups are considered. Moreover, we also notice that large sampled neighbors significantly increase the running time. To strike a balance between running time and performance, we set $S = 4$ for both IPM and HRL components of HyperGroup on two datasets.

6.2.2 Impact of N_x. We investigate the performance of HyperGroup with respect to different values of N_x, which denotes the number of negative samples utilized for per positive sample. As the results shown in Table 6, there is a high variance induced by the number of sampled negative examples. From this result, we can find that generating more negative samples for per positive sample is helpful to obtain a more accurate recommendation model. The best performance of our method is achieved when $N_x = 4$. This result also indicates that very few negative samples can already lead to satisfactory results. In experiments, we set $N_x = 1$ on both Yelp and Douban-Event as in [25] to make our results comparable with them.

6.2.3 Impact of w. To explore the impact of the residual connection (i.e., the hyper-parameter w in Eq.(5)), we further conduct experiments by varying the values of w, which plays a role control the contributions of the two types of group representations. The experimental results are shown

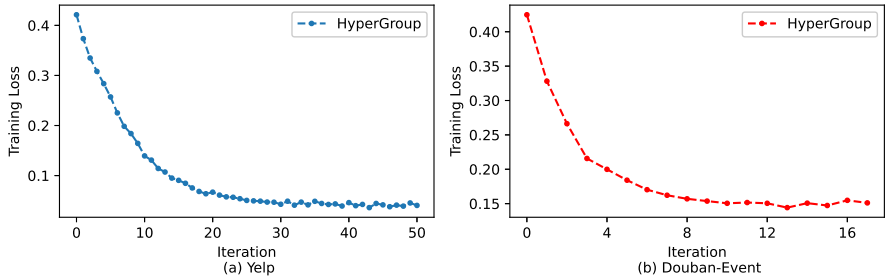

Fig. 7. Training loss of HyperGroup w.r.t the number of iterations on Yelp and Douban-Event.

in Table 7. The best performance is achieved at $w = 0.5$ on Yelp, and $w = 0.3$ on Douban-Event, from which we can observe that if we pay less attention to the individual preferences, we cannot get a better group representation, since the group preference learning takes the preferences of individual group members as foundations.

6.2.4 Impact of l. To study the performance of our method on groups with different sizes, we evaluate HyperGroup by splitting groups in the testing data into three bins based on their size, that is, small ($l < 3$), medium ($3 \leq l \leq 7$) and large ($l > 7$). The experimental results are reported in Table 8, which indicate that HyperGroup is more suitable to make recommendations for medium groups. The best result is achieved on the medium group bin (i.e., $3 \leq l \leq 7$). The main reason behind this result is that in a small group we do not have enough group-user interactions to explore the group-level similarities, and in a large group, the group members are more difficult to reach consensus due to the personal interests of individuals.

6.2.5 Impact of τ. To test our model's performance on different levels of item interaction sparsity (cold-start vs. popular items), we conduct experiments on items with different activity levels on Yelp, where group-item interactions in the test data are split into two bins based on item activity, that is, interactions with cold-start items ($\tau \leq 3$) and interactions with popular items ($\tau > 3$). The experimental results are reported in Table 9, from which we can observe that HyperGroup achieves expected performance on popular items. But we also find that HyperGroup achieves comparable results on cold-start items, which demonstrates the importance of group members' individual preferences and social interests in recommending items to occasional groups, as well as the effectiveness of our hierarchical hyperedge embedding-based solution.

6.2.6 Convergence. To demonstrate the rationality of our learning scheme, we report the value of training loss along with each iteration using the optimal parameter setting in Fig. 7. From this result, we can observe that with the increasing number of iterations the training loss of HyperGroup gradually decreases on both datasets. On Yelp HyperGroup converges fast in the first 20 iterations, and reaches its optimal results around the 30th iteration, while on Douban-Event it achieves its best performance around the 10th iteration. This result indicates the rationality of our training strategy.

We also explore the impact of different feature generation methods utilized in the IPM component (i.e., the features denoted by $\boldsymbol{x}_u$), but the results achieved by different initialization strategies for embedding features are very close after the HyperGroup model is fully trained. Hence, we do not report these results.

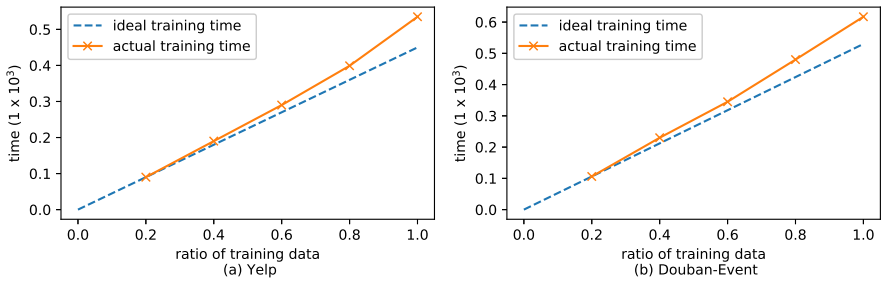

Fig. 8. Training time of HyperGroup with different data sizes.

6.3 Training Efficiency and Scalability (RQ4)

To investigate the practicality of our recommendation method in real-world applications, we validate the training efficiency and scalability of HyperGroup via measuring the time cost for the model training with different proportions of the training data (Yelp and Douban-Event). That is, we vary the ratios of the training data in $\{0.2, 0.4, 0.6, 0.8, 1.0\}$, and then report the corresponding training time in Fig. 8. The experimental results of HyperGroup are obtained with all the hyper-parameters are fixed. To make our results comparable, the expected ideal training time that is linearly associated with the number of training samples is also reported in Fig. 8. From the experimental results, we can observe that when the ratio of the training data gradually increases from 0.2 to 1.0, the time cost for training HyperGroup on Yelp grows from 0.09×10^3 seconds to 0.535×10^3 seconds and it grows from 0.106×10^3 seconds to 0.617×10^3 seconds on Douban-Event. The overall trend on these two datasets shows that the dependency of times cost for training HyperGroup on the data scale is approximately linear. This result provides us positive evidence to answer RQ4, that is, HyperGroup is scalable to large scale datasets.

7 CONCLUSIONS

In this work, we investigated the OGR problem, and proposed a hierarchical GNN-based group recommender HyperGroup to learn the group preference via the hyperedge embedding technique based on the learned individual preferences of group members. In this way, our method not only can model the individual-level preferences, but also the group-level communications. Specifically, to alleviate the sparsity issue of user-item interactions, we first learned group members' personal preferences by leveraging their social interests to provide solid foundations for group representation learning. Then, to enhance the group representations by leveraging the group similarity, we connected all groups as a hypergraph, and proposed a hyperedge embedding method to solve the OGR problem in the higher-layer of our network. Finally, to leverage the user-item interactions to further accelerate the training process of the group-item recommendation task, two joint optimization strategies were developed. To validate the effectiveness of our HyperGroup, we conducted extensive experiments on two real-world datasets that are proposed for OGR task. The experimental results demonstrated the superiority of our hierarchical hyperedge embedding-based solution in making recommendations for occasional groups.

Besides the Yelp and Douban-Event datasets, our method can also be applied to other real-world settings, such as the users who attend an academic conference or the friends that meet at social events. As in the above cases where groups are formed occasionally, there are no historical group activities. Recommending items (such as trips or restaurants) to these kinds of groups falls into the OGR scenario, where our HyperGroup method can be applied. That is, first learning the interests of

individual users by leveraging their social connections, and then inferring groups' representations via the overlapping relationship among them.

ACKNOWLEDGMENTS

This work was supported by National Natural Science Foundation of China (No. 61602282), ARC Discovery Project (No. DP190101985) and China Postdoctoral Science Foundation (No. 2016M602181).

REFERENCES

[1] Sihem Amer-Yahia, Senjuti Basu Roy, Ashish Chawlat, Gautam Das, and Cong Yu. 2009. Group recommendation: Semantics and efficiency. *International Conference on Very Large Data Bases* 2, 1 (2009), 754–765.

[2] Song Bai, Feihu Zhang, and Philip H. S. Torr. 2019. Hypergraph Convolution and Hypergraph Attention. arXiv:1901.08150 [cs.LG]

[3] Yunsheng Bai, Hao Ding, Yang Qiao, Agustin Marinovic, Ken Gu, Ting Chen, Yizhou Sun, and Wei Wang. 2019. Unsupervised Inductive Graph-Level Representation Learning via Graph-Graph Proximity. arXiv:1904.01098 [cs.LG]

[4] Linas Baltrunas, Tadas Makcinskas, and Francesco Ricci. 2010. Group recommendations with rank aggregation and collaborative filtering. In *ACM Conference on Recommender Systems*. 119–126.

[5] Y. Bengio, A. Courville, and P. Vincent. 2013. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 8 (2013), 1798–1828.

[6] Claude Berge. 1984. *Hypergraphs: combinatorics of finite sets*. Vol. 45. Elsevier.

[7] Shlomo Berkovsky and Jill Freyne. 2010. Group-based recipe recommendations: analysis of data aggregation strategies. In *ACM Conference on Recommender Systems*. 111–118.

[8] Da Cao, Xiangnan He, Lianhai Miao, Yahui An, Chao Yang, and Richang Hong. 2018. Attentive group recommendation. In *ACM SIGIR Conference on Research and Development in Information Retrieval*. 645–654.

[9] D. Cao, X. He, L. Miao, G. Xiao, H. Chen, and J. Xu. 2019. Social-Enhanced Attentive Group Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2019), 1–1.

[10] H. Chen, H. Yin, T. Chen, W. Wang, X. Li, and X. Hu. 2020. Social Boosted Recommendation with Folded Bipartite Network Embedding. *IEEE Transactions on Knowledge and Data Engineering* (2020), 1–1.

[11] Tong Chen, Hongzhi Yin, Hongxu Chen, Rui Yan, Quoc Viet Hung Nguyen, and Xue Li. 2019. Air: Attentional intention-aware recommender systems. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 304–315.

[12] Wanyu Chen, Pengjie Ren, Fei Cai, and Maarten de Rijke. 2019. Improving End-to-End Sequential Recommendations with Intent-aware Diversification. arXiv:1908.10171 [cs.IR]

[13] Yen-Liang Chen, Li-Chen Cheng, and Ching-Nan Chuang. 2008. A Group Recommendation System with Consideration of Interactions among Group Members. *Expert Syst. Appl.* 34, 3 (2008), 2082–2090.

[14] Zhiyong Cheng, Xiaojun Chang, Lei Zhu, Rose Catherine Kanjirathinkal, and Mohan S. Kankanhalli. 2019. MMALFM: Explainable Recommendation by Leveraging Reviews and Images. *ACM Trans. Inf. Syst.* 37, 2 (2019), 16:1–16:28.

[15] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *ACM Conference on Recommender Systems*. 39–46.

[16] Andrew Crossen, Jay Budzik, and Kristian J Hammond. 2002. Flytrap: intelligent group music recommendation. In *International conference on Intelligent user interfaces*. 184–185.

[17] Toon De Pessemier, Simon Dooms, and Luc Martens. 2014. Comparison of group recommendation algorithms. *Multimedia tools and applications* 72, 3 (2014), 2497–2541.

[18] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The World Wide Web Conference*. 417–426.

[19] Li Gao, Jia Wu, Zhi Qiao, Chuan Zhou, Hong Yang, and Yue Hu. 2016. Collaborative social group influence for event recommendation. In *ACM Conference on Information and Knowledge Management*. 1941–1944.

[20] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*. 249–256.

[21] Jagadeesh Gorla, Neal Lathia, Stephen Robertson, and Jun Wang. 2013. Probabilistic group recommendation via information matching. In *The World Wide Web Conference*. 495–504.

[22] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 855–864.

[23] Junpeng Guo, Yanlin Zhu, Aiai Li, Qipeng Wang, and Weiguo Han. 2016. A social influence approach for group user modeling in group recommendation systems. *IEEE Intelligent Systems* 31, 5 (2016), 40–48.

[24] Lei Guo, Hongzhi Yin, Qinyong Wang, Tong Chen, Alexander Zhou, and Nguyen Quoc Viet Hung. 2019. Streaming Session-Based Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 1569–1577.

[25] Lei Guo, Hongzhi Yin, Qinyong Wang, Bin Cui, Zi Huang, and Lizhen Cui. 2020. Group Recommendation with Latent Voting Mechanism. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. 121–132.

[26] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Conference on Neural Information Processing Systems*. 1024–1034.

[27] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *ACM Conference on Information and Knowledge Management*. 1661–1670.

[28] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. *arXiv preprint arXiv:2002.02126* (2020).

[29] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *The World Wide Web Conference*. 173–182.

[30] Zhixiang He, Chi-Yin Chow, and Jia-Dong Zhang. 2020. 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 (SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 649–658.

[31] Liang Hu, Jian Cao, Guandong Xu, Longbing Cao, Zhiping Gu, and Wei Cao. 2014. Deep modeling of group preferences for group-based recommendation. In *Conference on Artificial Intelligence*.

[32] Linmei Hu, Siyong Xu, Chen Li, Cheng Yang, Chuan Shi, Nan Duan, Xing Xie, and Ming Zhou. 2020. Graph Neural News Recommendation with Unsupervised Preference Disentanglement. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 4255–4264.

[33] Z. Huang, X. Xu, H. Zhu, and M. Zhou. 2020. An Efficient Group Recommendation Model With Multiattention-Based Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020), 1–14.

[34] Kazi Zainab Khanam, Gautam Srivastava, and Vijay Mago. 2020. The Homophily Principle in Social Network Analysis. arXiv:2008.10383 [cs.SI]

[35] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

[36] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[37] Chenghao Liu, Xin Wang, Tao Lu, Wenwu Zhu, Jianling Sun, and Steven Hoi. 2019. Discrete social recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 208–215.

[38] Jiangming Liu and Yue Zhang. 2017. Attention modeling for targeted sentiment. In *The European Chapter of the Association for Computational Linguistics*. 572–577.

[39] Xingjie Liu, Yuan Tian, Mao Ye, and Wang-Chien Lee. 2012. Exploring personal impact for group recommendation. In *ACM Conference on Information and Knowledge Management*. 674–683.

[40] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* 27, 1 (2001), 415–444.

[41] Mark O'Connor, Dan Cosley, Joseph A. Konstan, and John Riedl. 2001. *PolyLens: A Recommender System for Groups of Users*. Springer Netherlands, Dordrecht, 199–218.

[42] Ruihong Qiu, Zi Huang, Jingjing Li, and Hongzhi Yin. 2020. Exploiting Cross-Session Information for Session-Based Recommendation with Graph Neural Networks. *ACM Trans. Inf. Syst.* 38, 3, Article 22 (May 2020), 23 pages.

[43] Elisa Quintarelli, Emanuele Rabosio, and Letizia Tanca. 2016. Recommending new items to ephemeral groups using contextual user influence. In *ACM Conference on Recommender Systems*. 285–292.

[44] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 452–461.

[45] Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. 2004. The Author-Topic Model for Authors and Documents. AUAI Press, Arlington, Virginia, USA, 487–494.

[46] Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed. 2018. Deep Inductive Network Representation Learning. In *Companion Proceedings of the The Web Conference 2018* (Lyon, France) *(WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 953–960.

[47] Alan Said, Shlomo Berkovsky, and Ernesto W De Luca. 2011. Group recommendation in context. In *Proceedings of the 2nd challenge on context-aware movie recommendation*. 2–4.

[48] Amirali Salehi-Abari and Craig Boutilier. 2015. Preference-oriented social networks: Group recommendation and inference. In *ACM Conference on Recommender Systems*. 35–42.

[49] Aravind Sankar, Yanhong Wu, Yuhang Wu, Wei Zhang, Hao Yang, and Hari Sundaram. 2020. 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 (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 1279–1288.

[50] Shunichi Seko, Takashi Yagi, Manabu Motegi, and Shinyo Muto. 2011. Group Recommendation Using Feature Space Representing Behavioral Tendency and Power Balance among Members. In *Proceedings of the Fifth ACM Conference on Recommender Systems* (Chicago, Illinois, USA) *(RecSys '11)*. Association for Computing Machinery, New York, NY, USA, 101–108.

[51] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15, 56 (2014), 1929–1958.

[52] Peijie Sun, Le Wu, and Meng Wang. 2018. Attentive recurrent social recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 185–194.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Conference on Neural Information Processing Systems*. 5998–6008.

[54] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*. OpenReview.net.

[55] Lucas Vinh Tran, Tuan-Anh Nguyen Pham, Yi Tay, Yiding Liu, Gao Cong, and Xiaoli Li. 2019. Interact and decide: Medley of sub-attention networks for effective group recommendation. In *ACM SIGIR Conference on Research and Development in Information Retrieval*. 255–264.

[56] Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. 2020. Next-Item Recommendation with Sequential Hypergraphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 1101–1110.

[57] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community preserving network embedding. In *Conference on Artificial Intelligence*.

[58] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 165–174.

[59] Xin Wang, Wenwu Zhu, and Chenghao Liu. 2019. Social Recommendation with Optimal Limited Attention. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 1518–1527.

[60] Yuandong Wang, Hongzhi Yin, Hongxu Chen, Tianyu Wo, Jie Xu, and Kai Zheng. 2019. Origin-Destination Matrix Prediction via Graph Convolution: A New Perspective of Passenger Demand Modeling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD'19)*. Association for Computing Machinery, New York, NY, USA, 1227–1235.

[61] Lin Xiao, Zhang Min, Zhang Yongfeng, and Gu Zhaoquan. 2017. Disparity-Aware Group Formation for Recommendation. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems* (São Paulo, Brazil) *(AAMAS '17)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1604–1606.

[62] Lin Xiao, Zhang Min, Zhang Yongfeng, Gu Zhaoquan, Liu Yiqun, and Ma Shaoping. 2017. Fairness-Aware Group Recommendation with Pareto-Efficiency. In *Proceedings of the Eleventh ACM Conference on Recommender Systems* (Como, Italy) *(RecSys '17)*. Association for Computing Machinery, New York, NY, USA, 107–115.

[63] Lin Xiao, Zhang Min, Zhang Yongfeng, Gu Zhaoquan, Liu Yiqun, and Ma Shaoping. 2017. Fairness-aware group recommendation with pareto-efficiency. In *ACM Conference on Recommender Systems*. 107–115.

[64] Linchuan Xu, Xiaokai Wei, Jiannong Cao, and Philip S Yu. 2017. Embedding identity and interest for social networks. In *The World Wide Web Conference*. 859–860.

[65] Hongzhi Yin, Qinyong Wang, Kai Zheng, Zhixu Li, Jiali Yang, and Xiaofang Zhou. 2019. Social influence-based group representation learning for group recommendation. In *IEEE International Conference on Data Engineering*. IEEE, 566–577.

[66] H. Yin, Q. Wang, K. Zheng, Z. Li, and X. Zhou. 2020. Overcoming Data Sparsity in Group Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2020), 1–1.

[67] Hongzhi Yin, Lei Zou, Quoc Viet Hung Nguyen, Zi Huang, and Xiaofang Zhou. 2018. Joint event-partner recommendation in event-based social networks. In *IEEE International Conference on Data Engineering*. IEEE, 929–940.

[68] Quan Yuan, Gao Cong, and Chin-Yew Lin. 2014. COM: a generative model for group recommendation. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 163–172.