ConsRec: Learning Consensus Behind Interactions for Group Recommendation

Xixi Wu
21210240043@m.fudan.edu.cn
Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
Shanghai, China

Yun Xiong*
yunx@fudan.edu.cn
Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
Shanghai, China

Yao Zhang
yaozhang@fudan.edu.cn
Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
Shanghai, China

Yizhu Jiao
yizhuj2@illinois.edu
University of Illinois at Urbana-Champaign
IL, USA

Jiawei Zhang
jiawei@ifmlab.org
IFM Lab, Department of Computer Science, University of California, Davis
CA, USA

Yangyong Zhu
yyzhu@fudan.edu.cn
Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
Shanghai, China

Philip S. Yu
psyu@uic.edu
University of Illinois at Chicago
IL, USA

ABSTRACT

Since group activities have become very common in daily life, there is an urgent demand for generating recommendations for a group of users, referred to as *group recommendation* task. Existing group recommendation methods usually infer groups' preferences via aggregating diverse members' interests. Actually, groups' ultimate choice involves compromises between members, and finally, an agreement can be reached. However, existing individual information aggregation lacks a holistic group-level consideration, failing to capture the consensus information. Besides, their specific aggregation strategies either suffer from high computational costs or become too coarse-grained to make precise predictions.

To solve the aforementioned limitations, in this paper, we focus on exploring consensus behind group behavior data. To comprehensively capture the group consensus, we innovatively design three distinct views which provide mutually complementary information to enable multi-view learning, including member-level aggregation, item-level tastes, and group-level inherent preferences. To integrate and balance the multi-view information, an adaptive fusion component is further proposed. As to member-level aggregation, different from existing linear or attentive strategies, we design a novel hypergraph neural network that allows for efficient hypergraph convolutional operations to generate expressive member-level aggregation. We evaluate our ConsRec on two real-world datasets and experimental results show that our model outperforms state-of-the-art methods. An extensive case study also verifies the effectiveness of consensus modeling.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

Group Recommendation, Graph Representation Learning, Data Mining

ACM Reference Format:
Xixi Wu, Yun Xiong, Yao Zhang, Yizhu Jiao, Jiawei Zhang, Yangyong Zhu, and Philip S. Yu. 2023. ConsRec: Learning Consensus Behind Interactions for Group Recommendation. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3543507.3583277

1 INTRODUCTION

Owing to the prevalence of social media [24, 25], online group activities have become very common. Various existing online social media sites can provide group-related services for users. For example, travel enthusiasts can plan group trips on Mafengwo[1], and young people can also organize group parties on Meetup[2]. As traditional recommendation methods only target suggesting relevant items to individuals, there is an urgent demand in generating recommendations for a group of users, referred to as the *group recommendation* task [4, 5]. Formally, group recommendation aims to reach an agreement among group members to provide contents that can satisfy most members [39].

Previous works [1–6, 11, 12, 15, 16, 20, 29–31, 38, 39] on group recommendation usually focus on aggregating diverse members' preferences to predict group interests. Traditional aggregation methods are based on pre-defined heuristic rules, including the average [3],

*Corresponding author

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583277

[1] https://www.mafengwo.cn
[2] https://www.meetup.com

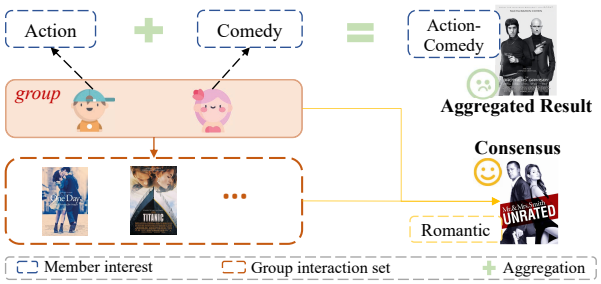

Figure 1: An illustrative example of the gap between aggregated result and group's consensus. Merely aggregating diverse members' interests lacks the holistic consideration of the group's overall taste, failing to capture the consensus.

the least misery [1], and the maximum satisfaction [2]. There also exist several deep-learning based models [4, 5, 12, 15, 31] that incorporate the attention mechanism to realize a learnable aggregation instead. Recently, CubeRec [6] utilizes the geometric expressiveness of hypercubes [26, 40] to aggregate multi-faceted member preferences, achieving state-of-the-art performance.

Despite their effectiveness, we argue that current group recommendation methods fall short in modeling two factors: (1) **Consensus**. An important fact has been neglected: groups' ultimate choice involves compromises between members, and finally, an agreement need to be reached [6, 12]. The consensus drives the group to make the final decision while existing individual information aggregation lacks a holistic group-level consideration, failing to capture the consensus. Taking Figure 1 as an example, the group consists of a couple where the man prefers action films while the lady enjoys comedies. When they plan to watch a movie together, the consensus may be a romantic movie rather than the aggregated action-comedy. Such situations are quite common in real scenes but are largely overlooked by the aforementioned methods. (2) **Member-level Aggregation**. A typical strategy for group interests aggregation is the attentive mechanism [4, 5, 12, 15, 16, 31] that calculates different attention weights for different candidate items. Considering the huge amount of candidate products in real-world scenarios, such methods are hardly applicable. Besides, this approach tends to favor members with frequent interactions since they are likely to obtain higher attention weights [23]. Though CubeRec [6] resorts to hypercubes for a better solution, the obtained hypercube may be too large due to diversity of member interests to make precise predictions. Therefore, an appropriate aggregation strategy that realizes both efficiency and effectiveness is also demanded.

In this paper, we focus on exploring group consensus behind group-item and user-item interactions, so as to improve the performance and meaningfulness of recommendation. We propose a new model, **ConsRec**, to solve the foregoing limitations correspondingly: (1) **Group Consensus Modeling**. To comprehensively capture the group consensus, we innovatively design three distinct views which provide complementary information. Besides the member-level view that provides fine-grained explanation for group consensus, we argue that sometimes the consensus of groups can be captured from the characteristics of interacted items as well as their inherent attributes. For example, a group formed by a family is more inclined to watch family-style comedies. Therefore, we suggest the novel item-level and group-level views to obtain groups' item-level tastes and inherent interests, respectively. For each data view, we design a specific graph structure to encode the behavior data and utilize graph representation learning techniques to generate representations for groups. To integrate and balance multi-view information, an adaptive fusion component is further proposed to synthesize the final group consensus. During optimization, recommendation predictions can supervise the learning process to adaptively adjust the contributions of different views to extract the most discriminative consensus information. (2) **Effective Member-level Aggregation**. We propose a novel hypergraph learning architecture to obtain member-level aggregation. Compared with existing attentive aggregation, our schema wins in efficiency as obtaining aggregation via hypergraph convolutional operations, fairness as designing meaningful side information for balancing the favor of some members, and expressiveness as incorporating high-order collaborative information. Extensive experiments on two public datasets show the effectiveness as well as efficiency of our proposal.

To summarize, the contributions of our work are as follows:

- We study the group recommendation task and reveal the consensus behind groups' decision-making. To comprehensively capture the consensus, we design three novel and complementary views, including member-level aggregation, item-level tastes, and group-level inherent interests.
- We propose a novel hypergraph neural network to obtain member-level aggregation. Compared with existing attentive aggregation, our method wins in efficiency, fairness, and expressiveness.
- Extensive experiments on two public datasets show the effectiveness as well as efficiency of our proposal.

2 RELATED WORK

2.1 Preference Aggregation

Existing group recommendation techniques usually follow the aggregation strategy, which first learns members' preferences from user-item interactions, and then performs preferences aggregation to infer the overall interest of a group [6]. Sometimes, group's ultimate choice is determined by group-level consensus. However, existing methods lack the holistic consideration of the group's overall taste, failing to capture the consensus.

2.1.1 Score Aggregation. The approaches of this category generate the scores of all members in a group for a candidate item and then aggregate individual scores to obtain the preference score of the group through some hand-crafted heuristics, such as the average [3], the least misery [1], and the maximum satisfaction [2]. For example, the average [3] strategy takes the average score across members as the final recommendation score. Though intuitive, these pre-defined aggregation rules are inflexible to reflect diverse member intentions [16].

2.1.2 Neural Aggregation. For better performance, some researchers propose to aggregate different members' intentions in a learnable way [4–6, 12, 15, 30, 31, 38]. Technically, these methods firstly represent each member as an implicit embedding based on user interactions, and then calculate the group embedding by summing

the embeddings of all members with the learned weights. For example, attentive neural networks are proposed in [4] to selectively aggregate user representations within a group, and [31] further captures the fine-grained interactions between group members via a sub-attention network. Recently, CubeRec [6] utilizes the geometric expressiveness of hypercubes to adaptively aggregate the multi-faceted user preferences, achieving better performance.

2.2 Recent Directions

For better representation, recent studies on group recommendation resort to more complex structures, such as hypergraphs [11, 16, 39] and hypercubes [6]. Besides, self-supervised learning techniques are also adopted to counteract the data-sparsity issue [29, 39].

2.2.1 Hypergraph Learning. As a more general topological structure that preserves the tuple-wise relationship, hypergraph naturally fits group profiling. Therefore, recent works [11, 16, 39] represent the group as a hyperedge and employ hypergraph neural networks to generate more expressive vectors. We point out that these methods mainly utilize hypergraph structures to propagate user-level collaborative signals, and exhaustively employ the attentive aggregation [32] for group-level prediction. In fact, expressive member-level aggregation can be obtained more efficiently from hypergraph networks, which is neglected by these methods.

2.2.2 Self-supervised Learning. As self-supervised learning (SSL) has shown its effectiveness in general recommendation tasks [35, 37, 41], attempts are also made to design SSL objectives to counteract the data sparsity issue for group recommendation task [6, 29, 39]. For example, GroupIM [29] proposes a user-group mutual information maximization scheme to jointly learn informative user and group representations.

3 PRELIMINARY

In this section, we present the definition of group recommendation task and the concepts of hypergraph to facilitate comprehension. Formally, we use bold capital letters (e.g., $\mathbf{X}$) and bold lowercase letters (e.g., $\mathbf{x}$) to represent matrices and vectors, respectively. We employ non-bold letters (e.g., x) to denote scalars, and calligraphic letters (e.g., $\mathcal{X}$) to denote sets. Notations are summarized in Table 6 in the Appendix.

3.1 Task Definition

Let $\mathcal{U} = \{u_1, u_2, ..., u_M\}$, $\mathcal{I} = \{i_1, i_2, ..., i_N\}$, and $\mathcal{G} = \{g_1, g_2, ..., g_K\}$ be the sets of users, items, and groups, respectively, where M, N, and K are the sizes of these three sets. There are two types of observed interactions among $\mathcal{U}$, $\mathcal{I}$, and $\mathcal{G}$, namely, group-item interactions and user-item interactions. We use $\mathbf{Y} \in \mathbb{R}^{K \times N}$ to denote the group-item interactions where the element $\mathbf{Y}(t, j) = 1$ if group g_t has interacted with item i_j otherwise $\mathbf{Y}(t, j) = 0$. Likewise, we use $\mathbf{R} \in \mathbb{R}^{M \times N}$ to denote the user-item interactions. The t-th group $g_t \in \mathcal{G}$ consists of a set of user members $\mathcal{G}_t = \{u_1, u_2, ..., u_s, ..., u_{|\mathcal{G}_t|}\}$ where $u_s \in \mathcal{U}$ and $|\mathcal{G}_t|$ is the size of $\mathcal{G}_t$. We denote the interaction set of g_t as $\mathcal{Y}_t = \{i_1, i_2, ..., i_j, ..., i_{|\mathcal{Y}_t|}\}$ where $i_j \in \mathcal{I}$ and $|\mathcal{Y}_t|$ is the size of $\mathcal{Y}_t$. Then, given a target group g_t, the group recommendation task is defined as recommending items that g_t may be interested in.

3.2 Hypergraph

Different from simple graphs, a hypergraph is a more general topological structure where the edge (namely hyperedge in the hypergraph) could connect two or more nodes [7, 33, 37]. Formally, we define the hypergraph as $G = (\mathcal{V}, \mathcal{E}, \mathbf{H})$ where $\mathcal{V}$ is the vertex set, $\mathcal{E}$ is the hyperedge set, and $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$ depicts the connectivity of the hypergraph as $\mathbf{H}(v, e) = 1$ if the hyperedge e connects the vertex v, otherwise $\mathbf{H}(v, e) = 0$. On this basis, we further give some notations in G. Let $\mathcal{E}_v$ denote a set of related hyperedges that connect to the node $v (i.e., \mathcal{E}_v = \{e \in \mathcal{E} | \mathbf{H}(v, e) = 1\})$ and $\mathcal{V}_e$ denote a set of nodes to which the hyperedge e connects (i.e., $\mathcal{V}_e = \{v \in \mathcal{V} | \mathbf{H}(v, e) = 1\}$).

4 METHODOLOGY

In this section, we first elaborate on the multi-view modeling of group consensus. Then, we introduce their specific encoding process enhanced by graph neural networks. We move on to introduce the adaptive fusion mechanism for integrating multi-view information. Finally, we demonstrate our optimization approach. The overall architecture of ConsRec is shown in Figure 2. Notations used in this section are also summarized in Table 6 in the Appendix.

4.1 Multi-view Modeling of Consensus

To effectively exploit the group behavior data for capturing group consensus, we design three novel views which provide complementary information to enable multi-view learning. The illustrative example is shown in the left part of Figure 2.

4.1.1 Member-level. Since groups consist of diverse members, we first analyze each group from a member constitution view, which provides a fine-grained explanation for group decisions. However, conventional graph structure only supports *pairwise* relationship that reflects one-to-one relation. Therefore, we propose to model the *tuplewise* relationship between groups and members with a hypergraph. Formally, we construct the member view with a hypergraph $G^m = (\mathcal{V}^m, \mathcal{E}^m, \mathbf{H}^m)$ where $\mathcal{V}^m = \mathcal{U} \cup \mathcal{I}$ denotes the node set, $\mathcal{E}^m = \mathcal{G}$ denotes the hyperedge set, and the adjacency matrix $\mathbf{H}^m \in \mathbb{R}^{|\mathcal{V}^m| \times |\mathcal{E}^m|}$ denotes the affiliations among nodes and edges. For the t-th group g_t, we represent it as the t-th hyperedge $e_t \in \mathcal{E}^m$ and connect it to corresponding member nodes as well as item nodes, i.e., $\mathbf{H}^m(s, t) = 1$ if node $v_s \in \mathcal{G}_t \cup \mathcal{Y}_t$ where $\mathcal{G}_t$ and $\mathcal{Y}_t$ refer to g_t's member set and interaction set, respectively. For example, in Figure 2, as group g_1 consists of u_1 and u_2, and have interacted with i_1 and i_2, we would represent group g_1 with the hyperedge e_1 that connects to its member and item nodes.

4.1.2 Item-level. Sometimes, the consensus of groups can be inferred from their interacted items. For example, in Figure 1, we can conclude the group's agreement towards romantic films based on its interaction history. Therefore, to capture the groups' item-level tastes, we construct the item-level view where groups and their interacted items form a bipartite graph. Specifically, we design the item view with the graph $G^i = (\mathcal{V}^i, \mathcal{E}^i, \mathbf{A}^i)$ where $\mathcal{V}^i = \mathcal{G} \cup \mathcal{I}$ denotes the node set, $\mathcal{E}^i$ denotes the edge set as $\mathcal{E}^i = \{(g_t, i_j) | g_t \in \mathcal{G}, i_j \in \mathcal{I}, \mathbf{Y}(t, j) = 1\}$, and $\mathbf{A}^i \in \mathbb{R}^{(K+N) \times (K+N)}$ is the adjacency

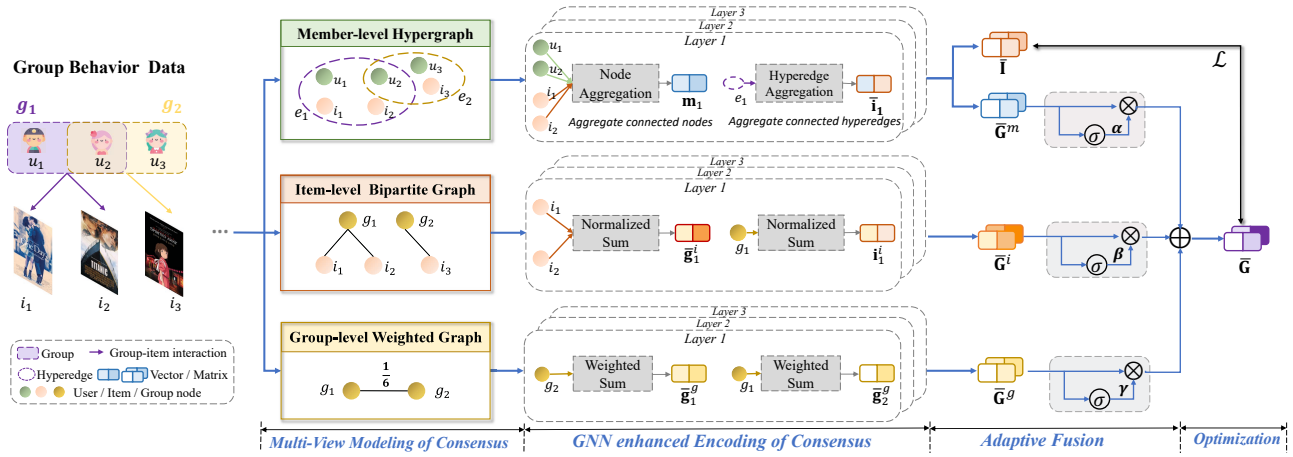

Figure 2: ConsRec Overview. We construct three distinct views for consensus modeling and adopt specific graph neural networks for representation learning. We further integrate these view-specific representations for group-item prediction.

matrix as $\mathbf{A}^i = \begin{bmatrix} \mathbf{0} & \mathbf{Y} \\ \mathbf{Y}^\mathrm{T} & \mathbf{0} \end{bmatrix}$. As shown in Figure 2, we connect node g_2 and i_3 since group g_2 has watched the movie i_3.

4.1.3 Group-level. In real-world scenarios, groups may have their inherent preferences. For example, a group formed by a family tend to watch family-style comedies. To capture and propagate such signals among similar groups, we devise the group-level view where otherwise isolated groups are connected. Specifically, each group is modeled as a single node and different groups are connected if they share at least a common member or item. Formally, the group-level graph is represented as $G^g = (\mathcal{V}^g, \mathcal{E}^g, \mathbf{A}^g)$ where $\mathcal{V}^g = \mathcal{G}$ and $\mathcal{E}^g = \{(g_p, g_q) | g_p, g_q \in \mathcal{G}, |\mathcal{G}_p \cap \mathcal{G}_q| \geq 1 \text{or } |\mathcal{Y}_p \cap \mathcal{Y}_q| \geq 1\}$. We utilize the adjacency matrix $\mathbf{A}^g$ to discriminate relevance between groups as assigning each edge (g_p, g_q) with a weight $\mathbf{A}^g(p,q) = \frac{|\mathcal{G}_p \cap \mathcal{G}_q| + |\mathcal{Y}_p \cap \mathcal{Y}_q|}{|\mathcal{G}_p \cup \mathcal{G}_q| + |\mathcal{Y}_p \cup \mathcal{Y}_q|}$. For example, as depicted in Figure 2, since g_1 and g_2 share a common member, we connect these two groups and compute the weight as $\frac{1+0}{3+3} = \frac{1}{6}$.

4.1.4 Multi-view Modeling. In this paper, we aim to embed users, items, and groups to a lower-dimension vector representation. Based on the above multi-view modeling of group consensus, we can represent the learned d-dimensional embedding vectors of users, items, and groups as matrices $\mathbf{U} \in \mathbb{R}^{M \times d}$, $\mathbf{I} \in \mathbb{R}^{N \times d}$, and $\mathbf{G} \in \mathbb{R}^{K \times d}$, respectively. Such matrices can be learned from the following graph representation learning techniques.

4.2 Member-level Hypergraph Networks

In this subsection, we introduce the member-level view where we aim to obtain a meaningful member-level aggregation for estimating the consensus. Specifically, we devise a novel preference-aware hypergraph neural network that realizes an efficient, fair, and expressive aggregation schema.

4.2.1 Preference-aware hypergraph neural network (P-HGNN). We feed user embeddings $\mathbf{U} \in \mathbb{R}^{M \times d}$ and item embeddings $\mathbf{I} \in \mathbb{R}^{N \times d}$ to the hypergraph neural network. Meanwhile, since we represent each group as a hyperedge e, its embedding can be retrieved from the group embedding table $\mathbf{G}$ as $\mathbf{g}_e = \mathbf{G}(e, :)$. Technically, our goal is to obtain refined item embeddings $\bar{\mathbf{I}}$ and groups' member-level representations $\bar{\mathbf{G}}^m$ via hypergraph propagation. We move on to introduce the specific design of this hypergraph neural network with the illustrative process shown in Figure 3.

Efficiency. In hypergraph neural networks [9, 36], hyperedges serve as mediums for processing and transferring information. Recall that we represent each group as a hyperedge in the member-level hypergraph, it is straightforward to regard the carried messages of hyperedges during convolution as the member-level aggregation. In this way, member-level aggregation can be acquired much more efficiently compared with the exhaustive and candidate-reliant attentive aggregation adopted by [4, 5, 15, 16, 39].

Formally, a hyperedge $e \in \mathcal{E}^m$ connects both member nodes and item nodes which preserve different semantic information. Therefore, we separate the item aggregation and user aggregation to maintain their distinction. Specifically, for hyperedge e, we compute its member aggregated message $\mathbf{m}_{e,u}$ within its member set $\mathcal{G}_e$ as $\mathbf{m}_{e,u} = \mathbf{AGG}_{node}(\{\mathbf{u}_s | u_s \in \mathcal{G}_e\})$. Here $\mathbf{AGG}_{node}(\cdot)$ denotes a specific node-aggregation function and $\mathbf{u}_s \in \mathbb{R}^d$ denotes the embedding of s-th user as $\mathbf{u}_s = \mathbf{U}(s, :)$. Likewise, we obtain e's item-side message $\mathbf{m}_{e,i}$ within its interaction set $\mathcal{Y}_e$ as $\mathbf{m}_{e,i} = \mathbf{AGG}_{node}(\{\mathbf{i}_j | i_j \in \mathcal{Y}_e\})$ where $\mathbf{i}_j$ denotes the embedding of j-th item as $\mathbf{i}_j = \mathbf{I}(j, :)$. Then, it is intuitive to utilize these messages to synthesize the hyperedge e's representation $\mathbf{m}_e$.

Fairness. As shown in Figure 3, the commonly adopted attentive aggregation assigns more weights to active members and thus obtain an unfair aggregation [23]. Hopefully, this problem can be alleviated as we introduce the item-side messages as a supplementary. Furthermore, we utilize the hyperedge e's independent representation $\mathbf{g}_e$ to interact with the item messages $\mathbf{m}_{e,i}$ for generating extra collaborative signals. Then, we fuse these different information via linear transformation as follows:

$$\mathbf{m}_e = \mathbf{CONCAT}(\mathbf{m}_{e,u}\ ,\ \mathbf{m}_{e,i}\ ,\ \mathbf{m}_{e,i} \odot \mathbf{g}_e)\mathbf{W}^f, \tag{1}$$

where $\mathbf{m}_e$ denotes the synthesized messages carried by e, $\odot$ stands for element-wise product, and $\mathbf{W}^f \in \mathbb{R}^{3d \times d}$ denotes the trainable

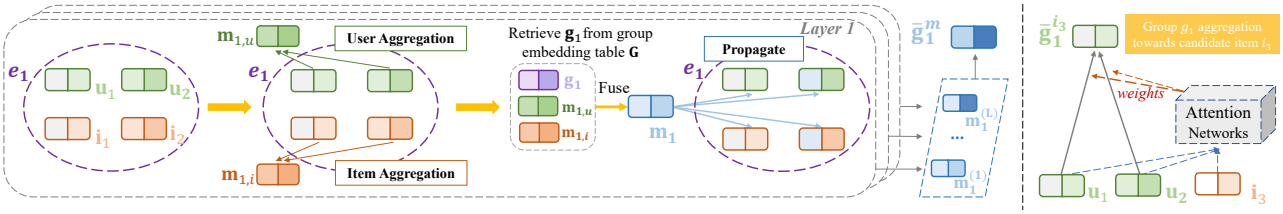

Figure 3: Comparison between our hypergraph learning-based aggregation (left) and the commonly adopted attentive aggregation (right). Ours wins in efficiency, fairness, and expressiveness with details explained in Section 4.2.2.

weight matrix for messages fusion. After applying this operation, the messages contained by hyperedges are more meaningful and unbiased as revealing the common information between members and groups. Finally, we can refine the representations of node v by collecting the messages from its related hyperedges. Formally, for a target item node i_j, its representation is updated as:

$$\bar{\mathbf{i}}_j = \mathbf{AGG}_{he}(\{\mathbf{m}_e | e \in \mathcal{E}_j\}), \tag{2}$$

where $\bar{\mathbf{i}}_j$ denotes the refined embedding of node i_j, $\mathbf{AGG}_{he}(\cdot)$ denotes a specific hyperedge-aggregation function, and $\mathcal{E}_j$ is the set of hyperedges that connect to node i_j. So far, we have accomplished the goal of obtaining refined item embeddings $\bar{\mathbf{I}}$ and groups' member-level representations $\overline{\mathbf{G}}^m$ by stacking all $\bar{\mathbf{i}}_j$ $(j = 1, ..., N)$ and $\mathbf{m}_e$ $(e = 1, ..., K)$, respectively.

Expressiveness. To improve expressiveness, we further stack aforementioned propagation module for multiple layers so that both nodes' representations and hyperedges' messages can benefit from high-order neighbors. Finally, we average the embedding obtained at each layer to generate the final representation for node i_j as:

$$\bar{\mathbf{i}}_j = \frac{1}{L+1}\sum_{l=0}^{L} \mathbf{i}_j^{(l)},$$

where L is the total number of convolutional layers, $\mathbf{i}_j^{(l)}$ is the representation of node i_j in the l-th layer (we have $\mathbf{i}_j^{(0)} = \mathbf{i}_j$ and the calculation of subsequent layers can follow Equations 1 and 2). Similarly, we average the aggregated messages during each layer to generate the final representation for hyperedge (group) e as:

$$\overline{\mathbf{g}}_e^m = \frac{1}{L+1}\sum_{l=0}^{L} \mathbf{m}_e^{(l)},$$

where $\overline{\mathbf{g}}_e^m$ denotes the member-level preferences of group e, and $\mathbf{m}_e^{(l)}$ denotes the messages contained by e in the l-th layer which can be calculated by Equation 1. By stacking all the $\overline{\mathbf{g}}_e^m$ $(e = 1, ..., K)$, we can get the groups' member-level representations $\overline{\mathbf{G}}^m$. Besides, we implement $\mathbf{AGG}_{node}(\cdot)$ and $\mathbf{AGG}_{he}(\cdot)$ with the average pooling due to its simplicity and effectiveness.

4.2.2 Discussions. We compare our aggregation with existing attentive aggregation adopted by [4, 5, 11, 15, 16, 39]. The illustrative example is shown in Figure 3. Given a candidate item, the attentive method first computes the attention weights by feeding the concatenation of member and item representations to an attention network, then sums over members based on the weights to obtain the group's representation. Considering the huge amount of candidate items in real-world scenes, this candidate-reliant method is infeasible. Besides, this mode is prone to assigning active members with more weights, leading to a biased aggregation. On the contrary, our aggregation schema wins in efficiency as obtaining aggregation via hypergraph convolutional operations, fairness as introducing side information for supplementary, and expressiveness as incorporating high-order collaborative information.

4.3 Item-level Graph Networks

Recall that we design the group-item bipartite graph G^i to model the consensus from item-level interests. In this subsection, we aim to employ the graph neural networks [13, 17] to capture collaborative signals between groups and items, finally obtaining the groups' discriminative representations at the item-level.

We feed the concatenation of group embeddings $\mathbf{G} \in \mathbb{R}^{K\times d}$ and item embeddings $\mathbf{I} \in \mathbb{R}^{N\times d}$ to the graph convolutional network, denoted as $\mathbf{E} = \begin{bmatrix}\mathbf{G}\\ \mathbf{I}\end{bmatrix}$. Referring to the spectral graph convolution [13, 17, 34], we define our graph convolution in the l-th layer as:

$$\mathbf{E}^{(l+1)} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}^i\mathbf{D}^{-\frac{1}{2}}\mathbf{E}^{(l)}, \tag{3}$$

where $\mathbf{E}^{(l)}$ denotes the node representations in the l-th layer and $\mathbf{D}$ is the diagonal node degree matrix of adjacency matrix $\mathbf{A}^i$. We pass $\mathbf{E}^{(0)} = \mathbf{E}$ through L convolutional layers, and then average the embeddings obtained at each layer to get the final embeddings as $\overline{\mathbf{E}} = \frac{1}{L+1}\sum_{l=0}^{L}\mathbf{E}^{(l)} = \begin{bmatrix}\overline{\mathbf{G}}^i\\ \overline{\mathbf{I}}^i\end{bmatrix}$. Therefore, group e's item-level consensus can be obtained as $\overline{\mathbf{g}}_e^i = \overline{\mathbf{G}}^i(e, :)$. As shown in Figure 2, during propagation, interacted items' information has been explicitly injected to groups' item-level representations, and thus the obtained $\overline{\mathbf{G}}^i$ is expressive enough to reflect item-level tastes.

4.4 Group-level Graph Networks

Since the constructed group-level graph G^g depicts the connectivity between groups, we still apply graph neural networks to encode the high-order relations among different groups. Note that we distinguish the relevance between groups by computing different weights, allowing each group to enrich its expressivity by absorbing information from the most relevant neighbors.

Specifically, we feed the group embeddings $\mathbf{G} \in \mathbb{R}^{K\times d}$ to the graph convolutional network, denoted as $\mathbf{G}^{(0)} = \mathbf{G}$. Then the propagation mechanism at each layer is similar to Equation 3. Therefore,

after passing $\mathbf{G}^{(0)}$ through L convolutional layers, we can obtain the final group-level embeddings as $\overline{\mathbf{G}}^{g} = \frac{1}{L+1}\sum_{l=0}^{L}\mathbf{G}^{(l)}$. We regard $\overline{\mathbf{G}}^{g}$ as group-level inherent preferences as it explores and preserves the group-level proximity.

4.5 Adaptive Fusion & Optimization

As mentioned before, we have modeled the groups' consensus from three different views, including member-level aggregation $\overline{\mathbf{G}}^{m}$, item-level interests $\overline{\mathbf{G}}^{i}$, and group-level inherent preferences $\overline{\mathbf{G}}^{g}$. Then the core problem takes down to fuse them to obtain the final representations of group consensus. To adaptively control the combination of three view-specific group representations, we propose to employ three different gates as:

$$\overline{\mathbf{G}} = \boldsymbol{\alpha}\overline{\mathbf{G}}^{m} + \boldsymbol{\beta}\overline{\mathbf{G}}^{i} + \boldsymbol{\gamma}\overline{\mathbf{G}}^{g}, \tag{4}$$

where $\boldsymbol{\alpha} = \sigma(\overline{\mathbf{G}}^{m}\mathbf{W}^{m})$, $\boldsymbol{\beta} = \sigma(\overline{\mathbf{G}}^{i}\mathbf{W}^{i})$, and $\boldsymbol{\gamma} = \sigma(\overline{\mathbf{G}}^{g}\mathbf{W}^{g})$. Here, $\mathbf{W}^{m}, \mathbf{W}^{i}$, and $\mathbf{W}^{g} \in \mathbb{R}^{d}$ are three different trainable weights and σ is the activation function. The $\boldsymbol{\alpha}, \boldsymbol{\beta}$, and $\boldsymbol{\gamma}$ denote the learned weights to balance the contributions of member-level, item-level, and group-level information, respectively. On this basis, we can obtain the final group representations $\overline{\mathbf{G}}$. During optimization, based on the abundant supervision signals, our model can automatically discriminate the importance of different views.

Then, we introduce our optimization strategy that jointly learns user-item and group-item interactions. Detailed training procedure is shown in Algorithm 1. For the group-item pair (g_t, i_j), we feed their corresponding representations to a Multi-layer Perceptron (MLP) [28] to compute the final prediction score $\hat{y}_{t,j}$ as $\hat{y}_{tj} = \mathbf{MLP}(\overline{\mathbf{g}}_t \odot \bar{\mathbf{i}}_j)$, where $\overline{\mathbf{g}}_t = \overline{\mathbf{G}}(t,:)$ denotes the final group representation of the target group g_t, $\bar{\mathbf{i}}_j = \bar{\mathbf{I}}(j,:)$ denotes the refined embedding of the candidate item i_j, and $\mathbf{MLP}(\cdot)$ is designed with input dimension set as d and output dimension as 1. With the group-item interaction data, we utilize the *Bayesian Personalized Ranking* (BPR) loss [27] for optimization as follows:

$$\mathcal{L}_{group} = -\sum_{g_t \in \mathcal{G}} \frac{1}{|\mathcal{D}_{g_t}|} \sum_{(j,j') \in \mathcal{D}_{g_t}} \ln \sigma(\hat{y}_{tj} - \hat{y}_{tj'}), \tag{5}$$

where $\mathcal{D}_{g_t}$ denotes the group-item training set sampled for group g_t, in which each instance is a pair (j, j') meaning that group g_t has interacted with item i_j, but has not interacted with item $i_{j'}$.

To further utilize supervision signals, we propose to incorporate the user-item interaction data to optimize the group-item and user-item tasks simultaneously. Similarly, for the user-item pair (u_s, i_j), we compute the prediction score $\hat{r}_{sj} = \mathbf{MLP}(\mathbf{u}_s \odot \mathbf{i}_j)$, where the $\mathbf{MLP}(\cdot)$ is shared with the group-item MLP network, and $\mathbf{u}_s$ and $\mathbf{i}_j$ denote the corresponding user and item embedding. We utilize the same pairwise loss function for optimization as follows:

$$\mathcal{L}_{user} = -\sum_{u_s \in \mathcal{U}} \frac{1}{|\mathcal{D}_{u_s}|} \sum_{(j,j') \in \mathcal{D}_{u_s}} \ln \sigma(\hat{r}_{sj} - \hat{r}_{sj'}), \tag{6}$$

where $\mathcal{D}_{u_s}$ denotes the user-item training set sampled for user u_s and (j, j') represents user u_s prefers observed item i_j over unobserved item $i_{j'}$. We jointly train $\mathcal{L}_{group}$ and $\mathcal{L}_{user}$ on all the group-item and user-item interactions as $\mathcal{L} = \mathcal{L}_{group} + \mathcal{L}_{user}$.

Table 1: Statistics of datasets.

Dataset	#Users	#Items	#Groups	#U-I interactions	#G-I interactions
Mafengwo	5,275	1,513	995	39,761	3,595
CAMRa2011	602	7,710	290	116,344	145,068

5 EXPERIMENT

In this section, we present our experimental setup and empirical results. Our experiments are designed to answer the following research questions (RQs):

- **RQ1:** How does our proposed ConsRec perform compared with various group recommendation methods?
- **RQ2:** Whether the proposed different views capture the consensus information and provide better recommendation?
- **RQ3:** How efficient is our method compared with other group recommendation techniques?
- **RQ4:** How do our preference-aware hypergraph neural network (P-HGNN) work?
- **RQ5:** How do different settings of hyperparameters affect the model performance?

Due to space limitation, we move the RQ4 and 5 to the Appendix.

5.1 Experimental Settings

5.1.1 Datasets. We conduct experiments on two real-world public datasets, Mafengwo and CAMRa2011 [4]. Specifically, Mafengwo is a tourism website where users can record their traveled venues, create or join a group travel. CAMRa2011 is a real-world dataset containing the movie rating records of individual users and households. Table 1 reports the detailed statistics of experimental datasets.

5.1.2 Baselines. To evaluate the effectiveness of ConsRec, we compare it with the following representative approaches. Due to space limitation, we only show their belonging categories and move the details to Appendix B.3. Note that these categories have overlaps. For example, the baseline S^2-HHGR [39] integrates both hypergraph learning and self-supervised learning.

- **Non-personalized method**: Popularity (Pop) [8]
- **Classical neural network based method**: NCF [14]
- **Attentive aggregation method**: AGREE [4]
- **Hypergraph learning-enhanced**: HyperGroup [11], HCR [16]
- **Self-supervised learning-enhanced**: GroupIM [29], S^2-HHGR [39], and CubeRec [6]

5.1.3 Evaluation Metrics. Following previous settings [4, 5, 16], we adopt two widely used evaluation metrics in terms of top-K recommendation, *i.e.*, Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG). Higher HR and NDCG indicate better performance. To alleviate the heavy computation with all items serving as the candidates, we randomly sample 100 negative items for each ground-truth item and rank them according to the calculated interaction probabilities [4, 16]. We evaluate the performance of all methods over the same metrics and test data. Due to space limitation, data pre-processing and implementation details are moved to Appendix B.2.

Table 2: Performance comparison of all methods on group recommendation task in terms of HR@K and NDCG@K.

Dataset	Metric	Pop	NCF	AGREE	HyperGroup	HCR	GroupIM	S^2-HHGR	CubeRec	**ConsRec**
Mafengwo	HR@5	0.3115	0.4701	0.4729	0.5739	0.7759	0.7377	0.7568	0.8613	**0.8844**
	HR@10	0.4251	0.6269	0.6321	0.6482	0.8503	0.8161	0.7779	0.9025	**0.9156**
	NDCG@5	0.2169	0.3657	0.3694	0.4777	0.6611	0.6078	0.7322	0.7574	**0.7692**
	NDCG@10	0.2537	0.4141	0.4203	0.5018	0.6852	0.6330	0.7391	0.7708	**0.7794**
CAMRa2011	HR@5	0.4324	0.5803	0.5879	0.5890	0.5883	**0.6552**	0.6062	0.6400	0.6407
	HR@10	0.5793	0.7693	0.7789	0.7986	0.7821	**0.8407**	0.7903	0.8207	0.8248
	NDCG@5	0.2825	0.3896	0.3933	0.3856	0.4044	0.4310	0.3853	0.4346	**0.4358**
	NDCG@10	0.3302	0.4448	0.4530	0.4538	0.4670	0.4914	0.4453	0.4935	**0.4945**

Table 3: Performance comparison of all methods on user recommendation task in terms of HR@K and NDCG@K.

Dataset	Metric	Pop	NCF	AGREE	HyperGroup	HCR	GroupIM	S^2-HHGR	CubeRec	**ConsRec**
Mafengwo	HR@5	0.4047	0.6363	0.6357	0.7235	0.7571	0.1608	0.6380	0.1847	**0.7725**
	HR@10	0.4971	0.7417	0.7403	0.7759	0.8290	0.2497	0.7520	0.3734	**0.8404**
	NDCG@5	0.2876	0.5432	0.5481	0.6722	0.6703	0.1134	0.4637	0.1099	**0.6884**
	NDCG@10	0.3172	0.5733	0.5738	0.6894	0.6937	0.1420	0.5006	0.1708	**0.7107**
CAMRa2011	HR@5	0.4624	0.6119	0.6196	0.5728	0.6262	0.6113	0.6153	0.5754	**0.6774**
	HR@10	0.6026	0.7894	0.7897	0.7601	0.7924	0.7771	0.8173	0.7827	**0.8412**
	NDCG@5	0.3104	0.4018	0.4098	0.4410	0.4195	0.4064	0.3978	0.3751	**0.4568**
	NDCG@10	0.3560	0.4535	0.4627	0.5016	0.4734	0.4606	0.4641	0.4428	**0.5104**

5.2 Overall Performance (RQ1)

We compare the performance of ConsRec with all baselines. Tables 2 and 3 show the experimental performance on group recommendation task and user recommendation task, respectively. According to the results, we note the following key observations:

- For the group recommendation task shown in Table 2, ConsRec outperforms all baselines under most evaluation metrics on two benchmark datasets. We own our superiority to the proposed three novel views of consensus modeling, capturing the preferences of groups in the most comprehensive manner.
- Among all baselines, hypergraph learning-enhanced models (HyperGroup [11] and HCR [16]) achieve better performance than classical aggregation method (AGREE [4]), indicating the necessity of capturing high-order collaborative information. Besides, S^2-HHGR [39] and CubeRec [6] achieve further improvement due to the introduction of self-supervised learning objectives. Nonetheless, our model realizes the best performance.
- For the user recommendation task shown in Table 3, our model still has advantages. We attribute this to our optimization strategy that boosts group and user recommendation simultaneously. It is worth noting that GroupIM [29] and CubeRec [6], though achieving ideal performance on group recommendation, perform poorly on user recommendation. This is because they separate user-item and group-item training, sacrificing the performance of user recommendation to overfit the group recommendation.

5.3 Effectiveness of Consensus Learning (RQ2)

5.3.1 Multi-view Modeling. To capture the group consensus, we innovatively devise three views, including member-level aggregation, item-level interests, and group-level inherent preferences. To verify whether each view has captured distinct aspect of group information, we further conduct ablation study. Technically, at each time, we remove one single view and only fuse the remaining two views following the same adaptive fusion mechanism as Equation 4. These three variants are denoted as "w/o. M", "w/o. I", and "w/o. G", respectively. We report the experimental results in Table 4. Due to space limitations, we only list the results on Mafengwo as CAMRa2011 shares a similar performance pattern. From this table, we can observe that removing any view degrades the performance, showing that each view plays a distinct role in capturing the consensus.

Table 4: Ablation study on different views with group recommendation results reported. "w/o. M", "w/o. I", and "w/o. G" refer to the variant that eliminates the member-level, item-item, and group-level view, respectively.

Dataset	Metric	w/o. M	w/o. I	w/o. G	**Full**
Mafengwo	HR@5	0.8201	0.8704	0.8593	**0.8844**
	HR@10	0.8724	0.9075	0.9005	**0.9156**
	NDCG@5	0.7021	0.7597	0.7376	**0.7692**
	NDCG@10	0.7192	0.7718	0. 7510	**0.7794**

5.3.2 Case Study. We further conduct a case study to explore how ConsRec captures consensus information on the specific example. However, our two experimental datasets do not contain any semantic information (*e.g.*, item's name or content) to facilitate interpretability. Therefore, we crawl an extra Mafengwo dataset from its official website. This dataset is named "Mafengwo-S" where "S" refers to semantics. We move the detailed crawling process to the Appendix B.1.2. Finally, we obtain a new dataset that contains 11,027 users, 1,215 groups, and 1,236 items. Besides, each item has distinct semantic information, *i.e.*, the location. For comparison, we only

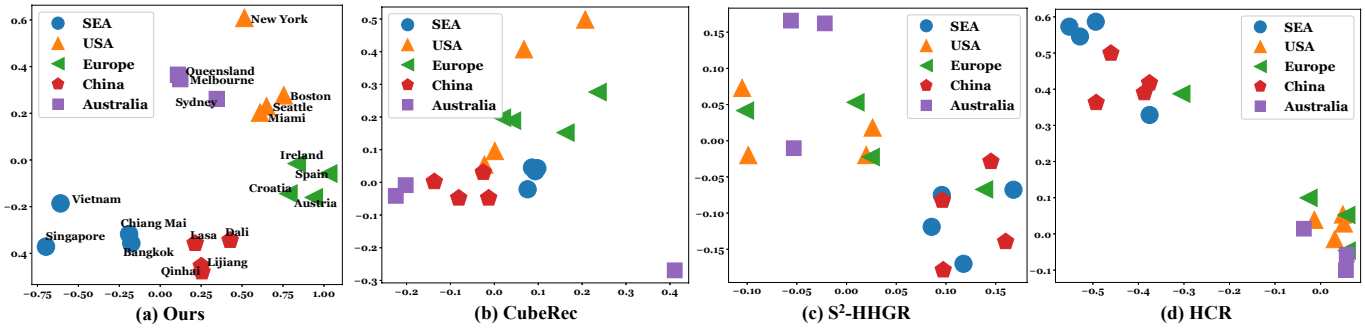

Figure 4: Visualization of learned item embeddings. We plot two dimensions of item representations on Mafengwo-S. ConsRec learns the latent properties of items as geographically similar items are close to each other in the embedding space.

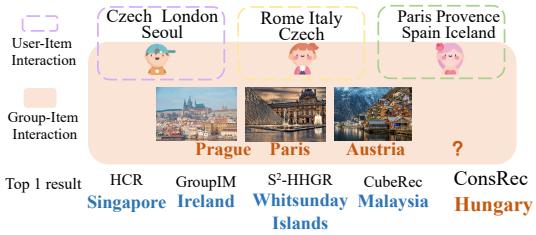

Figure 5: Case study on Mafengwo-S. Both the group and its members have visited European cities. ConsRec captures this consensus and suggests Hungary that hits the ground truth. On the contrary, HCR, GroupIM, S^2-HHGR, and CubeRec are biased by one member's interests towards Iceland and recommend unsatisfying islands or coastal cities.

consider HCR [16], GroupIM [29], S^2-HHGR [39], and CubeRec [6] since other baselines have been shown inferior performance.

As shown in Figure 5, this is a group consisting of three members, and both members and the group have visited European cities. ConsRec captures the group's consensus towards European cities and suggests Hungary that hits the ground truth. HCR, GroupIM, S^2-HHGR, and CubeRec are biased by one member's preference toward Iceland and then recommend unsatisfying islands.

Besides, we also investigate the items' (travel locations) representations learned by different methods. Since GroupIM [29] do not maintain item embedding tables, we only compare ConsRec with HCR [16], S^2-HHGR [39], and CubeRec [6]. Specifically, we select different locations from Southeast Asia (SEA), the USA, Europe, China, and Australia, and visualize their corresponding embeddings in Figure 4. It is worth noting that ConsRec can learn the latent properties of items as geographically similar items are close to each other in the embedding space. On the contrary, the embeddings learned by other baselines are not that discriminative.

We also present the overall performance comparison on the group recommendation task. From Table 5, ConsRec consistently ourperforms other strong baselines, showing its superiority.

5.4 Efficiency Study (RQ3)

We evaluate the efficiency of ConsRec by directly comparing the total running time (training plus testing) with all baselines. Figure 6 shows the performance (NDCG@5) and running time (seconds or minutes) on two experimental datasets. Notably, our method is quite efficient among various group recommendation baselines and it achieves the best performance simultaneously. On CAMRa2011, our advantages are more obvious. Since CAMRa2011 is much denser than Mafengwo, this phenomenon illustrates our superiority in alleviating computational overhead in larger datasets.

Table 5: Performance comparison on group recommendation task on Mafengwo-S dataset.

Metric	HCR	GroupIM	S^2-HHGR	CubeRec	**ConsRec**
HR@5	0.4845	0.5824	0.5928	0.6237	**0.6409**
HR@10	0.6099	0.6959	0.6546	0.6873	**0.6993**
NDCG@5	0.3947	0.4591	0.5348	0.5357	**0.5447**
NDCG@10	0.4353	0.4983	0.5545	0.5567	**0.5642**

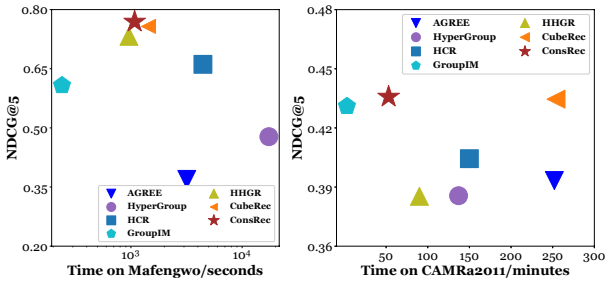

Figure 6: Efficiency Study

6 CONCLUSION

In this paper, we reveal the consensus behind groups' decision-making and propose a novel recommender ConsRec. To capture the consensus information, ConsRec designs three novel views, including member-level aggregation, item-level tastes, and group-level inherent preferences. Especially, in member-level view, we utilize hypergraph learning to realize an efficient and expressive member-level aggregation. Extensive experiments on two real-world datasets show the effectiveness and efficiency of ConsRec.

ACKNOWLEDGMENTS

This work is also partially supported by the National Natural Science Foundation of China Projects No. U1936213, NSF through grants IIS-1763365, IIS-2106972, III-1763325, III-1909323, III-2106758, and SaTC-1930941.

REFERENCES

[1] Sihem Amer-Yahia, Senjuti Basu Roy, Ashish Chawla, Gautam Das, and Cong Yu. 2009. Group Recommendation: Semantics and Efficiency. *Proc. VLDB Endow.* 2 (2009), 754–765.

[2] Linas Baltrunas, Tadas Makcinskas, and Francesco Ricci. 2010. Group recommendations with rank aggregation and collaborative filtering. In *RecSys '10*.

[3] Ludovico Boratto and Salvatore M. Carta. 2011. State-of-the-Art in Group Recommendation and New Approaches for Automatic Identification of Groups. In *Information Retrieval and Mining in Distributed Environments*.

[4] Da Cao, Xiangnan He, Lianhai Miao, Yahui An, Chao Yang, and Richang Hong. 2018. Attentive Group Recommendation. *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (2018).

[5] Da Cao, Xiangnan He, Lianhai Miao, Guangyi Xiao, Hao Chen, and Jiao Xu. 2021. Social-Enhanced Attentive Group Recommendation. *TKDE* 33 (2021), 1195–1209.

[6] Tong Chen, Hongzhi Yin, Jing Long, Quoc Viet Hung Nguyen, Yang Wang, and Meng Wang. 2022. Thinking inside The Box: Learning Hypercube Representations for Group Recommendation. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2022).

[7] Dian Cheng, Jiawei Chen, Wen-Chih Peng, Wen Ye, Fuyu Lv, Tao Zhuang, Xiaoyi Zeng, and Xiangnan He. 2022. IHGNN: Interactive Hypergraph Neural Network for Personalized Product Search. *Proceedings of the ACM Web Conference 2022* (2022).

[8] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys '10*.

[9] Yifan Feng, Haoxuan You, Zizhao Zhang, R. Ji, and Yue Gao. 2019. Hypergraph Neural Networks. *ArXiv* abs/1809.09401 (2019).

[10] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*.

[11] Lei Guo, Hongzhi Yin, Tong Chen, Xiangliang Zhang, and Kai Zheng. 2022. Hierarchical Hyperedge Embedding-based Representation Learning for Group Recommendation. *ACM Trans. Inf. Syst.* 40 (2022), 3:1–3:27.

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

[13] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2020).

[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. *Proceedings of the 26th International Conference on World Wide Web* (2017).

[15] Zhixiang He, Chi-Yin Chow, and Jiadong Zhang. 2020. GAME: Learning Graphical and Attentive Multi-view Embeddings for Occasional Group Recommendation. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2020).

[16] Renqi Jia, Xiaofei Zhou, Linhua Dong, and Shirui Pan. 2021. Hypergraph Convolutional Network for Group Recommendation. *2021 IEEE International Conference on Data Mining (ICDM)* (2021), 260–269.

[17] Thomas Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. *ArXiv* abs/1609.02907 (2017).

[18] Congcong Li, Yucheng Dong, and Francisco Herrera. 2019. A Consensus Model for Large-Scale Linguistic Group Decision Making With a Feedback Recommendation Based on Clustered Personalized Individual Semantics and Opposing Consensus Groups. *IEEE Transactions on Fuzzy Systems* 27 (2019), 221–233.

[19] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*.

[20] Xingjie Liu, Yuan Tian, Mao Ye, and Wang-Chien Lee. 2012. Exploring personal impact for group recommendation. *Proceedings of the 21st ACM international conference on Information and knowledge management* (2012).

[21] Yujia Liu, Changyong Liang, Francisco Chiclana, and Jian Wu. 2017. A trust induced recommendation mechanism for reaching consensus in group decision making. *Knowl. Based Syst.* 119 (2017), 221–231.

[22] Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical Gating Networks for Sequential Recommendation. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2019).

[23] André F. T. Martins and Ramón Fernández Astudillo. 2016. From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification. In *ICML*.

[24] Liqiang Nie, Meng Wang, Yue Gao, Zhengjun Zha, and Tat-Seng Chua. 2013. Beyond Text QA: Multimedia Answer Generation by Harvesting Web Information. *IEEE Transactions on Multimedia* 15 (2013), 426–441.

[25] Liqiang Nie, Yi-Liang Zhao, Xiangyu Wang, Jialie Shen, and Tat-Seng Chua. 2014. Learning to Recommend Descriptive Tags for Questions in Social Forums. *ACM Trans. Inf. Syst.* 32 (2014), 5:1–5:23.

[26] Hongyu Ren, Weihua Hu, and Jure Leskovec. 2020. Query2box: Reasoning over Knowledge Graphs in Vector Space using Box Embeddings. *ArXiv* abs/2002.05969 (2020).

[27] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. *ArXiv* abs/1205.2618 (2009).

[28] Frank Rosenblatt. 1963. PRINCIPLES OF NEURODYNAMICS. PERCEPTRONS AND THE THEORY OF BRAIN MECHANISMS. *American Journal of Psychology* 76 (1963), 705.

[29] Aravind Sankar, Yanhong Wu, Yuhang Wu, Wei Zhang, Hao Yang, and H. Sundaram. 2020. GroupIM: A Mutual Information Maximization Framework for Neural Group Recommendation. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2020).

[30] Young-Duk Seo, Young-Gab Kim, Euijong Lee, Kwangsoo Seol, and Doo-Kwon Baik. 2018. An enhanced aggregation method considering deviations for a group recommendation. *Expert Syst. Appl.* 93 (2018), 299–312.

[31] Lucas Vinh Tran, Tuan-Anh Nguyen Pham, Yi Tay, Yiding Liu, G. Cong, and Xiaoli Li. 2019. Interact and Decide: Medley of Sub-Attention Networks for Effective Group Recommendation. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2019).

[32] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*.

[33] Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. 2020. Next-item Recommendation with Sequential Hypergraphs. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2020).

[34] Felix Wu, Tianyi Zhang, Amauri H. de Souza, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. *ArXiv* abs/1902.07153 (2019).

[35] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised Graph Learning for Recommendation. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2021).

[36] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Pratim Talukdar. 2019. HyperGCN: A New Method For Training Graph Convolutional Networks on Hypergraphs. In *NeurIPS*.

[37] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. 2021. Self-Supervised Multi-Channel Hypergraph Convolutional Network for Social Recommendation. *Proceedings of the Web Conference 2021* (2021).

[38] Quan Yuan, G. Cong, and Chin-Yew Lin. 2014. COM: a generative model for group recommendation. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014).

[39] Junwei Zhang, Min Gao, Junliang Yu, Lei Guo, Jundong Li, and Hongzhi Yin. 2021. Double-Scale Self-Supervised Hypergraph Learning for Group Recommendation. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (2021).

[40] Shuai Zhang, Huoyu Liu, Aston Zhang, Yue Hu, Ce Zhang, Yumeng Li, Tanchao Zhu, Shaojian He, and Wenwu Ou. 2021. Learning User Representations with Hypercuboids for Recommender Systems. *Proceedings of the 14th ACM International Conference on Web Search and Data Mining* (2021).

[41] Kun Zhou, Haibo Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji rong Wen. 2020. S3-Rec: Self-Supervised Learning for Sequential Recommendation with Mutual Information Maximization. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (2020).

A NOTATIONS

We list the important notations used in this paper in Table 6.

Table 6: Important Notations

Symbol	Description
$\mathcal{U}, \mathcal{I}, \mathcal{G}$	Sets of users, items, and groups
M, N, K	Numbers of users, items, and groups
u_s, i_j, g_t	The s-th user, the j-th item, and the t-th group
$\mathbf{Y}$	Group-item interaction matrix
$\mathbf{R}$	User-item interaction matrix
$\mathcal{G}_t$	Member set of the t-th group
$\mathcal{Y}_t$	Interaction set of the t-th group
$\mathbf{U}, \mathbf{I}, \mathbf{G}$	Embedding tables of users, items, and groups
$G^m = (\mathcal{V}^m, \mathcal{E}^m, \mathbf{H}^m)$	Member-level hypergraph
e_t	The t-th hyperedge in G^m
$\mathbf{m}_{e,u}$	Aggregated user messages of hyperedge e
$\mathbf{m}_{e,i}$	Aggregated item messages of hyperedge e
$\mathbf{m}_e^{(l)}$	Messages of hyperedge e in the l-th layer
$\overline{\mathbf{G}}^m$	Group's member-level representations
$\overline{\mathbf{g}}_t^m = \overline{\mathbf{G}}^m(t,:)$	Member-level representation of the t-th group
$G^i = (\mathcal{V}^i, \mathcal{E}^i, \mathbf{A}^i)$	Item-level bipartite graph
$\overline{\mathbf{G}}^i$	Group's item-level representations
$\overline{\mathbf{g}}_t^i = \overline{\mathbf{G}}^i(t,:)$	Item-level representation of the t-th group
$G^g = (\mathcal{V}^g, \mathcal{E}^g, \mathbf{A}^g)$	Group-level graph
$\overline{\mathbf{G}}^g$	Group-level representations
$\overline{\mathbf{g}}_t^g = \overline{\mathbf{G}}^g(t,:)$	Group-level representation of the t-th group
$\bar{\mathbf{I}}$	Refined item embeddings
$\bar{\mathbf{G}}$	Final group embeddings

Algorithm 1: Training Procedure of ConsRec

Input: Sets of users $\mathcal{U}$, items $\mathcal{I}$, and groups $\mathcal{G}$, group-item interaction matrix $\mathbf{Y}$, and user-item interaction matrix $\mathbf{R}$
Output: All model parameters collectively referred to as Θ

Randomly initialize Θ;
while *not converge* **do**
  Randomly draw (g_t, i_j) from $\mathbf{Y}$ and sample negative examples for g_t to constitute $\mathcal{D}_{g_t}$, and compute the group prediction loss $\mathcal{L}_{group}$ *w.r.t.* Equation 5;
  Randomly draw (u_s, i_j) from $\mathbf{R}$ and sample negative examples for u_s to constitute $\mathcal{D}_{u_s}$, and compute the user prediction loss $\mathcal{L}_{user}$ *w.r.t.* Equation 6;
  Take a gradient step to update Θ *w.r.t.* $\mathcal{L}_{group} + \mathcal{L}_{user}$;

B REPRODUCIBILITY

B.1 Datasets

B.1.1 Pre-processing. Following previous works [4, 5, 16], we filter out the groups which have at least 2 members and have interacted with at least 3 items for pre-processing. Since both datasets only contain positive instances, *i.e.*, observed interactions, we randomly sample from missing data as negative instances to pair with each positive instance.

B.1.2 Crawling Process. For the group page in Mafengwo, the group's traveling locations as well as joined members' information can be directly extracted. Then each member's personal visiting records can be obtained from his/her personal page. In this way, we can collect the group-level interacted locations and each member's visited locations. Repeatedly, a new dataset can be constructed that contains rich semantic information of items. Finally, we get a new dataset consisting of 11,027 users, 1,215 groups, and 1,236 items. This dataset is named "Mafengwo-S" with "S" referring to semantics.

B.2 Implementation

We implement ConsRec in PyTorch and optimize with Adam optimizer. For the initialization of the embedding layer, we apply the Glorot initialization strategy [10]. For hidden layers, we randomly initialize their parameters with a Gaussian distribution of a mean of 0 and a standard deviation of 0.1. For hyperparameters, we tune the number of convolutional layers L in $\{1, 2, 3, 4\}$ and the number of negative instances in $\{2, 4, 6, 8, 10, 12\}$. We empirically set the embedding dimension of 32. We implement the $\mathbf{MLP}(\cdot)$ in prediction layer with a 3-layer setting and ReLU activation.

B.3 Baselines

We compare ConsRec with the following baselines:

- **Popularity** [8] is a non-personalized method to benchmark the performance of other personalized methods. It recommends items to users and groups based on the popularity of items, which is measured by the number of interactions in the training set.
- **NCF** [14] is adopted as treating each group as a virtual user for prediction.
- **AGREE** [4] is a classical attentive aggregation-based group recommendation solution that selectively aggregates user representations within a group.
- **HyperGroup** [11] models the group as hyperedge and proposes a hyperedge embedding-based representation learning method.
- **HCR** [16] is a strong group recommendation baseline that proposes a dual channel hypergraph convolutional network to capture member-level and group-level preferences.
- **GroupIM** [29] aggregates users' preferences as group preferences via the attention mechanism. Particularly, to alleviate the data sparsity issue, it adds an extra self-supervised learning objective by maximizing the mutual information between the users and their belonging groups.
- **S^2-HHGR** [39] is another strong baseline that integrates hypergraph learning and self-supervised learning for better group preferences' prediction.
- **CubeRec** [6] is the state-of-the-art deep model in group recommendation. It utilizes the geometric expressiveness of hypercubes to adaptively aggregate members' interests.

For conducting the performance comparison, we use their official codes released at Github: NCF[3], AGREE[4], HCR[5], GroupIM[6], S^2-HHGR[7], and CubeRec[8]. Since the source codes of HyperGroup [11]

[3]https://github.com/hexiangnan/neural_collaborative_filtering
[4]https://github.com/LianHaiMiao/Attentive-Group-Recommendation
[5]https://github.com/GroupRec/GroupRec
[6]https://github.com/CrowdDynamicsLab/GroupIM
[7]https://github.com/0411tony/HHGR
[8]https://github.com/jinglong0407/CubeRec

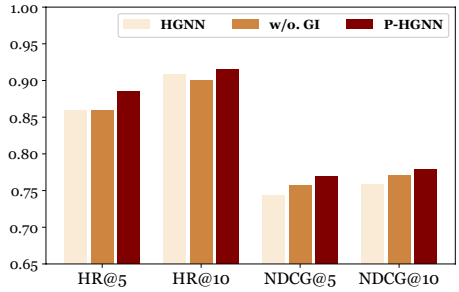

Figure 7: Ablation study on different hypergraph neural networks on Mafengwo with group recommendation results reported.

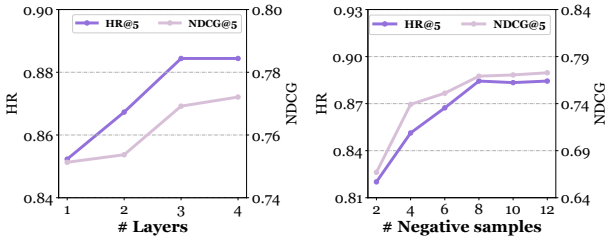

Figure 8: Parameters study on group recommendation task on Mafengwo.

are unavailable, we implement this method based on its original paper. Actually, the released codes of some baselines are either problematic or incomplete, so we refactor their original codes.

To boost the development of group recommendation, we release ConsRec at https://github.com/FDUDSDE/WWW2023ConsRec. We also release our implementations of above group recommendation baselines. For all baseline models, we refer to their best parameter set-ups reported in the original papers. If we use the same datasets and evaluation settings, we directly report their results. We conduct all the experiments on GPU machines of Nvidia Tesla V100 with 32GB memory.

C EXPERIMENTS

C.1 Effectiveness of P-HGNN (RQ4)

Recall that we propose a preference-aware hypergraph neural network to yield a more expressive member-level aggregation. The concrete computation mechanism is shown in Figure 3 and Equations 1 and 2. To verify its effectiveness, we conduct the ablation study by comparing it with two different variants. One is the basic hypergraph neural network (HGNN) [36]. The other variant eliminates group-item element-wise product as $\mathbf{m}_e = \mathbf{CONCAT}(\mathbf{m}_{e,u}, \mathbf{m}_{e,i})\mathbf{W}^f$. This variant is denoted as "w/o. GI".

We show the experimental results in Figure 7. From this figure, "w/o. GI" variant realizes performance improvement compared with "HGNN", showing the necessity of preserving the distinct semantics of users and items. Our P-HGNN outperforms "w/o. GI", indicating the effectiveness of the introduction of group-item element-wise products. In a word, P-HGNN has the advantage of generating a more meaningful member-level aggregation, thus reinforcing groups' representations.

C.2 Parameters Study (RQ5)

In this subsection, we investigate the influence of two key parameters in our model, *i.e.*, the number of (hyper)graph convolutional layers L and the number of negative samples.

C.2.1 Number of Convolutional Layers. The performance of graph convolutional network is affected by the number of graph convolution layers. As the number increases, the convolutional networks are faced with the problem of over-smoothing [19] where node representations are not discriminative enough. To illustrate its influence, we show the performance *w.r.t.* the number of convolutional layers in Figure 8. We observe that when the layer is 3, better results can be obtained on Mafengwo. Therefore, we choose 3 as a default setting.

C.2.2 Number of Negative Samples. The strategy of negative sampling has been proven rational and effective in [22]. It randomly samples various numbers of missing data as negative samples to pair with each positive instance. To illustrate the impact of negative sampling for our model, we show the performance of ConsRec *w.r.t.* different numbers of negative samples in Figure 8. We can observe that too small negative samples are not enough for optimization. With the increase of negative samples, the model performance firstly improves and then becomes stable. Therefore, we choose 8 as a default setting.

D DISCUSSIONS ABOUT CONSENSUS

Though consensus has also been studied in group decision making (GDM) task [18, 21], here we point out the differences between GDM task and group recommendation (GR) task to show why GDM methods can not be directly applied for consensus modeling in GR scenarios.

- Task setting: GDM necessitates each member's preferences on all alternatives (*i.e.*, items) as input, which relies on manual labeling by domain expertise. Instead, GR task automatically learns user-level preferences from user historical behaviors.
- Application scenario: Due to the heavy reliance on human efforts, GDM only works for some specific groups to choose from limited choices. Differently, GR task can deal with large-scale datasets that contain huge amounts of items and groups.