Self-supervised Graph Learning for Occasional Group Recommendation

Bowen Hao  
Renmin University of China  
China  
jeremyhao@ruc.edu.cn

Hongzhi Yin  
The University of Queensland  
Australia  
h.yin1@uq.edu.au

Jing Zhang  
Renmin University of China  
China  
zhang-jing@ruc.edu.cn

Cuiping Li  
Renmin University of China  
China  
llicuiping@ruc.edu.cn

Hong Chen  
Renmin University of China  
China  
chong@ruc.edu.cn

ABSTRACT

We study the problem of recommending items to occasional groups (a.k.a. cold-start groups), where the occasional groups are formed ad-hoc and have few or no historical interacted items. Due to the extreme sparsity issue of the occasional groups’ interactions with items, it is difficult to learn high-quality embeddings for these occasional groups. Despite the recent advances on Graph Neural Networks (GNNs) incorporate high-order collaborative signals to alleviate the problem, the high-order cold-start neighbors are not explicitly considered during the graph convolution in GNNs. This paper proposes a self-supervised graph learning paradigm, which jointly trains the backbone GNN model to reconstruct the group/user/item embeddings under the meta-learning setting, such that it can directly improve the embedding quality and can be easily adapted to the new occasional groups. To further reduce the impact from the cold-start neighbors, we incorporate a self-attention-based meta aggregator to enhance the aggregation ability of each graph convolution step. Besides, we add a contrastive learning (CL) adapter to explicitly consider the correlations between the group and non-group members. Experimental results on three public recommendation datasets show the superiority of our proposed model against the state-of-the-art group recommendation methods.

CCS CONCEPTS

- Information systems → Recommendation.

KEYWORDS

Occasional group recommendation, self-supervised learning, graph neural network

ACM Reference Format:

Bowen Hao, Hongzhi Yin, Jing Zhang, Cuiping Li, and Hong Chen. 2018. Self-supervised Graph Learning for Occasional Group Recommendation. In Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3242135.3242136

1 INTRODUCTION

Recommender systems have played a crucial role in social media platforms, due to their promising ability in mitigating the information overload issue. With the recent advances in social platform services like Meetup and Facebook Event [23, 30], it is increasingly convenient for people with similar backgrounds (e.g., hobbies, locations) to form social groups to participate in activities such as group tours, class reunion, family dinners [9, 40]. The proliferation of groups in the social media platforms demands an effective way to perform group recommendation. This paper addresses the problem of recommending items to occasional groups (a.k.a. cold-start groups), where the occasional groups are formed ad-hoc and have few or no historical interacted items. Due to the extreme sparsity issue of the occasional groups’ interactions with items, it is difficult to learn high-quality embeddings for these occasional groups.

To solve this problem, some early studies adopt heuristic predefined aggregation strategy such as average [2], least misery [1] and maximum satisfaction [3] to aggregate the user preference to obtain the group preference. However, due to the fixed aggregation strategies, these methods are insufficient to capture the complicated and dynamic process of group decision making, which results in the unstable recommendation performance [25]. Further, Cao et al. [4] propose to assign each user an attention weight, which denotes the influence of group member in deciding the group’s choice on the target item. However, when some users in the occasional group...
only interact with few items (a.k.a. cold-start users), the attention weight assigned for each user is diluted by these cold-start users, and thus results in biased group profile.

Recently, inspired by the development of graph neural networks (GNNs), a few GNN-based recommender models are proposed [12, 13, 29, 34]. The basic idea is to incorporate high-order neighbors to enhance the representations of the cold-start users, and then obtain refined group representation. As shown in Figure 1, the GNN model first conducts graph convolution multiple steps on the user-user and user-item interaction graphs to learn the preference of group members, and then performs average [12], summation and pooling [43] or attention mechanism [12] to aggregate the preferences of group members to obtain the group representation. Finally, based on the aggregated embeddings of groups and users, the likelihood of a group/user adopt an item is estimated, and the BPR loss [28] or cross entropy loss [15] is usually adopted to compare the likelihood and the true observations. Moreover, Zhang et al. [44] propose hypergraph convolution network (HHGR) with self-supervised node dropout strategy, which can model complex high-order interactions between groups and users. Through incorporating self-supervised signals, HHGR can alleviate the data sparsity issue in some extent.

However, the above methods still suffer from the following challenges. First, the group representation not only depends on the group members’ preferences, but also relies on the group-level preferences towards items and collaborative group signals (the groups that share common users/items). Although some GNNs consider either group-level preferences [19] or collaborative group signals [12, 44] to form the group representation, they do not consider all these signals together. Second, the GNNs can not explicitly deal with the high-order cold-start neighbors when performing graph convolution. For example in Figure 1, for the target group $g_2$, its group member $u_1$ and high-order neighbor $i_1$ only have few interactions. The embeddings of $u_1$ and $i_1$ are inaccurate, which will affect the embedding of $g_2$ when performing graph convolution. Third, existing GNNs only consider the correlations between the group and its members, but ignore the correlations between the group and non-group members. Thus, this inspires the following research problem: **how can we learn more accurate embeddings by GNNs for occasional group recommendation?**

To this end, motivated by the self-supervised learning (SSL) technique [16, 21, 26], which aims to spontaneously find the supervised signals from the input data itself and can further benefit the downstream tasks, we propose a new Self-supervised Graph Learning paradigm for Group recommendation (SGG), which trains the backbone GNN model to jointly reconstruct the group/user/item embeddings from multiple interaction graphs under the meta-learning setting [33], such that it can directly improve the embedding quality. Specifically, we first pick groups/users/items with sufficient interactions as the target groups/users/items and then learn their ground truth embeddings from the observed abundant interactions. To simulate the cold-start scenarios, in each training episode, we randomly sample $K$ neighbors for each target group/user/item in their counterpart interaction graphs\(^1\), based on which we perform graph convolution multiple steps in each interaction graph, and fuse the corresponding refined embeddings to predict the target embedding. Finally, we jointly optimize the reconstruction losses between the predicted embeddings and the ground truth embeddings of groups/users/items, making the GNN model easily and rapidly being adapted to new cold-start groups/users/items. Through the above process, the reconstructed group representation can contain all the signals as presented in the first challenge.

Nevertheless, the above proposed pretext task still can not explicitly deal with the high-order cold-start neighbors. Besides, the correlations between the groups and non-group members are still not explicitly considered. To further deal with the high-order cold-start neighbors, we incorporate a meta aggregator to enhance the aggregation ability of each graph convolution step. Specifically, the meta aggregator learns cold-start node’s embedding on its first-order neighbors in the counterpart interaction graphs by self-attention mechanism under the same meta-learning setting, which is then incorporated into each graph convolution step to enhance the aggregation ability. To explicitly consider the correlations between groups and non-group members, motivated by Contrastive Learning (CL) [7], which pulls the similar instances together while pulling away dissimilar instances, we propose a CL adapter, which contrasts the group embedding with its members’ and non-members’ embeddings to regularize the group and user embeddings under the same meta-learning setting. The contributions are as follows:

- We design a new self-supervised graph learning paradigm for group recommendation, which jointly trains the backbone GNN model to reconstruct the group/user/item embedding under the meta-learning setting. To deal with the cold-start neighbors, we further introduce a meta aggregator to enhance the aggregation ability of each graph convolution step.
- To explicitly consider the correlations between the group and non-group members, we further propose a CL adapter to regularize the group and user embeddings.
- Experimental results on the group recommendation task demonstrate the superiority of our proposed model against the state-of-the-art group recommendation models.

## 2 PRELIMINARIES

In this section, we first define the problem and then present a base GNN framework that can be used to solve the problem.

### 2.1 Problem Definition

There are three sets of entities in the group recommendation scenario: a user set $U = \{u_1, \ldots, u_{|U|}\}$, an item set $I = \{i_1, \ldots, i_{|I|}\}$ and a group set $G = \{g_1, \ldots, g_{|G|}\}$. There are three kinds of observed interaction graphs among $U$, $I$ and $G$: group-item subgraph $G_{GI}$, user-item subgraph $G_{IU}$ and group-user subgraph $G_{GU}$. Since the social connections of user-user and group-group are also important to depict the user and group profiles, we build two kinds of implicit interaction graphs based on $G_{GI}$ and $G_{IU}$, namely user-user subgraph $G_{UU}$ and group-group subgraph $G_{GG}$. In $G_{IU}$, the two users are connected if they share with more than $c_u$ items. Similarly, in $G_{GG}$, the two groups are connected if they share with more than $c_g$ items. Formally, we use notation $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to denote the set of observed and implicit interaction graphs, i.e., $\mathcal{G} =$ \(^1\)For each group, its counterpart graphs are group-group, group-item and group-user graphs; for each user, his counterpart graphs are user-user and user-item graphs; for each item, its counterpart graph is user-item graph.
Although existing GNN-based group recommendation methods can obtain the refined group embeddings \( h^G \), where the neighbors are sampled from \( G \) at the \( l \)-th graph convolution step, \( \text{CONV} \) can be instantiated from these subgraphs. Same as existing works [12, 19, 19, 44], we further average the first-order neighbors in \( G \) to obtain the aggregated group embedding \( h^G_{G\text{u}} \),

\[
h^G_{G\text{u}} = f_{agg}(h^G_{G\text{u}}(u \in N(g(u)))) ,
\]

where \( h^G_{G\text{u}} \) is obtained by performing graph convolution \( L \) steps in \( G \), \( N(g(u)) \) denotes the first-order user set sampled from \( G \), \( f_{agg} \) is the aggregate function such as average [12], summation and pooling [43], or attention mechanism [32]. In our experiments, we find attention mechanism leads to the best performance. Finally, we use attention mechanism to aggregate the above embeddings to form the final group embedding \( h^G_g \):

\[
h^G_g = \sum_{c \in \{G, G_u, G_u^u\}} a_c h^G_{g,c}, \quad a_c = \frac{\exp(W_c h^G_{g,c})}{\sum_{c' \in \{G, G_u, G_u^u\}} \exp(W_{c'} h^G_{g,c'})},
\]

where \( \{W_c|c \in \{G, G_u, G_u^u\}\} \) are trainable parameters, \( \{a_c|c \in \{G, G_u, G_u^u\}\} \) are the learned attention weights.

### 2.2.2 Jointly Training Module.

This module jointly optimizes the user preferences over items with the user-item loss \( \mathcal{L}_u \), and the group preferences over items with the group-item loss \( \mathcal{L}_g \), i.e.,

\[
\mathcal{L}_{\text{main}} = \mathcal{L}_u + \lambda \mathcal{L}_g, \quad \mathcal{L}_u = \sum_{(u,i) \in \mathcal{E}_{\text{UI}}} -\log \sigma(y(u,i) - y(u,j)), \quad \mathcal{L}_g = \sum_{(g,i) \in \mathcal{E}_{\text{GI}}} -\log \sigma(y(g,i) - y(g,j)),
\]

where \( y(u,i) = h^T_i \sigma h^T_u \), \( y(g,i) = h^T_i \sigma h^T_g \), \( \sigma \) is an activation function, \( \mathcal{E}_{\text{UI}} \) and \( \mathcal{E}_{\text{GI}} \) represent the edges in \( G_{\text{UI}} \) and \( G_{\text{GI}} \).

Although the above presented GNNs can address the occasional groups through incorporating high-order collaborative signals, they still can not deal with the groups/users/items with few interactions, and thus can not learn high-quality embeddings for them.

### 3 THE PROPOSED MODEL

We present the proposed self-supervised graph learning paradigm for group recommendation (SGG). We first describe the process of embedding reconstruction with GNN, and then explain a meta aggregator and a CL adapter that are incorporated in the GNN model to further improve the embedding quality. Finally, we explain how SGG is trained and analyze its time complexity. The overall framework of SGG is shown in Figure 2.

#### 3.1 Embedding Reconstruction with GNN

We propose embedding reconstruction with GNN, which jointly reconstructs the groups/users/items embeddings from multiple subgraphs under the meta-learning setting. Here we take the group embedding reconstruction as an example. User and item embedding reconstruction can be explained in the same way. To achieve this...
Figure 2: The overall framework of SGG for group recommendation. SGG contains a meta aggregator which has incorporated a self-attention-based meta learner at each step of the original GNN aggregation, and a CL adapter, which explicitly captures the correlations between the group and non-group members.

goal, we need abundant occasional groups as the training instances. Since we also need ground truth embeddings of the occasional groups to learn $f$, we simulate those groups from the target groups with abundant interactions with items. The ground truth embeddings for each group $g$, i.e., $h_g$, is learned upon the observed abundant interactions by any group recommendation model such as AGREE [4] or LightGCN [17] \(^2\). To mimic the occasional groups, in each training episode, for each target group, we randomly sample $K$ items, $K$ users and $K$ groups from the corresponding group-item subgraph $G_{GI}$, group-user subgraph $G_{GU}$ and group-group subgraph $G_{GG}$. Then for each subgraph, we repeat the sampling process $L$ steps from the target group to the $L$-order neighbors, which results in at most $K^L(1 \leq l \leq L)$-order neighbors for each target group. Next we perform graph convolution $L$ steps from scratch using Eq. (1) to obtain the refined group embeddings $h^L_{GI}$, $h^L_{GU}$ and $h^L_{GG}$, use Eq. (2) to obtain the aggregated group embedding $h^L_{GI,GG}$, and use Eq. (3) to obtain the fused group embedding $h^L$. Finally, same with \([16, 20]\), we use cosine similarity to measure the similarity between the predicted embedding $h^L$ and the ground-truth embedding $h_g$, as the cosine similarity is a popularity indicator for the semantic similarity between embeddings:

$$L_g = \arg\max_{\Theta_f} \sum_g \cos(h^L, h_g),$$

where $\Theta_f$ is the set of the parameters in $f$. Similarly, we can reconstruct the user embedding based on $G_{UI}$ and $G_{UU}$ with loss $L_u$, and reconstruct the item embedding based on $G_{II}$ with loss $L_i$. In practice, we jointly optimize group/user/item embedding reconstruction tasks with loss $L_R$:

$$L_R = L_g + L_u + L_i.$$  \hspace{1cm} (6)

Training GNNs in the meta-learning setting can explicitly reconstruct the group embeddings, making GNNs easily and rapidly being adapted to new occasional groups. After the model is trained, for a new arriving occasional group, based on its first- and high-order neighbors, we can predict an accurate embedding for it. However, the basic embedding reconstruction task does not specially address the cold-start neighbors. During the original graph convolution process, the inaccurate embeddings of the cold-start neighbors and the embeddings of other neighbors are equally treated and aggregated to represent the target group. Although some GNN models such as GrageSAGE [14] or FastGCN [6] filter neighbors before aggregating them, they usually follow the random or importance sampling strategies, which ignores the cold-start characteristics of the neighbors. Out of this consideration, we incorporate a meta aggregator into the above embedding reconstruction model.

3.2 Meta Aggregator

We propose the meta aggregator to deal with the high-order cold-start neighbors. Specifically, before training the GNN $f$, we train another function $f_{meta}$ under the same meta-learning setting as proposed in Section 3.1. The meta aggregator $f_{meta}$ learns an additional embedding for each node only based on its first-order neighbors sampled from the counterpart graphs, thus it can quickly adapt to new cold-start nodes and produce more accurate embeddings for them. The embedding produced by $f_{meta}$ is combined with the original embedding at each convolution in $f$. Although both the GNN model and the meta aggregator are trained under the same meta-learning setting, the GNN model is to tackle the cold-start target nodes, but the meta aggregator is to enhance the cold-start neighbors’ embeddings.

Here we take group embedding as an example. User and item embedding can be explained in a similar way. Specifically, we instantiate $f_{meta}$ as a self-attention encoder \([32]\). For each group $g$, $f_{meta}$ accepts the randomly initialized first-order embeddings $(h^0_{gGI}, \ldots, h^0_{gGI_k})$, $(h^0_{gGU}, \ldots, h^0_{gGU_k})$ and $(h^0_{gGG}, \ldots, h^0_{gGG_k})$ from the corresponding subgraphs $G_{GI}$, $G_{GU}$ and $G_{GG}$ as input, calculates the attention scores of all the neighbors to each neighbor of $g$, aggregates all the neighbors’ embeddings according to the attention scores to produce the smoothed embeddings $(\hat{h}^{\prime}_{gGI}, \ldots, \hat{h}^{\prime}_{gGI_k})$, $(\hat{h}^{\prime}_{gGU}, \ldots, \hat{h}^{\prime}_{gGU_k})$ and $(\hat{h}^{\prime}_{gGG}, \ldots, \hat{h}^{\prime}_{gGG_k})$. The process is:

$$(h^0_{GI_k}, \ldots, h^0_{GI_k}) \leftarrow \text{SELF_ATTENTION}((h^0_{GI_k}, \ldots, h^0_{GI_k})), \quad \hat{h}^\prime_{gGI} = \text{AVERAGE}((\hat{h}^{\prime}_{gGI}, \ldots, \hat{h}^{\prime}_{gGI_k})), \hspace{1cm} (7)$$

where the embedding $\hat{h}^\prime_{gGI}$ and $h^{\prime}_{gGI_k}$ can be obtained in the same way. Furthermore, the aggregated group embedding $h^L_{GI,GG}$ in Eq. (2) is also considered to reconstruct the group embedding. Finally, $f_{meta}$ fuses these embeddings using Eq. (3) to obtain the predicted embedding $\hat{h}_g$ as the meta embedding of group $g$.

The self-attention technique, which pushes the dissimilar neighbors further apart and pulls the similar neighbors closer together, can capture the major preference of the nodes from its neighbors. The same cosine similarity described in Eq.(5) is used as the loss function to measure the similarity between the predicted meta embedding $\hat{h}_g$ and the ground truth embedding $h_g$. Once $f_{meta}$ is learned, we add the meta embedding $\hat{h}^\prime_{gGI}$, $\hat{h}^\prime_{gGU}$ and $\hat{h}^\prime_{gGG}$ into each graph convolution step of the GNN $f$ in Eq. (1):

\[^2\]They are good enough to learn high-quality group embeddings from the abundant interactions.
For a target group $g$, Eq. (8) is repeated $L$ steps to obtain the embeddings $\hat{h}_g^L$, $\hat{h}_U^L$, $\hat{h}_G^L$, Eq. (2) is used to obtain the aggregated group embedding $\hat{h}_{gG}^L$, and Eq. (3) is applied to get the final embedding $\hat{h}_g^L$. Finally, the same cosine similarity in Eq. (5) is used for the meta item embedding on $\hat{G}_G$, and the meta user embedding on $\hat{G}_U$, and the meta item embedding on $\hat{G}_U$.

3.3 CL Adapter

To further consider the correlations between the group and non-group members, motivated by the contrastive learning technique \cite{mikolov2013distributed}, which pulls the similar instances together while pulling away dissimilar instances, we propose a CL adapter, which contrasts the group embedding with its members/’non-members’ embeddings to regularize the group and user embeddings under the same meta-learning setting. The group and its members form the positive pair, while the group and its non-group members form the negative pair. The CL loss is defined as:

$$L_M = \sum_g \sum_{u \in N(g_{CU})} -\log \frac{\exp(\cos(h_g^L, h_u^L)/\tau)}{\sum_{u' \in \hat{U}, \hat{N}(g_{CU})} \exp(\cos(h_g^L, h_{u'}^L)/\tau)},$$

where $N(g_{CU})$ denotes the group member set of group $g$, $u$ denotes the group member, $u'$ denotes non-group member, where for each $u$, we sample at most $K$ instances for $u'$. \text{\textbackslash} defines set subtraction operation. $h_g^L$, $h_u^L$, $h_{u'}^L$ are the reconstructed embeddings, and $\tau$ is a temperature parameter.

3.4 Multi-task Training

To improve recommendation with the SSL task, we leverage a multi-task training strategy \cite{peng2016joint} to jointly optimize the classic recommendation task (cf. Eq. (4)) and the self-supervised learning task (cf. Eq. (6) and Eq. (9)):

$$L = L_{main} + \lambda_1 (L_R + L_M) + \lambda_2 ||\Theta||_2^2,$$

where $\Theta = \{\Theta_f, \Theta_{f\text{meta}}\}$ is the model parameters, $\lambda_1$ and $\lambda_2$ are hyperparameters to control the strengths of SGG and L2 regularization, respectively. We also consider the alternative optimization – pre-training on $L_R + L_M$ and fine-tuning on $L_{main}$. We detail its recommendation performance in Section 4.4.3.

3.5 Complexity Analysis

Here we present the time and space complexity of SGG. Same as LightGCN \cite{he2020lightgcn}, we implement SGG as the matrix form. Suppose the number of edges in the interaction graph $G$ is $|E|$. The number of edges in the masked interaction graph $\hat{G}$ is $|\hat{E}|$. Since we mask a large proportion neighbors for each node in $G$ to simulate the cold-start scenario, the size of masked edge set is far less than the original edge set, i.e., $|\hat{E}| \ll |E|$. Let $s$ denote the number of epochs, $d$ denote the embedding size and $L$ denote the number of

Table 1: The comparison of analytical time complexity between GNN and SGG.

| Component | GNN | SGG |
|-----------|-----|-----|
| Adjacency Matrix | $O(2|E|)$ | $O(10|E| + 2|E|)$ |
| Graph Convolution | $O(2|E||L_d|s^2)$ | $O(2|E||L_d|s^2)$ |
| BPR Loss | $O(2|E|ds)$ | $O(2|E|ds)$ |
| Self-supervised Loss | - | $O(24|\hat{E}|L_d)$ |

SGG introduces meta embedding to enhance the aggregation ability, the space complexity of SGG is twice than that of the vanilla GNN model. The time complexity comes from four parts, namely normalization of adjacency matrix, graph convolution, recommendation loss and self-supervised loss. As there is no change on the model structure and inference process, the time complexity of SGG in the graph convolution and recommendation loss is the same as the vanilla GNN model. We present the main differences between the vanilla GNN and SGG models as follows:

- Normalization of adjacency matrix. Since we generate five independent subgraphs per epoch, given the fact that the number of non-zero elements in the adjacency matrices of full training graph and the five subgraphs are $2|E|$, $2|\hat{E}_{UUI}|$, $2|\hat{E}_{UU}|$, $2|\hat{E}_{GG}|$, $2|\hat{E}_{GU}|$ and $2|\hat{E}_{GG}|$, respectively, its total complexity is $O(2|\hat{E}_{UU}| + 2|\hat{E}_{UUI}| + 2|\hat{E}_{UU}| + 2|\hat{E}_{GG}| + 2|\hat{E}_{GU}| + 2|\hat{E}_{GG}|s + 2|E|) \approx 10|\hat{E}|$.
- Self-supervised loss. We evaluate the self-supervised tasks upon the masked subgraphs. For the user or item embedding reconstruction task, the time complexity is $O(2d + (2|\hat{E}_{UU}| + 2|\hat{E}_{UUI}|)s + L) \approx 8|\hat{E}||L_d|$. For the group embedding reconstruction task, the time complexity is $O(2d + (2|\hat{E}_{GG}| + 2|\hat{E}_{GU}| + 2|\hat{E}_{GG}|)s + L) \approx 12|\hat{E}||L_d|$. Where $2d$ represents the concatenated embedding size, as we incorporate the meta embedding to the graph convolution process. For the mutual information maximization task, the time complexity is $O(2d + 2|\hat{E}_{GG}|s + L) \approx 4|\hat{E}||L_d|$. Thus the total time complexity of self-supervised loss is $24|\hat{E}||L_d|$.

We summarize the time complexity between the vanilla GNNs and SGG in Table 1, from which we observe that the time complexity of SGG is in the same magnitude with the vanilla GNNs, which is totally acceptable, since the increased time complexity of SGG is only from the self-supervised loss. The details are shown in Section 4.4.1.

4 EXPERIMENTS

To verify the superiority and effectiveness of SGG, we conduct extensive experiments and answer the following research questions:

- **RQ1**: How does SGG perform occasional group recommendation compared with the state-of-the-art GNN models?
- **RQ2**: What are the benefits of performing pretext tasks in occasional group recommendation?
- **RQ3**: How do different settings influence the effectiveness of the proposed SGG model?

\[\text{We use notation } |\hat{E}| \text{ to denote the masked edge length } |\hat{E}_{UUI}|, |\hat{E}_{UUI}|, |\hat{E}_{GG}|, |\hat{E}_{GU}| \text{ and } |\hat{E}_{GG}| \text{ for simplicity.}\]
Table 2: Statistics of the Datasets.

|                | Weeplaces | CAMRa2011 | Douban |
|----------------|-----------|-----------|--------|
| Users          | 8,643     | 602       | 70,743 |
| Items          | 25,081    | 7,710     | 60,028 |
| Groups         | 22,733    | 290       | 109,538|
| U-I Interactions| 1,358,458 | 116,314   | 3,422,266|
| G-I Interactions| 180,229  | 145,068   | 164,153|
| U-I Sparsity   | 6.27%     | 2.51%     | 0.081% |
| G-I Sparsity   | 0.03%     | 6.49%     | 0.002% |

4.1 Experimental Settings

4.1.1 Datasets. We evaluate on three public datasets including Weeplaces [29], CAMRa2011 [4] and Douban [39]. Table 1 illustrates the statistics of these datasets.

4.1.2 Baselines. We select three types of baselines including the state-of-the-art attention based model, the general GNN models and the hypergraph GNN models for group recommendation:

- **MoSAN** [31]: adopts sub-attention mechanism to model the group-item interactions.
- **AGREE** [4]: adopts attention mechanism for jointly modelling user-item and group-item interactions.
- **SIMG** [39]: further incorporates social relationships of groups and users to model the attentive group and user representations.
- **GroupIM** [29]: further regularizes group and user representations by maximizing the mutual information between the group and its members.
- **GAME** [19]: performs graph convolution only based on the first-order neighbors from the group-group, group-user and group-item graphs for group recommendation.
- **GCMC** [37]: employs the standard GCN [22] model to learn the node embeddings.
- **NGCF** [35]: adds second-order interactions upon the neural passing based GNN model [10].
- **LightGCN** [17]: devises the light graph convolution upon NGCF.
- **HHGR** [44]: designs coarse- and fine-grained node dropout strategies upon the hypergraph for group recommendation.

We discard potential baselines like Popularity [8], COM [42], and CrowdRec [27], since previous works [4, 19, 29, 31, 39] have validated the superiority over the compared ones. For the GNN model GCMC, NGCF and LightGCN, we extend it to address group recommendation as proposed in Section 2.2. Besides, we use notation GNN* to denote the corresponding proposed model SGG. We further evaluate three variants of SGG, named Basic-GNN, Meta-GNN and CL-GNN, which are equipped with basic embedding reconstruction with GNNs (Section 3.1), meta aggregator (Section 3.2) and only CL adapter (Section 3.3), respectively.

4.1.3 Training Settings. We present the details of dataset segmentation, model training process and hyper-parameter settings.

Dataset Segmentation. We first select the groups/users with sufficient interactions as the target groups/users in the meta-training set $D_T^g/D_T^u$, and obtain the meta-training set $D_T^g/D_T^u$, as we need the true embeddings of groups/users inferred from the sufficient interactions. In order to avoid information leakage, we further select items with sufficient interactions from $D_T^g/D_T^u$, and obtain the meta-training set $D_T^g/D_T^u$. For simplicity, we use $D_T$ and $D_N$ to denote these meta-training and meta-test sets. For each group/user in $D_N$, we further select top c% of its/his interacted items in chronological order into the training set $Train_N$, and leave the rest items into the test set $Test_N$.

For addressing the occasional groups, we divide the groups with the number of direct interacted items more than $n_g$ into $DT$ and leave the rest groups into $DN$. We select $n_g$ as 10 for Weeplaces and Douban. Similarly, we divide the users (or items) with the number of direct interacted items (users) more than $n_u$ ($n_i$) into $DT$ and leave the rest users (items) into $DN$, where both $n_u$ and $n_i$ is set as 10 for Weeplaces and Douban. In CAMRa2011, since the groups, users and items have abundant interactions, we randomly select 70% groups, users and items in $DT$ and leave the rest in $DN$. For each group and user in $DN$, we only keep top 10 interacted items in chronological order to simulate the real cold-start scenario. Similarly, for each item in $DN$, we only keep its first 5 interacted groups/users.

Model Training Process. We train each of the baseline methods to obtain the ground-truth embeddings on $DT$, as these methods are good enough to learn high-quality embeddings from the abundant interactions. For MoSAN, AGREE, SIRG, GroupIM and GAME, we directly fetch the trained embeddings as ground-truth embeddings; For HHGR, GCMC, NGCF and LightGCN, we combine the embeddings obtained at each layer to form the ground-truth embeddings, take the group embedding as an example, $h_g = h^0_g + \cdots + h^h_g$. User and item embeddings can be explained in the similar way.

The SSL tasks is trained on $DT$, while the recommendation task is trained on $DT$ and $Train_N$. Both the SSL and the recommendation tasks are evaluated in $Test_N$. We adopt Recall@K and NDCG@K as the metrics to evaluate the items ranked by the relevance scores.

Hyper-parameter Settings. For fair comparison, all models are trained from scratch which are initialized with the Xavier method [11]. The learning rate is 0.001 and mini-batch size is 256. We tune $\lambda_1$ with the ranges of $\{0.01, 0.1, 0.5, 1.0, 1.2\}$, and empirically set $\lambda_2$ as 1 and $1e-6$, respectively. We tune $\gamma_u$ and $\gamma_g$ with the ranges of $\{0.1, 0.2, 0.3\}$, respectively. We tune $\lambda_1$ with the ranges of $\{0.01, 0.1, 0.5, 1.0, 1.2\}$, and empirically set $\lambda_2$ as 1 and $1e-6$, respectively. We tune $\gamma_u$ and $\gamma_g$ with the ranges of $\{0.1, 0.2, 0.3\}$, respectively.

4.2 Performance Comparison (RQ1)

4.2.1 Overall Performance Comparison. We report the overall recommendation performance in Table 3. The results show that compared with other baselines, our proposed SGG (denoted as GNN*) significantly improves the recommendation performance, which indicates the proposed SSL tasks are useful to learn high-quality embeddings, and can further benefit the recommendation task. Besides, SGG is better than the most competitive baseline method HHGR, which indicates the superiority of the proposed SSL tasks in dealing with high-order cold-start neighbors.

4.2.2 Interacted Number and Sparse Rate Analysis. It is still unclear how does SGG handle the cold-start groups and users with different interacted items ($n_g$ and $n_u$) and different sparse rate $c\%$. To this end,
we change $n_g$ and $n_u$ in the range of $\{5, 10, 15\}$ while keeping $c\%$ as 0.1, $L$ as 3 and $K$ as 5; and change $c\%$ in the range of $\{0.1, 0.2, 0.3\}$ while keeping $n_g$ and $n_u$ as $5, L$ as 3 and $K$ as 5. We compare our proposed model SGG (denoted as LightGCN+, in which we select LightGCN as the backbone GNN model) with competitive baseline methods AGREE, GroupIM, HHGR and LightGCN, and report the recommendation performance in Figure 3. The smaller $n_g$, $n_u$ and $c\%$ are, the groups and users in $D_N$ have fewer interactions. Based on the results, we find that: (1) LightGCN+ is consistently superior to all the other baselines, which justifies the superiority of SGG in handling cold-start recommendation with different $n_g$, $n_u$ and $c\%$. (2) When $n_g$ and $n_u$ decrease from 15 to 5, and when $c\%$ decreases from 0.3 to 0.1, SGG all has a larger improvement compared with other baselines, which verifies its capability to solve the cold-start groups/users with extremely sparse interactions.

4.3 Ablation Study (RQ2)

It is still not clear which part of the pretext tasks is responsible for the good performance in SGG. To answer this question, we conduct an ablation study to investigate the recommendation performance of SGG and its variant models in Table 4. We find that: (1) Basic-GNN, Meta-GNN and CL-GNN is consistently superior than the vanilla GNNs, which indicates the effectiveness of the proposed SSL tasks. (2) Among all the variant models, Meta-GNN performs the best, which indicates enhancing the cold-start neighbors’ embedding quality is much more important. (3) GNN+ performs the best, which verifies the superiority of combining these SSL tasks.

### 4.4 Study of SGG (RQ3)

#### 4.4.1 Effectiveness of Meta-Learning Setting

As mentioned in Section 3.1, we train SGG under the meta-learning setting. To explore whether the meta-learning setting can benefit the recommendation performance and have satisfactory time complexity, we compare SGG and the vanilla GNN model with a variant model SGG-M, which removes the meta-learning setting. More concretely, in SGG-M, for each group/user/item, we do not sample $K$, which removes the meta-learning setting. To explore

![Figure 3: Recommendation performance under different interacted numbers $n_g$ and $n_u$ (shown in Fig 3(a)), and under different sparse rate $c\%$ (shown in Fig 3(b)). Results on CAMRa2011 and Douban show the same trend which are omitted for space.](image)

Table 3: Overall recommendation performance with sparse rate $c\% = 0.1$, layer size $L=3$ and neighbor size $K=5$.

| Methods   | Weeplaces | CAMRa2011 | Douban |
|-----------|-----------|-----------|--------|
|           | Recall    | NDCG      | Recall | NDCG |
| MoSAN     | 0.0232    | 0.0208    | 0.0214 | 0.0166 |
| AGREE     | 0.0266    | 0.0233    | 0.0237 | 0.0168 |
| GroupIM   | 0.0228    | 0.0283    | 0.0277 | 0.0169 |
| Game      | 0.0312    | 0.0349    | 0.0348 | 0.0171 |
| LightGCN  | 0.0316    | 0.0323    | 0.0106 | 0.0183 |
| HHGR      | 0.0488    | 0.0422    | 0.1494 | 0.0576 |
| SGG       | 0.0113    | 0.0148    | 0.1112 | 0.0394 |
| LightGCN+ | 0.0232    | 0.0426    | 0.1634 | 0.0353 |

Table 4: The comparison of different SGG variants with sparse rate $c\% = 0.1$, layer size $L=3$ and neighbor size $K=5$.

| Methods   | Weeplaces | CAMRa2011 | Douban |
|-----------|-----------|-----------|--------|
|           | Recall    | NDCG      | Recall | NDCG |
| GCMC      | 0.0312    | 0.0083    | 0.0048 | 0.0171 |
| CL-GCMC   | 0.0333    | 0.0127    | 0.0075 | 0.0216 |
| Basic-GCMC| 0.0412    | 0.0293    | 0.0061 | 0.0273 |
| Meta-GCMC | 0.0441    | 0.0328    | 0.0086 | 0.0119 |
| GCMC+     | 0.0513    | 0.0448    | 0.1112 | 0.0394 |
| NGCF      | 0.0336    | 0.0093    | 0.0288 | 0.0177 |
| Basic-NGCF| 0.0390    | 0.0244    | 0.0971 | 0.0253 |
| Meta-NGCF | 0.0480    | 0.0382    | 0.1172 | 0.0319 |
| NGCF+     | 0.0486    | 0.0413    | 0.1256 | 0.0342 |
| LightGCN  | 0.0116    | 0.0213    | 0.1036 | 0.0183 |
| LightGCN+ | 0.0231    | 0.0426    | 0.1634 | 0.0353 |

Table 5: Recommendation performance, training time per epoch and convergent epochs w/wo meta-learning setting.

| Dataset   | Methods   | Recall@20 | NDCG@20 | Time Epoch | Epoch | Convergent Epoch |
|-----------|-----------|-----------|---------|------------|------|------------------|
| Weeplaces | GCMC      | 0.0112    | 0.0083  | 188.8s     | 33   | 0.0754           |
|           | CL-GCMC   | 0.0509    | 0.0453  | 721.3s     | 30   | 0.1021           |
|           | Basic-GCMC| 0.0513    | 0.0448  | 499.6s     | 12   | 0.1122           |
|           | Meta-GCMC | 0.0438    | 0.0413  | 141.8s     | 8    | 0.1121           |
|           | GCMC+     | 0.0513    | 0.0448  | 182.4s     | 30   | 0.0288           |
|           | CL-NGCF   | 0.0465    | 0.0403  | 702.0s     | 20   | 0.1123           |
|           | Basic-NGCF| 0.0438    | 0.0413  | 499.6s     | 12   | 0.1122           |
|           | Meta-NGCF | 0.0438    | 0.0413  | 182.4s     | 30   | 0.0288           |
|           | NGCF      | 0.0465    | 0.0403  | 702.0s     | 20   | 0.1123           |
|           | NGCF+     | 0.0511    | 0.0402  | 683.6s     | 18   | 0.1435           |
|           | LightGCN  | 0.0523    | 0.0426  | 843.4s     | 10   | 0.1634           |

Self-supervised Graph Learning for Occasional Group Recommendation
Figure 4: Sensitive analysis of ground-truth embeddings.

Table 6: Recommendation performance, training time for each epoch and convergent epochs under the multi-task learning or pre-training paradigms.

| Dataset    | Weeplaces | CAMRa2011 |
|------------|-----------|-----------|
| Method     | Recall    | NDCG     | Time | Epoch | Recall | NDCG | Time | Epoch |
| GCMC      | 0.0511   | 0.0418   | 60.3s | 26    | 0.1112 | 0.0794 | 112.3s | 17   |
| NGCF-P    | 0.0411   | 0.0388   | 283.1s | 28    | 0.1182 | 0.0518 | 60.2s  | 28   |
| NGCF      | 0.0486   | 0.0411   | 307.1s | 8     | 0.1256 | 0.0342 | 100.8s | 8    |
| LightGCN-P | 0.0253   | 0.0246   | 431.4s | 10    | 0.1634 | 0.0553 | 99.3s  | 6    |
| LightGCN  | 0.0253   | 0.0246   | 431.4s | 10    | 0.1634 | 0.0553 | 99.3s  | 6    |

by different models. To this end, we use competitive baselines to learn the ground-truth embeddings as proposed in Section 4.1.3, and report the performance of NGCF* and LightGCN* in Figure 4. Notation NGCF*-AGREE denotes SGG is equipped with the ground-truth embeddings, which are obtained by AGREE. Other notations are defined in a similar way. The results show that all the models that equipped with different ground-truth embeddings achieve almost the same performance, which indicates SGG is not sensitive to the ground-truth embeddings, as the baselines are good enough to learn high-quality embeddings from the abundant interactions.

4.4.3 Multi-task Learning Vs Pre-training. Here we would like to answer the question in Section 3.4: Can the recommendation performance benefit from the pre-training or the multi-task learning paradigm? Towards this goal, we first pre-train the SSL tasks on $D_t$ to obtain the model parameters, use them to initialize SGG, and then fine-tune SGG on $D_v$ via optimizing the main task. We term this variant as SGG-P. We report the recommendation performance, the average training time in each epoch and the average convergent epoch in Table 6. Based on the results, we find that: (1) SGG-P performs worse than SGG, but still better than other baselines (cf. Table 3). As jointly training in SGG admits that the representations in the main and SSL tasks are mutually enhanced with each other, which is superior than only offer a better initialization for the GNNs in SGG-P. This finding is consistent with the observation in previous studies [36]. (2) Compared with SGG-P, SGG has faster convergence speed. Although SGG-P has smaller training time per epoch, it still has larger total training time than SGG. This verifies the multi-task learning paradigm can speed up the model convergence.

4.4.4 Hyper-parameter analysis. We move on to study different designs of the layer depth $L$ and the neighbor size $K$. We select LightGCN*, NGCF* and GCMC*, and report their performances under different layer depth $L$ in Figure 5(a) and Figure 5(b), under different neighbor size $K$ in Figure 5(c) and Figure 5(d). Results on CAMRa2011 and Douban show the same trend with Weeplaces which are omitted for space.

5 RELATED WORK

5.1 Group Recommendation

Existing works on group recommendation can be generally divided into two categories: score aggregation and profile aggregation.

Score Aggregation. This strategy pre-defines a scoring function to obtain the preference score of all members in a group on the target item. The scoring functions include average [2], least misery [1] and maximum satisfaction [3]. However, due to the static recommendation process of the predefined functions, these methods easily fall into local optimal solutions.

Profile Aggregation. This strategy aggregates the profiles of group members and feeds the fused group profile into individual recommendation models. Essentially, probabilistic generative models and deep learning based models are proposed to aggregate the group profile. The generative model first selects group members for a target group, and then generates items based on the selected members and their associated hidden topics [24, 38, 42]. The deep learning based model conducts attention mechanism to assign each user an attention weight, which denotes the influence of group member in deciding the group’s choice on the target item [4, 5, 39]. However, both of these two methods suffer from the data sparsity issue. Recently, researchers propose GNN-based recommendation models, which incorporate high-order collaborative signals in the built graph [13, 19, 34] or hypergraph [12, 41]. Moreover, Zhang et al. [44] propose hypergraph convolution network (HHGR) with self-supervised node dropout strategy to alleviate the data sparsity issue. However, the GNNs still can not deal with the cold-start neighbors when performing graph convolution, and do not explicitly capture the correlations between the group and non-group members. Motivated by the SSL technique, we propose to jointly reconstruct the group/user/item embeddings under the meta-learning setting, and
further incorporate a meta aggregator and a CL adapter to improve the embedding quality.

6 CONCLUSION

We design a new self-supervised graph learning paradigm for group recommendation, which trains the GNN model to reconstruct the group embedding under the meta-learning setting. To deal with the cold-start neighbors during the graph convolution process, we further introduce a meta aggregator to enhance the aggregation ability of each graph convolution step. To explicitly consider the correlations between the group and non-group members, we further propose a CL adapter to regularize the group and user embeddings. Experimental results demonstrate the superiority of our proposed model against the state-of-the-art group recommendation models.

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, 1 (2009), 619–630.

[2] Linas Baltrunas, Tadas Mačiukinas, and Francesco Ricci. 2010. Group Recommendations with Rank Aggregation and Collaborative Filtering. In RecSys ’10: 119–126.

[3] Ludovico Boratto and Salvatore 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. Vol. 324. 1–20.

[4] Da Cao, Xiangnan He, Lianhai Miao, Yahui An, Chao Yang, and Richang Hong. 2018. Attentive Group Recommendation. In SIGIR ’18. ACM, 645–654.

[5] Da Cao, Xiangnan He, Lianhai Miao, Guangyi Xiao, Hao Chen, and Jia Xu. 2021. Social-Enhanced Attentive Group Recommendation. IEEE Trans. Knowl. Data Eng. 33, 3 (2021), 1195–1209.

[6] Jie Chen, Tengfei Ma, and Tao Xiao. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In ICLR ’18. OpenReview.net.

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In ICML 20 (Proceedings of Machine Learning Research, Vol. 119). PMLR, 1587–1607.

[8] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of Recommendation Algorithms. In RecSys ’10. ACM, 649–658.

[9] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In NeurIPS ’17. Curran Associates, Inc., 1024–1034.

[10] Xingjie Liu, Yuan Tian, Mao Ye, and Wang-Chien Lee. 2021. Exploring personal and Attentive Multi-view Embeddings for Occasional Group Recommendation. In CIKM ’21. ACM, 674–683.

[11] Ziniu Hu, Ting Chen, Kai-Wei Chang, and Yizhou Sun. 2019. Few-Shot Representation Learning for Out-Of-Vocabulary Words. In ACL ’19. 2497–2511.

[12] Ziniu Hu, Xuyiao Dong, Kuansan Wang, and Jia-Wei Chang, and Yizhou Sun. 2020. GPT-GNN: Generative Pre-Training of Graph Neural Networks. In SIGKDD ’20. ACM, 1857–1867.

[13] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In ICLR ’17. OpenReview.net.

[14] Chengzhao Liu, Xian Wang, Tao Lu, Wenwu Zhu, Junling Sun, and Steven C. H. Hsiao. 2019. Discrite Social Recommendation. In AAAI ’19. AAAI Press, 208–215.

[15] Xingjie Liu, Yuan Tian, Mao Ye, and Wang-Chien Lee. 2012. Exploring personal impact for group recommendation. In CIKM ’12. ACM, 674–683.

[16] Toon De Passemer, Simon Dooms, and Luc Martens. 2014. Comparison of group recommendation algorithms. Multim. Tools Appl. 72, 3 (2014), 2497–2511.

[17] Qinyong Wang, Jundong Li, and Hongzhi Yin. 2021. Self-Supervised Graph Learning for Occasional Group Recommendation. In CIKM’21. ACM.

[18] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiaodong Zhang. 2021. Semi-Supervised Multi-Channel Hypergraph Convolutional Network for Social Recommendation. In WWW ’21.

[19] Quan Yuan, Gao Cong, and Chin-Yew Lin. 2014. COM: a generative model for recommendations with Rank Aggregation and Collaborative Filtering. In RecSys ’10. ACM, 639–648.

[20] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiaodong Zhang. 2021. Self-Supervised Multi-Channel Hypergraph Convolutional Network for Group Recommendation. In CIKM’21. ACM.

[21] Jeffrey Ng, Ashish Relan, and Ashish Sabharwal. 2019. Leveraging Implicit Feedback: A Deep Feedforward Neural Network. In NeurIPS ’19. Curran Associates, Inc., 3391–3401.
Figure 7: Recommendation performance under different neighbor size $K$, $c\% = 0.1$ and $L=3$.

Figure 8: Recommendation performance under different balancing parameter $\lambda_1$, $K=5$, $c\% = 0.1$ and $L=3$.

Figure 6: Recommendation performance under different layer $L$, $c\% = 0.1$ and $K=5$.

A APPENDIX

A.1 Interacted Number and Sparse Rate Analysis

We report the recommendation performance under different interacted number and sparse rate on CAMRa2011 and Douban in Figure 9. The results are consistent with the findings in Section 4.2.2.

A.2 Hyper-parameter analysis.

- In terms of the balancing parameter $\lambda_1$, we report the recommendation performance of LightGCN*, NGCF* and GCMC* on CAMRa2011 and Douban in Figure 8. The results show the performance first increases and then drops when increasing $\lambda_1$ from 0.01 to 1.2. The peak point is 1 at all cases. This indicates the auxiliary SSL tasks are as important as the main recommendation task.

- In terms of the layer depth $L$, we report the recommendation performance of LightGCN*, NGCF* and GCMC* on CAMRa2011 and Douban in Figure 6. The results are consistent with the findings of Section 4.4.4.

- In terms of the neighbor size $K$, we report the recommendation performance of LightGCN*, NGCF* and GCMC* on CAMRa2011.
and Douban in Figure 7. The results are consistent with the findings of Section 4.4.4.