2019 IEEE 35th International Conference on Data Engineering (ICDE)

Social Influence-based Group Representation Learning for Group Recommendation

Hongzhi Yin[†] Qinyong Wang[†] Kai Zheng[§] Zhixu Li[‡] Jiali Yang[‡] Xiaofang Zhou[†]
[†]*The University of Queensland, School of Information Technology and Electrical Engineering*
[§]*School of Computer Science and Engineering, University of Electronic Science and Technology of China*
[‡]*School of Computer Science and Technology, Soochow University*
[†]{h.yin1, qinyong.wang, zxf}@uq.edu.au [§]zhengkai@uestc.edu.cn
[‡]{zhixuli,jialiyang}@suda.edu.cn

***Abstract*—As social animals, attending group activities is an indispensable part in people's daily social life, and it is an important task for recommender systems to suggest satisfying activities to a group of users. The major challenge in this task is how to aggregate personal preferences of group members to infer the decision of a group. Conventional group recommendation methods applied a predefined strategy for preference aggregation. However, these static strategies are too simple to model the real and complex process of group decision-making, especially for occasional groups which are formed ad-hoc. Moreover, group members should have non-uniform influences or weights in a group, and the weight of a user can be varied in different groups. Therefore, an ideal group recommender system should be able to accurately learn not only users' personal preferences, but also the preference aggregation strategy from data.**

In this paper, we propose a novel group recommender system, namely SIGR (short for "Social Influence-based Group Recommender"), which takes an attention mechanism and a bipartite graph embedding model BGEM as building blocks. Specifically, we adopt an attention mechanism to learn each user's social influence and adapt their social influences to different groups, and develop a novel deep social influence learning framework to exploit and integrate users' global and local social network structure information to further improve the estimation of users' social influences. BGEM is extended to model group-item interactions. In order to overcome the limitation and sparsity of the interaction data generated by occasional groups, we propose two model optimization approaches to seamlessly integrate the user-item interaction data. We create two large-scale benchmark datasets and conduct extensive experiments on them. The experimental results show the superiority of our proposed SIGR by comparing with state-of-the-art group recommender models.

I. Introduction

As social animals, group activities are essential for people's social life. For example, families often watch TV programs together at night; friends often dine out, watch movies, attend parties and travel together. With the recent development and prevalence of smart phones and social networking services (e.g., Meetup and Facebook Events), it is becoming more convenient and easier for people to get together to form a persistent or occasional group. It is highly demanded to develop group recommender systems to suggest relevant items/events (e.g., dining out, movie watching and parties) for a group of users, known as group recommendation.

The first group recommender system MusicFX [17] was developed to recommend music to a group of gym users. Since then, group recommendation has been seen in various recommendation applications, such as tourism [18] and social events [16]. There are two types of groups: persistent and occasional groups [22], [30]. Persistent groups refer to relatively static groups with stable members and sufficient group-item interaction records [7], [15], [25], [27], such as interest-oriented groups in Meetup; while occasional groups are formed ad-hoc and users may just constitute the groups for the first time (i.e., cold-start groups) [4], [16], [36].

To make recommendations to persistent groups, each group can be treated as a virtual user and the personalized recommendation algorithms developed for individual users can be straightforwardly employed since there are sufficient persistent group-item interaction records. However, for occasional groups, their historical interaction data is extremely sparse and even unavailable. Thus, it is infeasible to directly learn the preference representation of an occasional group, and we can only learn the preferences of an occasional group by aggregating the personal preferences of its members. In this paper, we focus on a more general scenario of group recommendation, i.e., making recommendations to occasional groups, as the recommendation techniques developed for occasional groups can also be applied to persistent groups.

Group recommendation is much more challenging than making recommendations to individual users, as different group members may have different preferences. A good group recommendation system should be able to not only accurately learn users' personal preferences, but also model how a decision or consensus among group members is reached. Prior studies [2], [28] on group recommendation systems have been focused on exploring various heuristic aggregation strategies (e.g., average, least misery and maximum pleasure) to find a consensus among group members on an item. However, all these heuristic and predefined aggregation strategies are too simple to model the real and complex process of group decision-making, leading to suboptimal group recommendation performance. Moreover, a user may exhibit different influences and have different weights in different groups.

In this paper, we focus on the essential problem in group recommendation – preference aggregation, that is how to aggregate personal preferences of group members to decide a group's choice on items. Rather than exploring new heuristic

2375-026X/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDE.2019.00057

and predefined strategies, we first introduce the notion of personal social influence to quantify and differentiate the contributions of group members to a group decision, and then propose to automatically learn the social influence-based aggregation strategy from the group-item interaction data. The key challenges are how to learn the social influence of each user, and how to adapt their influences to different groups. Inspired by the recent advancement of representation learning, we propose to overcome the challenges by learning group representation in the embedding space. To obtain an embedding vector that represents the preferences of a group, we aggregate the embedding of each group member in a learnable way. Specifically, we introduce a latent variable to represent a user's global social influence that is independent from specific groups, and then adopt the attention mechanism [3] to adapt the global social influence to different groups, which is capable of assigning different weights for a user in different groups without adding additional parameters. In this way, we can dynamically adjust the aggregation strategy for a group to capture the complex group decision-making process.

As the number of group activities a user attends is rather limited, it is difficult to accurately learn each user's social influence without over-fitting, let alone the users who have never attended any group activity before. To alleviate the data sparsity issue, we propose a novel deep social influence learning framework based on stacked denoising auto-encoders (SDAE) [29] to exploit and integrate the available user social network information to improve the estimation of user social influence in the form of regularization. Both global and local network structure information are considered. Specifically, we adopt various centrality measures (e.g., PageRank centrality, degree centrality and eigenvector centrality) to capture the global structure features, and apply the recent network embedding approaches [8] such as DeepWalk [20] and node2vec [10] to learn the local structure features of each user node by preserving the neighborhood information of each node.

We employ the Bipartite Graph Embedding Model (BGEM) [35] to learn the embedding of users and items in a low-dimensional space from the group-item interaction data. However, from the group-item interaction data, we can only learn embedding of users who have attended at least one group activity and items that interact with at least one group. Due to the cold-start nature of occasional groups, they may contain members who have never attended group activities before. Besides, the number of group activities that a user attends is also rather limited. To effectively overcome these limitations, we propose to leverage the sufficient individuals' activity data (i.e., user-item interaction data) to improve the embedding learning of both users and items. To seamlessly integrate user-item interaction data with group-item interaction data in the same embedding space, we propose two model optimization approaches to implement our Social Influence-based Group Recommender (SIGR): a two-stage optimization approach and a joint optimization approach.

Specifically, the two-stage approach first learns embedding of users and items from the user-item interaction data using BGEM in the first stage, which is then utilized to initialize the user/item embedding in the second stage. We will update the user/item embedding and learn the user social influence from both group-item interaction data and social network data in the second stage. The joint approach simultaneously learns the user/item embedding from both user-item interaction data and group-item interaction data, and the user social influence is also learned in this process. The key difference between these two approaches is that there are two objective functions to optimize in the two-stage approach, while there is only one unified objective function in the joint approach.

The main contributions of this paper are summarized below.

- To the best of our knowledge, we are the first to integrate the attention mechanism with the bipartite graph embedding technique for group recommendation. Specially, we propose a novel group recommender model (SIGR) based on these techniques to learn both user embedding and user social influences from data in a unified way, enabling the representation learning for occasional groups.
- We propose two model optimization approaches to leverage the user-item interaction data to overcome the limitations and alleviate the sparsity of the group-item interaction data, in which both novel positive sampling approach and negative sampling strategy are developed to advance the conventional stochastic gradient descent algorithm.
- To overcome the data sparsity and avoid the over-fitting in the estimation of personal social influence, we develop a novel deep social influence learning framework to exploit and integrate both global and local social network structure features to improve the group recommendation.
- We create two large-scale benchmark datasets for evaluating group recommendation systems, especially the recommenders that are able to make recommendations to occasional groups. Extensive experiments are conducted to evaluate the performance of our proposed SIGR, and the experimental results show its superiority by comparing with the state-of-the-art techniques.

II. Preliminaries

Generally speaking, our proposed SIGR model consists of two components: 1) interaction learning with BGEM for both group-item interaction data and user-item interaction data; and 2) social influence-based group representation learning that learns the group's preferences based on the preference aggregation of its members. We first present the notations and then formulate the group recommendation problem in this section. We then introduce the two key ingredients of our proposed SIGR in Section III and VI respectively. Section V discusses the model optimization approaches.

A. Notations and Problem Formulation

Following the convention, we use bold capital letters (e.g., $\mathbf{X}$) to represent both matrices and graphs, and use squiggle capital letters (e.g., $\mathcal{X}$) to denote sets. We use lowercase letters with superscript $\vec{}$ (e.g., $\vec{x}$) to denote vectors. All vectors are in column forms if not clarified. We employ normal lowercase letters (e.g., x) to denote scalars.

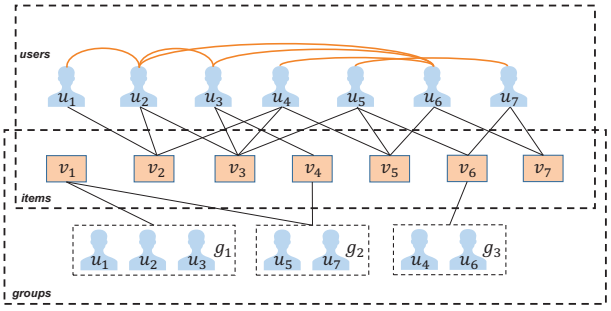


Fig. 1: Illustration of the input data for the task of making recommendations to occasional groups, including user-item interactions, group-item interactions and user-user interactions.

We assume that there are a set of users $\mathcal{U}$, a set of groups $\mathcal{G}$ and a set of items $\mathcal{V}$ in the group recommender system. The m-th group $g_m \in \mathcal{G}$ consists of a set of users, and we use $\mathcal{G}_m$ to denote this set of users. There are three kinds of observed interaction data among $\mathcal{U}$, $\mathcal{G}$ and $\mathcal{V}$: user-item interactions, group-item interactions and user-user interactions. We use bipartite graphs $\mathbf{G}_{UV}$ and $\mathbf{G}_{GV}$ to represent user-item interactions and group-item interactions respectively, and use a general graph to denote user-user interactions (i.e., user social network). Figure 1 illustrates the input data of our group recommendation task. Then, given an occasional group g_m, our task is to recommend a ranked list of items that group g_m may be interested in, which is formally defined as follows.

Input: A set of users $\mathcal{U}$, a set of groups $\mathcal{G}$, a set of items $\mathcal{V}$, group-item interactions $\mathbf{G}_{GV}$, user-item interactions $\mathbf{G}_{UV}$ and user-user interactions $\mathbf{G}_S$.

Output: A personalized ranking function that maps an item to a ranking score for a target occasional group $f_g : \mathcal{V} \longrightarrow \mathbb{R}$.

III. Interaction Representation and Learning

In this section, we take user-item interaction data as an example to show how to employ the BGEM model [35] to learn the embedding of heterogenous interaction entities. We choose BGEM to perform interaction learning because of its success in bipartite graph embedding [35], and its superiority has also been validated in our Experiment 1.

We use a bipartite graph to represent the interactions between users and items, i.e., $\mathbf{G}_{UV} = (\mathcal{U} \cup \mathcal{V}, \mathcal{E}_{UV})$, where $\mathcal{E}_{UV}$ is the set of edges between users and items. If user u_i interacts with item v_j, there is an edge e_{ij} between them. If the user's rating information is available, the weight on the edge e_{ij} is equal to u_i's normalized rating for item v_j (i.e., w_{ij} is normalized in the range $[0, 1]$); otherwise, the weight is set to be one. Given a user u_i, we define the probability of u_i interacting with an item v_j as follows:

$$p(v_j|u_i) = \frac{\exp(\vec{u}_i \cdot \vec{v}_j)}{\sum_{v_{j'} \in \mathcal{V}} \exp(\vec{u}_i \cdot \vec{v}_{j'})} \tag{1}$$

where $\vec{u}_i$ is the embedding of user u_i that represents her/his preferences in the latent space, and $\vec{v}_j$ is the embedding of item v_j.

Following the recent word and network embedding techniques [19], BGEM tries to minimize the KL-divergence between the estimated neighbor probability distribution of each user $p(\cdot|u_i)$ and the empirical distribution $\hat{p}(\cdot|u_i)$. The empirical distribution is defined as $\hat{p}(v_j|u_i) = w_{ij}/d_i$, where w_{ij} is the weight on the edge e_{ij} and d_i is the outdegree of user node u_i, i.e., $d_i = \sum_{v_j \in \mathcal{V}} w_{ij}$. By omitting some constants, we obtain the following objective function:

$$O_{UV} = -\sum_{e_{ij} \in \mathcal{E}_{UV}} w_{ij} \log p(v_j|u_i). \tag{2}$$

By minimizing the above objective function, we are able to learn each user's embedding $\vec{u}_i$ and each item's embedding $\vec{v}_j$ in a d-dimensional latent space.

However, directly optimizing the above objective function is computationally expensive because it requires traversing the entire set of items when computing the conditional probability $p(v_j|u_i)$. To overcome this issue, BGEM adopts the approach of negative sampling technique proposed in [19], which samples multiple negative items to form corrupted examples (or negative edges) according to some noise distribution for each positive example (u_i, v_j). More precisely, it specifies the following objective function for each positive example (u_i, v_j) (i.e.,each observed edge e_{ij}):

$$\log \sigma(\vec{u}_i \cdot \vec{v}_j) + \sum_{k=1}^{M} E_{v_k \sim P_n(v)}[\log \sigma(-\vec{u}_i \cdot \vec{v}_k)], \tag{3}$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function. The first term maximizes the probability of the observed positive example, and the second term minimizes the probability of M corrupted examples drawn from a noise distribution $P_n(v)$. $E_{v_k \sim P_n(v)}[\log \sigma(-\vec{u}_i \cdot \vec{v}_k)]$ is the expected value of $\log \sigma(-\vec{u}_i \cdot \vec{v}_k)$. We set the noise distribution $P_n(v) \propto d_v^{0.75}$, following [19], where d_v is the out-degree of item node v. The natural interpretation of an arbitrary missing interaction is that the user does not know about the existence of the item and thus there is no interaction. It is also possible that the user does know about the item and chooses not to interact with it, because she dislikes it. The more popular an item is, the more probable it is that the user knows about the item, thus it is more likely that a missing interaction expresses "dislike". Therefore we should sample items in proportion of their popularity.

We substitute $\log p(v_j|u_i)$ with Formula 3 in the objective function O_{UV}, and the objective can be reformulated as follows:

$$\begin{aligned} O_{UV} = -\sum_{e_{ij} \in \mathcal{E}_{UV}} w_{ij} \big(& \log \sigma(\vec{u}_i \cdot \vec{v}_j) \\ & + \sum_{k=1}^{M} E_{v_k \sim P_n(v)}[\log \sigma(-\vec{u}_i \cdot \vec{v}_k)] \big). \end{aligned} \tag{4}$$

Thus, the task is equivalent to distinguish positive examples (u_i, v_j) from the corresponding corrupted examples, using the logistic regression method. A straightforward way to optimize O_{UV} is using the stochastic gradient descent (SGD). To speed up the training process, we adopt the asynchronous stochastic gradient descent algorithm (ASGD) [23] for model

optimization, which guarantees the scalability and efficiency of our model in practice. In each step, the ASGD algorithm samples a positive example (u_i, v_j) and M negative examples (u_i, v_k) to update model parameters, and the gradients w.r.t. $\vec{u}_i$, $\vec{v}_j$ and $\vec{v}_k$ are calculated as follows:

$$\frac{\partial O_{UV}}{\partial \vec{u}_i} = w_{ij}\big(\sum_{k=1}^{M} E_{v_k \sim P_n(v)}[\sigma(\vec{u}_i \cdot \vec{v}_k)\vec{v}_k] - \sigma(-\vec{u}_i \cdot \vec{v}_j)\vec{v}_j\big), \quad (5)$$

$$\frac{\partial O_{UV}}{\partial \vec{v}_j} = -w_{ij}\sigma(-\vec{u}_i \cdot \vec{v}_j)\vec{u}_i, \quad (6)$$

$$\frac{\partial O_{UV}}{\partial \vec{v}_k} = w_{ij}\sigma(\vec{u}_i \cdot \vec{v}_k)\vec{u}_i. \quad (7)$$

IV. Group Representation Learning

Similarly, the interactions between groups and items can also be represented by a bipartite graph $\mathbf{G}_{GV} = (\mathcal{G} \cup \mathcal{V}, \mathcal{E}_{GV})$ where $\mathcal{G}$ is a set of groups and $\mathcal{E}_{GV}$ is a collection of edges between groups and items. If group g_m interacts with item v_j, there will be an edge e_{mj} between them. As the rating information of an occasional group is rarely available, we simply set the weight on the edge e_{mj} to be one. Given a group g_m, we define the probability of g_m interacting with an item v_j as follows:

$$p(v_j|g_m) = \frac{\exp(\vec{g}_m \cdot \vec{v}_j)}{\sum_{v_{j'} \in \mathcal{V}} \exp(\vec{g}_m \cdot \vec{v}_{j'})}, \quad (8)$$

where $\vec{g}_m$ is the embedding of group g_m in the latent space. Following the interaction learning presented in Section III, we try to minimize the KL-divergence between the estimated neighbor probability distribution of each group $p(\cdot|g_m)$ and the empirical distribution $\hat{p}(\cdot|g_m)$. By omitting some constants, we get the following objective function:

$$O_{GV} = -\sum_{e_{mj} \in \mathcal{E}_{GV}} w_{mj} \log p(v_j|g_m). \quad (9)$$

By adopting the negative sampling technique, the above objective function is reformulated as follows:

$$\begin{aligned} O_{GV} = -\sum_{e_{mj} \in \mathcal{E}_{GV}} w_{mj}\big(&\log \sigma(\vec{g}_m \cdot \vec{v}_j) \\ &+ \sum_{k=1}^{M} E_{v_k \sim P_n(v)}[\log \sigma(-\vec{g}_m \cdot \vec{v}_k)]\big). \end{aligned} \quad (10)$$

However, we are not able to directly learn the embedding of an occasional group $\vec{g}_m$ from the group-item interaction data due to the cold-start nature of occasional groups. In contrast to persistent groups, an occasional group is defined as a number of persons who do something occasionally together, like having a dinner, watching a movie, attending a party and visiting a POI [5]. Its members have a common aim only in a particular moment. There are many contexts where a group of persons is not established for some shared long-term interests, but might be occasionally interested in getting together for a common aim, e.g., people attending events together or traveling together. As occasional groups are typically short-lived by definition and many new occasional groups are being created, they often have little or no historical interaction data. The group-item interaction matrix is much sparser than user-item matrix (referring to relevant statistics of two real-life datasets in Table I). The problem of cold-start groups arises naturally, and the classic group recommendation techniques [7], [15] that assume groups have ample historical interaction records would significantly underperform in this scenario.

To address the cold-start problem, we propose to learn a group's embedding by aggregating the embedding of its members. Specifically, given an occasional group g_m, its embedding is represented as follows:

$$\vec{g}_m = \sum_{u_i \in \mathcal{G}_m} \lambda_{im} \vec{u}_i, \quad (11)$$

where $\mathcal{G}_m$ represents the set of users who constitute group g_m. As group members have different social statuses, expertise, reputation, personality and other social factors [1], [11], they are not equal and have different social influences in the group's decisions and choices. Thus, we use λ_{im} to denote u_i's social influence/weight in group g_m, and it also reflects how much u_i contributes to the group's decision-making. A user can exhibit different social influences in different groups that consist of different members. An important aspect of group activities is the need to reach consensus. In non-virtual environments, consensus results from negotiation among group members, especially those group members with low social influences are often willing to modify their initial individual opinions and compromise to satisfy the preferences of the influential members. Sometimes, a group's preferences reflect the preferences of a few influential members (e.g., group leaders or opinion leaders) rather than the common preferences of most group members.

A. Attention-based Social Influence Learning

How to learn the social influence λ_{im} of each member u_i in each group g_m from the group-item interaction data? As occasional groups have few historical interactions on items, it is impossible to directly learn the group-aware personal social influence λ_{im}. In light of this, we introduce a non-negative latent variable γ_i to represent the global social influence of user u_i, which is independent from specific groups. Then, the group-aware personal social influence λ_{im} is computed as follows, inspired by the attention mechanism [3].

$$\lambda_{im} = \frac{\exp(\gamma_i)}{\sum_{u_j \in \mathcal{G}_m} \exp(\gamma_j)}. \quad (12)$$

where $\exp(\gamma_i)$ reflects the relative importance to influence a group decision. Thus, we only need to learn a global personal social influence γ_i for each user u_i rater than a large number of group-aware personal social influence λ_{im} from the group-item interaction data.

However, the group-item interaction data is extremely sparse and the number of group activities a user attends is also rather limited, hence even learning global social influence may also suffer from over-fitting problems. Specifically, if a user u_i has attended only very few group activities, his/her estimated global social influence may not be reliable or accurate. Moreover, for users who have never participated in any group, we cannot

learn their social influences from the group-item interaction data. Inspired by [11], [16], we exploit and integrate the user social network information that may provide strong signals about users' global social influences. Specifically, we explore and exploit both global and local network structure information. We adopt various centrality measures (e.g., PageRank centrality, degree centrality, closeness centrality, betweenness centrality and eigenvector centrality) to capture the global structure features, and apply the recent network embedding approaches such as DeepWalk [20] and node2vec [10] to obtain the local structure features of each user node by preserving the neighborhood information of each node.

How to integrate the social network features into the social influence learning? We use $\vec{x}_i$ to denote the social network feature vector of user u_i, and also introduce a context vector $\vec{\omega}$ as a feature selector to assign different weights to different social network features. Then, we take dot product between the social network feature vector $\vec{x}_i$ and the context vector $\vec{\omega}$ as the Gaussian prior for the personal global social influence variable, i.e., $\gamma_i \sim \mathcal{N}(\vec{x}_i \cdot \vec{\omega} + b, \rho_S^2)$, where b is a global bias term. Rather than simply defining the personal global social influence γ_i being equal to the weighted sum of the social network features, we assume that γ_i follows a normal distribution with the mean $\vec{x}_i \cdot \vec{\omega} + b$, to learn a more robust personal global social influence, since personal global social influences may also be affected by other unknown factors.

As we introduce a Gaussian prior for the personal social influence parameter γ_i, a corresponding regularization term should be added to our objective function, and our new objective function is defined as follows.

$$O_{SGV} = O_{GV} + R_S, \tag{13}$$

where $R_S = \frac{1}{2\rho_S^2} \sum_{u_i \in \mathcal{U}} (\gamma_i - (\vec{x}_i \cdot \vec{\omega} + b))^2$, and the variance ρ_S^2 actually plays the role of controlling the weight of the regularization term in the new objective function. Note that all global social network feature values are normalized into the same range $[0, 1]$, and the local social network features (i.e., user node embedding) are normalized into the range $[-1, 1]$.

Note that by integrating the social network features, we are able to learn the global social influences of users who have never attended any group activity before, which effectively overcomes the limitation of the group-item interaction data.

B. Deep Social Influence Learning

However, the above linear approach for exploiting social network features ignores the correlation among social network features and also fails to capture the non-linear and complex inherent structure of social network features. It is crucial to account for the interactions between features, especially in a non-linear way. Inspired by the success of deep neural networks in learning non-linear feature interactions, we investigate how to apply an unsupervised deep learning model called stacked denoising auto-encoders (SDAE) to obtain a latent social network feature vector $\vec{h}_i$ from the raw/origial feature vector $\vec{x}_i$. SDAE [29] is a feedback neural network for learning the representation of the corrupted input data

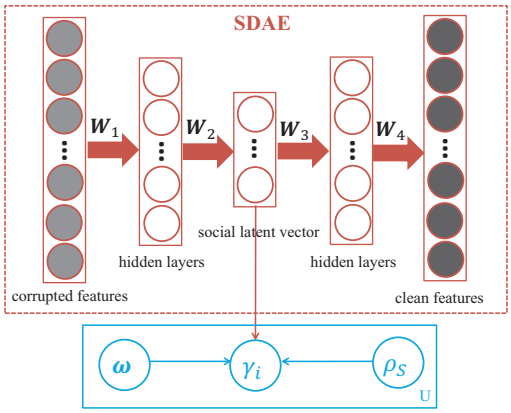


Fig. 2: **Deep Social Influence Learning from Social Network Features**.

by learning to predict the clean itself in the output. Before presenting the model detail, we first introduce the notations used in SDAE. We use L to denote the number of network layers, and use matrix $\mathbf{X}_l$ to represent the output of layer l in SDAE. Note that we use the last layer's output $\mathbf{X}_L$ to represent the original social network features of all users, where the i-th row is the original social network feature vector of user i (i.e., $\vec{x}_i$). Similarly, we use matrix $\mathbf{X}_0$ to represent the noise-corrupted matrix (randomly masking some entries of $\mathbf{X}_L$ by marking them zero). $\mathbf{W}_l$ and $\mathbf{b}_l$ are weight parameter matrix and bias parameter vector, respectively, for layer l.

Figure 2 illustrates a 4-layer SDAE. As shown in this figure, the first $L/2$ layers of the network (from $\mathbf{X}_0$ to $\mathbf{X}_2$) usually act as an encoder part, which maps the corrupted input $\mathbf{X}_0$ to a latent compact representation $\mathbf{X}_2$, and the last $L/2$ layers (from $\mathbf{X}_2$ to $\mathbf{X}_4$) usually act as the decoder part, which recovers the original input $\mathbf{X}_4$ from the latent representation $\mathbf{X}_2$. Given that both the original input $\mathbf{X}_L$ and the corrupted input $\mathbf{X}_0$ are observed, we present the generative process of each layer l in SDAE as follows:

1) For weight parameter $\mathbf{W}_l$, draw $\mathbf{W}_l \sim \mathcal{N}(\mathbf{0}, \rho_W^2 \mathbf{I})$.
2) For bias parameter, draw $\mathbf{b}_l \sim \mathcal{N}(\mathbf{0}, \rho_b^2 \mathbf{I})$.
3) For the output of each layer, draw $\mathbf{X}_l \sim \mathcal{N}(\sigma(\mathbf{X}_{l-1}\mathbf{W}_l + \mathbf{b}_l), \rho_X^2 \mathbf{I})$.

We take the output of the $L/2$-th layer as the latent representation of social network features, i.e., $\vec{h}_i$ is equal to the i-th row of $\mathbf{X}_{\frac{L}{2}}$. For each user i, his/her global social influence γ_i is generated as: $\gamma_i \sim \mathcal{N}(\vec{h}_i \cdot \vec{\omega} + b, \rho_S^2)$.

By integrating SDAE model into our group representation learning framework, we get the following joint objective function:

$$O_{SGV} = O_{GV} + O_{SDAE} + R_S, \tag{14}$$

where O_{SDAE} is defined as follows.

$$O_{SDAE} = \frac{1}{2} \sum_l \left(\frac{\|\sigma(\mathbf{X}_{l-1}\mathbf{W}_l + \mathbf{b}_l) - \mathbf{X}_l\|_2^2}{\rho_X^2} + \frac{\|\mathbf{W}_l\|_2^2}{\rho_W^2} + \frac{\|\mathbf{b}_l\|_2^2}{\rho_b^2} \right). \tag{15}$$

V. Model Optimization

By optimizing the above objective function O_{SGV} in Equation (14), we are only able to obtain the embedding of users

who have attended at least one group activity and embedding of items that interact with at least one group. However, due to the cold-start nature of occasional groups, they may contain members who have never attended any group activity before. Besides, the group-item interaction matrix is very sparse, thus the embedding of both users and items learned by optimizing O_{SGV} is not accurate or reliable. To effectively overcome the limitations and sparsity issue of the group-item interaction data, we propose to leverage the sufficient user-item interaction data. Technically, we develop two model optimization approaches to integrate O_{UV} with O_{SGV}: Two-stage Training and Joint Training.

Two-stage Training. We first optimize the objective function O_{UV} to obtain the embedding of users and items ($\vec{u}_i$ and $\vec{v}_j$) in the first stage. In the second stage, these learned embedding are taken as the initial values of the embedding of users and items in O_{SGV}, and then they will be also fine-tuned and updated during the process of optimizing the objective O_{SGV}. Specifically, we adopt the SGD algorithm to optimize O_{SGV}. In each gradient step, we randomly sample a positive example (g_m, v_j) and M negative examples (g_m, v_k) to update model parameters. The gradients w.r.t the model parameters $\mathbf{W}_l$, $\mathbf{b}_l$, γ_i and $\vec{\omega}$, are updated as follows:

$$\frac{\partial O_{SGV}}{\partial \mathbf{W}_l} = \frac{\partial O_{SDAE}}{\partial \mathbf{W}_l}, \qquad \frac{\partial O_{SGV}}{\partial \mathbf{b}_l} = \frac{\partial O_{SDAE}}{\partial \mathbf{b}_l}, \tag{16}$$

$$\frac{\partial O_{SGV}}{\partial \gamma_i} = \frac{\partial O_{GV}}{\partial \vec{g}_m}\frac{\partial \vec{g}_m}{\partial \gamma_i} + \frac{R_S}{\gamma_i}, \tag{17}$$

$$\frac{\partial O_{SGV}}{\partial \vec{\omega}} = \frac{\partial R_S}{\partial \vec{\omega}} = -\frac{1}{\rho_S^2}\sum_{u_j \in \mathcal{G}_m}\big(\gamma_j - (\vec{h}_j \cdot \vec{\omega} + b)\big)\vec{h}_j, \tag{18}$$

where the detailed gradient computation formulas for $\mathbf{W}_l$ and $\mathbf{b}_l$ in SDAE model can be found in [37]. $\frac{\partial O_{GV}}{\partial \vec{g}_m}$, $\frac{\partial \vec{g}_m}{\partial \gamma_i}$ and $\frac{R_S}{\gamma_i}$ are computed, as follows.

$$\frac{\partial O_{GV}}{\partial \vec{g}_m} = w_{mj}\big(\sum_{k=1}^{M} E_{v_k \sim P_n(v)}[\sigma(\vec{g}_m \cdot \vec{v}_k)\vec{v}_k] - \sigma(-\vec{g}_m \cdot \vec{v}_j)\vec{v}_j\big), \tag{19}$$

$$\frac{\partial \vec{g}_m}{\partial \gamma_i} = exp(\gamma_i)\frac{\vec{u}_i \sum_{u_j \in \mathcal{G}_m}\exp(\gamma_j) - \sum_{u_j \in \mathcal{G}_m}\exp(\gamma_j)\vec{u}_j}{(\sum_{u_j \in \mathcal{G}_m}\exp(\gamma_j))^2}, \tag{20}$$

$$\frac{\partial R_S}{\partial \gamma_i} = \frac{1}{\rho_S^2}(\gamma_i - (\vec{h}_i \cdot \vec{\omega} + b)). \tag{21}$$

And the gradients w.r.t. $\vec{u}_i$, $\vec{v}_j$ and $\vec{v}_k$ are updated as follows.

$$\frac{\partial O_{SGV}}{\partial \vec{u}_i} = \frac{\partial O_{GV}}{\partial \vec{u}_i} = w_{mj}\lambda_{im}\big(\sum_{k=1}^{M} E_{v_k \sim P_n(v)}[\sigma(\vec{g}_m \cdot \vec{v}_k)\vec{v}_k] - \sigma(-\vec{g}_m \cdot \vec{v}_j)\vec{v}_j\big), \tag{22}$$

$$\frac{\partial O_{SGV}}{\partial \vec{v}_j} = \frac{\partial O_{GV}}{\partial \vec{v}_j} = -w_{mj}\sigma(-\vec{g}_m \cdot \vec{v}_j)\vec{g}_m, \tag{23}$$

$$\frac{\partial O_{SGV}}{\partial \vec{v}_j} = \frac{\partial O_{UV}}{\partial \vec{v}_k} = w_{mj}\sigma(\vec{g}_m \cdot \vec{v}_k)\vec{g}_m. \tag{24}$$

Joint Training. By combining objectives O_{UV} and O_{SGV}, the joint objective function can be simply defined as follows:

$$O_{GUV} = O_{SGV} + O_{UV}. \tag{25}$$

To optimize the above joint objective function, we cannot straightforwardly use the Stochastic Gradient Descent algorithm (SGD), because O_{SGV} and O_{UV} in Equation 25 have different training instances: group-item pairs vs. user-item pairs. To address this issue, one possible solution is to first merge all edges in edge sets $\mathcal{E}_{UV}$ and $\mathcal{E}_{GV}$ into a big edge set, and then randomly sample a positive edge from the merged edge set in each gradient step, just as done in [31], [35]. However, the group-item interaction graph is much sparser than the user-item interaction graph, i.e., the number of edges in $\mathcal{E}_{GV}$ is much smaller than the number of edges in $\mathcal{E}_{UV}$. If we uniformly draw a positive edge from the merged edge set to perform stochastic gradient descent, most of sampled positive edges would be user-item edges, and there would not be enough group-item interaction edges for accurately estimating personal global social influences γ_i, feature weight vector $\vec{\omega}$ and other parameters in SDAE. To overcome the challenge of data skewness, we propose a novel joint training procedure in Algorithm 1. Instead of merging all edges into a big edge set, we will first draw or choose a bipartite graph with the sampling probabilities $\frac{1}{1+\eta}$ and $\frac{\eta}{1+\eta}$ for the group-item graph and user-item graph respectively, and then randomly draw a positive edge and M negative edges from the sampled bipartite graph to update the gradients. By doing so, the joint objective function is actually changed to the following equation:

Algorithm 1: Joint Training Algorithm

Input: $\mathbf{G}_{UV}$, $\mathbf{G}_{GV}$, number of positive samples N, number of negative samples per positive sample M;
Output: The parameter set $\Theta = \{\vec{u}_i, \vec{v}_j, \gamma_i, \vec{\omega}, \mathbf{W}_l, \mathbf{b}_l\}$

```
iter ← 0;
while iter ≤ N do
    Flip a coin c according to a bernoulli(1/(1+η));
    if c = 1 then
        Randomly draw a positive edge e_ij ∈ E_GV;
        Sample M negative edges for e_ij;
        Update the associated model parameters w.r.t.
        Equations (16, 17, 18, 22, 23, 24);
    end
    else
        Randomly draw a positive edge e_ij ∈ E_UV;
        Sample M negative edges for e_ij;
        Update the associated model parameters w.r.t.
        Equations (5, 6, 7);
    end
    iter=iter+1;
end
```

$$O_{GUV} = O_{SGV} + \eta O_{UV}, \tag{26}$$

where η is a non-negative hyper-parameter that is used to control the weight or contribution of the objective O_{UV}.

Time Complexity Analysis. The time complexity for each stochastic gradient step in Algorithm 1 is $O(d \cdot M) = O(d)$, where M is often small (less than 10) in large-scale datasets [35] and thus can be ignorable; d is the embedding dimension and typically smaller than 100. We assume that our model needs N samples (i.e., N stochastic gradient steps) to reach convergence, thus its overall time complexity is $O(d \cdot N)$. In practice, the required number of stochastic gradient steps N is typically proportional to the number of edges [35].

A. Negative Sampling of Items

How to sample M negative items to form M negative edges (i.e., corrupted examples) for each positive edge (i.e., each observed edge)? For a positive user-item edge (u_i, v_j) on $\mathbf{G}_{UV}$, we employ the widely adopted degree-based noise distribution $P_n(v_k) \propto d_{v_k}^{0.75}$ [19], where d_{v_k} is the out-degree of item node v_k on the user-item graph. However, this classic negative sampling method does not apply to the occasional group-item interaction graph $\mathbf{G}_{GV}$, because the group-item graph is extremely sparse and the variance of its node degrees is not so obvious. In this case, the degree-based negative sampling strategy may degrade to the uniform negative sampling. Most negative examples generated in this way are "too easy" and will contribute little to learning an effective discriminator, because they are obviously false. To generate more difficult and informative negative examples for each positive edge (g_m, v_j) on $\mathbf{G}_{GV}$, we propose a novel group-aware negative sampling technique by leveraging the user-item interaction graph. Specifically, given a group g_m, the noise distribution is changed to be group-aware, i.e., $P_n^{g_m}(v_k) \propto (d_{v_k}^{g_m} + \gamma)^{0.75}$, where $d_{v_k}^{g_m}$ represents the popularity of item v_k among its members, which can be easily derived from the user-item interaction graph $\mathbf{G}_{UV}$ as follows $d_{v_k}^{g_m} = \sum_{u_j \in \mathcal{G}_m} w_{jk}$; γ is a smoothing constant parameter which assigns a small probability to items that have no interactions with its group members. Thus, the generated negative items in this way will be more popular among the group's members, and they are more informative and helpful to learn discriminative group-aware social influences λ_{jm}, which finally leads to effective learning of global social influences γ_j, feature weight vector $\vec{\omega}$ and SDAE component.

B. Group Recommendation using SIGR

Once we have learned the model parameters $\Theta = \{\vec{u}_i, \vec{v}_j, \gamma_i, \vec{\omega}, \mathbf{W}_l, \mathbf{b}_l\}$ in SIGR, given an occasional group g_m, we first compute the group-aware social influence λ_{im} of each member in this group according to Equation (12), then we can obtain the group's embedding $\vec{g}_m$ according to Equation (11). Finally, a ranking score for each item v_j can be computed according to the dot product of $\vec{g}_m$ and $\vec{v}_j$, i.e., $f_g(g_m, v_j) = \vec{g}_m \cdot \vec{v}_j$, and then the top-$n$ items with highest ranking scores will be recommended to group g_m.

VI. Experiment Setup

In this section, we introduce the experimental settings, including research questions to answer, datasets, evaluation protocols and comparison methods.

A. Research Questions

We conduct extensive experiments on two large-scale benchmark datasets to answer the following research questions and validate our technical contributions.

RQ1: How does the bipartite graph embedding model BGEM perform in modeling interactions? Can it provide more accurate personalized recommendation for individual users?

RQ2: How does our proposed group recommender model SIGR perform as compared with state-of-the-art group recommenders and various predefined aggregation strategies?

RQ3: Can we improve the group recommendation by leveraging the social network structure information to improve the estimation of personal social influences? If yes, which approach is better at integrating the social network structure features, the linear modeling approach or our proposed SDAE-based deep social influence learning framework?

RQ4: Can we improve the group recommendation by integrating the user-item interaction data? If yes, how do our proposed two model optimization approaches perform on heterogenous interaction data? Furthermore, for the joint optimization approach, which negative sampling strategy is more suitable for the group-item interaction data?

RQ5: How do the hyper-parameters (e.g., η and M) affect the performance of SIGR?

TABLE I: Basic statistics of the two datasets

	Yelp	Douban-Event
# Users	34,504	70,743
# Groups	24,103	110,597
# Items	22,611	60,028
Avg. group size	4.45	4.82
Avg. #interactions for a group	1.12	1.48
Avg. #interactions for a user	13.98	48.38
Avg. #friends for a user	20.77	86.08

B. Datasets

As existing publicly available group recommendation datasets such as CAMRa2011[1] and Movielens-Group [36] consist of either a smaller number of persistent groups or randomly generated groups and they do not contain the user social network information, they are not suitable to evaluate our solution SIGR. This is why we need to create two large-scale benchmark group recommendation datasets based on the Yelp Challenge dataset[2] and Douban-Event dataset [35]. Douban Event is the largest online event-based social network in China that helps people publish and participate in social events. For each user, we acquired her event attendance list and social friend list. For each event, its time and venue were also collected. Yelp allows users to share their check-ins about local businesses (e.g., restaurants) and create social connections with other users. Each check-in or review contains a user, a timestamp and a business, indicating the user visited the business at that time. In our Yelp dataset, we only focus on the restaurants located in the Los Angeles area, where there are 34,504 users and 22,611 restaurants.

As both Yelp and Douban-Event do not contain explicit group information, we extract implicit group activities as follows: we assume if a set of users who are connected on the social network visit the same restaurant at the same time or attend the same event, they are the members of a group and the corresponding activities are group activities. The statistical information of the two datasets is shown in Table 1. From the table, we can see that group-item interaction data is much

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

[2] https://www.yelp.com/dataset/challenge

sparser than the user-item interaction data. For example, in the Douban-Event dataset, a group has only 1.48 interaction records on average, but a user has 48.38 interaction records.

C. Evaluation Methodology

To evaluate the performance of group recommendation systems, we first rank all group-item interaction records according to their timestamps in each dataset, and then use the 80-th percentile as the cut-off point so that the group-item interactions before this point will be used for training, and the rest are for testing. In the training dataset, we choose the last 10% records as the validation data to tune the model hyper-parameters such as η and M. According to the above dividing strategies, we split the group-item interaction records in each dataset $\mathcal{D}$ into the training set $\mathcal{D}_{training}$ and the test set $\mathcal{D}_{test}$.

We employ the widely adopted metric *Hits ratio* [9], [14], [35] to measure the recommendation accuracy. Specifically, for each group-item interaction (g, v) in the test set $\mathcal{D}_{test}$:
(1) We compute a ranking score for item v as well as other items that group g has never interacted with.
(2) We form a top-n recommendation list by picking n items with the highest ranking scores. If the ground-truth item v appears in the top-n recommendation list, we have a *hit*. Otherwise, we have a *miss*.

The metric Hits ratio is defined as follows:

$$Hits@n = \frac{\#hit@n}{|\mathcal{D}_{test}|}, \tag{27}$$

where $\#hit@n$ denotes the number of *hits* in the test set, and $|\mathcal{D}_{test}|$ is the total number of test cases in the test set. A good group recommender model should achieve higher Hits@n.

Besides Hits ratio, we also adopt the commonly used metric Mean Reciprocal Rank (MRR) to measure the recommendation accuracy, and it is defined as follows:

$$MRR = \frac{1}{|\mathcal{D}_{test}|} \sum_{(g,v)\in\mathcal{D}_{test}} \frac{1}{rank(v)}. \tag{28}$$

MRR is an average of the reciprocal rank of the ground-truth item v among all items except those which group g has also interacted with, and a good recommender model should have a bigger MRR value.

Similarly, we also apply the above evaluation procedure to the personalized recommendation for individual users.

D. Comparison Methods

To answer the five research questions, we design the following five experiments with different comparison methods.

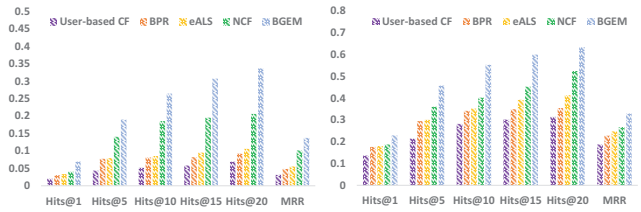
Experiment 1 To answer RQ1, we compare **BGEM** with the following four strong baseline models that learn personal preferences from the user-item interaction data for making recommendations to individual users.
User-based CF: This is a standard user-based collaborative filtering method, which has also been used for group recommendation by integrating predefined aggregation strategies.
BPR [24]: This method adopts a pairwise ranking loss function to optimize the matrix factorization model, which is tailored to learn from implicit feedback. It is a highly competitive baseline for top-n recommendation.
eALS [13]: This is a state-of-the-art weighed matrix factorization method for implicit feedback data. It treats all unobserved interactions as negative instances and weighting them non-uniformly by the item popularity.
NCF [12]: Neural Collaborative Filtering (NCF) is a state-of-the-art collaborative filtering model that uses a neural architecture to model the interactions between users and items.

Experiment 2 To answer RQ2, we compare our SIGR with the following two state-of-the-art group recommender models.
AGREE [7]: Attentive Group Recommendation system adopts NCF to model the group-item interaction data and also uses a standard attention network to learn the weight of each user in the group decision-making. However, this solution does not consider the data sparsity issue of the group-item interaction data in learning user weights, and thus is incapable of leveraging the social network structure information.
PIT [16]: Personal Impact Topic model (PIT) is a topic model developed for group recommendation based on the assumption that influential users will become the representatives of a group to make item selections. However, PIT only performs interaction learning on the group-item interaction data and cannot seamlessly integrate the user-item interaction data.

Although COM [36] is also a state-of-the-art group recommender model, we do not compare with it as AGREE has shown superior performance over COM on all their datasets [7]. To validate the effect of learning the aggregation strategy from data, we compare with another line of methods that apply a predefined aggregation strategy. For these methods, we first run **BGEM** to learn individuals' preferences, and then apply the aggregation strategy to get the group's preferences.
BGEM+avg: It adopts the simplest aggregation strategy [4] that averages the preferences of group members as the group preferences, and it assumes that each member contributes equally to the group's decision.
BGEM+lm: It applies the least misery strategy [2] in which the least satisfied member determines the final group decision, just like the well-known cask principle.
BGEM+mp: It employs the maximum pleasure strategy [4] that tries to maximize the satisfaction of group members.

Experiment 3 To answer RQ3, we design three different versions of our SIGR: **SIGR-N**, **SIGR-L** and **SIGR-D**. SIGR-N does not consider the social network structure information. SIGR-L and SIGR-D exploit and integrate the social network structure features by utilizing the linear modeling approach and SDAE-based deep learning approach, respectively. The node2vec [10] is employed to learn the local network structure features (i.e., user node embedding in the social network).

Experiment 4 To answer RQ4, we compare three model optimization approaches: Simple Training (**ST**), Two-stage Training (**TST**) and Joint Training (**JT**). **ST** optimizes our SIGR model only on the group-item interaction data, while **TST** and **JT** integrate the user-item interaction data. For the joint training approach, we further compare two different negative sampling strategies: the **classic** degree-based sampling



(a) **On Yelp Dataset** (b) **On Douban-Event Dataset**

Fig. 3: **Personalized Recommendation Performance**.

strategy and our proposed **group**-aware sampling strategy. Thus, we implement two versions of **JT**: **JT-C** and **JT-G**.

Experiment 5 To answer RQ5, we investigate how the performance of our SIGR varies w.r.t. different key hyper-parameter setup, including η that controls the contribution of the user-item interaction data during the model optimization, the variance $1/\rho_S^2$ that controls the weight of the regularization term in Equation (13), the number of iterations N and the number of negative samples M. For the hyper-parameters in the SDAE component, we follow their optimal setup in [37]. For other hyper-parameters, we perform cross-validation and use the grid search algorithm to obtain the optimal hyper-parameter setup on the validation dataset (e.g., the dimension of embedding space $d = 50$).

VII. Experimental Results

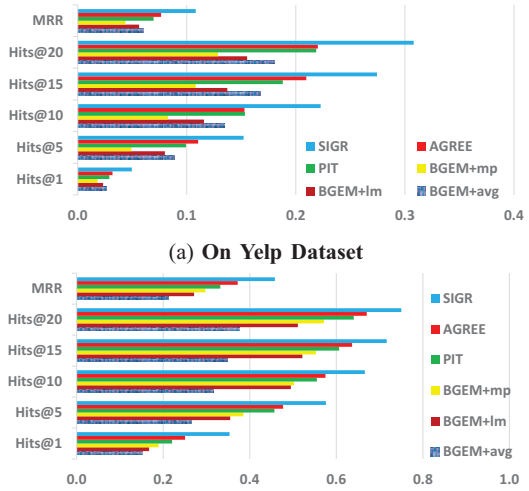
In this section, we present the results of our designed 5 experiments above.

A. Performance on Personalized Recommendation (RQ1)

This experiment studies BGEM's capability of modeling and predicting interactions, and the comparison results on the task of top-n recommendation for individual users (i.e., user-item interaction prediction) are shown in Fig. 3. All differences between BGEM and others are statistically significant ($p < 0.01$). We only show the performance where n is set to 1, 5, 10, 15, 20, as a greater value of n is usually ignored in the recommendation application. From the results, we can observe that BGEM significantly outperforms other state-of-the-art recommender models, which justifies our choice of BGEM as the foundation of our proposed SIGR model. Another observation is that all recommender models achieve higher recommendation accuracy on Douban-Event dataset, because the user-item interaction data on Yelp dataset is sparser. However, the superiority of BGEM is more obvious in the Yelp dataset, showing BGEM is able to better overcome the data sparsity by representing user-item interactions in the form of bipartite graph and extending the negative sampling technique for model optimization. BGEM beats the most advanced NCF model, because NCF is too complex with much more parameters to learn, which easily leads to overfitting in the sparse interaction data setting.

B. Overall Group Recommendation Performance (RQ2)

Fig. 4 shows the results of Experiment 2 on Yelp and Douban-Event datasets. We have the following observations. (1) Our proposed SIGR achieves the best performance on



(a) **On Yelp Dataset**

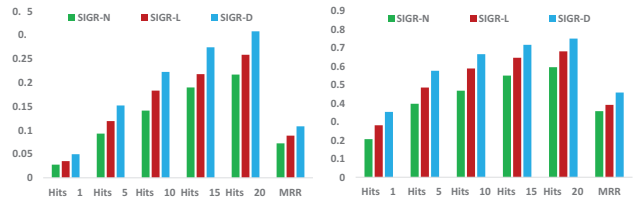
(b) **On Douban-Event Dataset**

Fig. 4: **Group Recommendation Performance**.

the two datasets for group recommendation, significantly outperforming other state-of-the-art group recommender models (all the p-values between our SIGR and each comparison method are much smaller than 0.01, which indicates that the improvements are statistically significant). This validates the effectiveness of our SIGR solution, especially our proposed novel techniques to exploit and integrate the user-item interaction data and the social network data to overcome the sparsity issue of the group-item interaction data in learning user embedding and personal social influence respectively. AGREE cannot leverage the social network structure information and PIT is incapable of integrating user-item interaction data. (2) SIGR, AGREE and PIT consistently outperform BGEM+avg, BGEM+mp and BGEM+ml, showing the advantage of automatically learning the aggregation strategy from data over the predefined aggregation strategies. (3) There is no obvious winner among the predefined aggregation strategies. An aggregation strategy might work well in some datasets but perform poorly in others due to the unique characteristics of the datasets. For example, the average aggregation strategy (BGEM+avg) achieves the best performance among the three predefined aggregation strategies on Yelp dataset, while the most pleasure aggregation strategy (BGEM+mp) outperforms the other two on Douban-Event dataset.

C. Importance of Exploiting Social Network Features (RQ3)

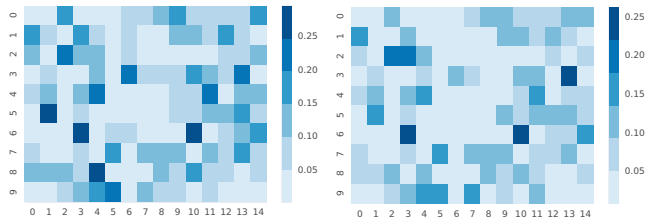
Fig. 5 shows the results of Experiment 3 on Yelp and Douban-Event datasets. The following observations are made from the results. (1) Both SIGR-L and SIGR-D significantly and consistently outperform SIGR-N on both Yelp and Douban-Event datasets, showing the importance of leveraging the social network structure information to improve the group recommendation. As the interaction data generated by occasional groups is extremely sparse, it is infeasible to accurately estimate personal social influences purely from the group-item interaction data. Moreover, for users who have not attended any group activities before (i.e., cold-start users), it



(a) **On Yelp Dataset** (b) **On Douban-Event Dataset**

Fig. 5: **Effect of Leveraging Social Network Structure Features**.

is impossible to infer their social influences. Integrating the social network data can effectively overcome these limitations of the group-item interaction data, as social network structure features strongly indicate user social influences. (2) SIGR-D achieves higher recommendation accuracy than SIG-L, which validates the effect of capturing and modelling non-linear interactions among social network features in estimating personal global social influences.



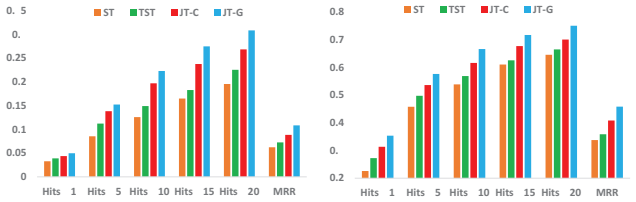
(a) **Attention Weights** (b) **PageRanks**

Fig. 6: **Visualization for the sampled 10 groups w.r.t. attention weights and normalized PageRanks of group members, where *x*-axis represents the member-ID in a group and the *y*-axis represents the group-ID. Note that all original group IDs and user IDs have been transformed into relative IDs. The larger a value is, the darker color its corresponding cell has.**

As we exploit and integrate both global and local social network structure features, we also evaluate the importance of each type of network structure features as well as each specific global feature. Due to the space limitation, we do not report the detailed experimental results but our findings. We find that global network structure features are more effective than local network structure features for improving the group recommendation by facilitating more accurate estimation of personal social influences. Besides, among all global network features, the PageRank feature is the most important one. We randomly select 10 groups for case studies and each group consists of 15 members. The heat map shown in Fig. 6 visualizes the correlation between the learned attention weight λ_{im} of each group member and her/his PageRank. For example, as shown in Fig. 6 (a), users 3 and 10 have the largest attention weights (i.e., the largest social influences) in group 6, which are indicated by their darkest cells. As shown in Fig. 6 (b), these two users also have the highest PageRanks in group 6.

D. Importance of User-Item Interaction Data (RQ4)

Fig. 7 shows the results of Experiment 4. We make three observations from the results. (1) All TST, JT-C and JT-G



(a) **On Yelp Dataset** (b) **On Douban-Event Dataset**

Fig. 7: **Comparison of Different Model Optimization Approaches**.

significantly outperform ST on both Yelp and Douban-Event datasets, showing the importance and necessity of leveraging the user-item interaction data for training our SIGR model. (2) JT-C and JT-G perform better than TST, which shows the advantage of our proposed joint model optimization approach over the two-stage optimization approach. This is because the embedding spaces separately learned from the user-item interaction data and group-item interaction data may not be compatible. Besides, the experimental results also indicate that our proposed joint model optimization approach can effectively perform model optimization on heterogenous interaction data (e.g., the mixture of both user-item and group-item interaction data) and address the issue of data skewness. (3) JT-G achieves better performance than JT-C, showing that our proposed group-aware negative sampling strategy is more suitable for the sparse group-item interaction graph than the classic degree-based negative sampling strategy.

E. Impact of Tuning Hyper-parameters (RQ5)

Tables II–V show the results of Experiment 5. Due to the space limitation, we only show the experimental results on Yelp dataset, and similar results are also achieved on Douban-Event dataset. The hyper-parameters η and $1/\rho_S^2$ play similar roles in our SIGR model, and they control the importance or contributions of the user-item interaction data and the social network data, respectively. $1/\rho_S^2$ is essentially a regularization weight. When η and $1/\rho_S^2$ become extremely small, the effects of the user-item interaction data and the social network data will become ignorable. To study their impacts, we test the performance of SIGR model by varying the values of η and $1/\rho_S^2$ from 0.1 to 1.0 and from 0.001 to 1.0 respectively, and the results are presented in Tables II and III. From the results, we observe that the recommendation accuracy of SIGR first increases with the increasing η and $1/\rho_S^2$, and then begins to decrease. Our SIGR achieves its best performances with $\eta = 0.5$ and $1/\rho_S^2 = 0.05$.

We also investigate the converging performance of our SIGR model with increasing the number of iterations N (i.e., the number of stochastic gradient steps) and the number of negative samples (M) drawn for each positive sample. Table IV presents the performance of our SIGR model w.r.t. the number of iterations. When N is larger than 5 millions, our model converges quickly and its performance becomes very stable. Table V shows the performance of SIGR w.r.t. the number of negative examples M. From the table, we observe that when the number of negative examples is larger than 6, the

performance becomes very stable. Therefore, for each positive example, we do not need to sample many negative examples and just need to sample a few, which ensures the training efficiency of our SIGR model.

VIII. Related Work

There are two lines of research on group recommendation based on the group types [22]. Groups with stable members and rich historical interactions are often referred as persistent groups (also called established groups) while groups formed by users ad-hoc are dubbed as occasional groups. As a persistent group can be treated as a virtual user, conventional personalized recommendation techniques can be straightforwardly adopted for making recommendation to persistent groups [15]. In this paper, we focus on making recommendations to occasional groups.

Making recommendations to occasional groups is much more challenging due to the lack of sufficient group-item interactions. Existing studies on occasional group recommendations focus on aggregation approaches that aggregate individual preferences or recommendation results of the group members as the group preferences or group recommendations. All these aggregation-based group recommendation approaches can be divided into two categories: **late aggregation** and **early aggregation**.

The late aggregation-based approaches [2], [28], [30] first generate the recommendation results or lists for each group member, and then aggregate these individual recommendation results to produce the group recommendations. A variety of aggregation strategies [2], [4], [22], [26], [28] have been proposed, such as average satisfaction, least misery and maximum pleasure, and most of them come from the social choice theory [6]. For example, average satisfaction assigns equal importance to each group member and assumes that each group member contributes equally to the group decision-making. However, these aggregation strategies are heuristic and manually predefined rather than data-driven or learned from data. [21] does a systematic evaluation of all existing predefined aggregation strategies. Their conclusion is that the best-performing aggregation strategy does not exist and their performances depend on the datasets. In other words, there does not exist a predefined fixed aggregation strategy which can perform best on all datasets.

In contrast, the early aggregation-based approaches, such as [7], [32], [36], first aggregate either explicit or implicit user profiles into a group profile or representation, and then produce the group recommendations based on the group profile or representation. The explicit user profiles refer to users' interaction records on items, and the implicit user profiles refer to the latent representations of users' preferences, such as user embedding. A line of this type of work is based on probabilistic generative models or more precisely topic models [16], [33], [34], [36], which model groups by capturing both personal preferences of group members and their impacts in the group. The basic assumption of these models is that users should be treated differently and the notion of *influence* is introduced to quantify the contribution of each group member to the group decision making and implement the data-driven aggregation. Although these models share the similar intuitions with our proposed SIGR model, they do not consider the sparsity issue of the group-item interaction data. Moreover, our SIGR model is technically different from them such as PIT and COM. Compared with our proposed SIGR that takes the bipartite graph embedding model BGEM as the foundation, these topic model-based group recommender models have limited modeling and expressive abilities, since they constrain the key parameters (e.g., users' personal preferences) to be a probability distribution. Moreover, these models are not as flexible as our SIGR, and they are incapable of seamlessly integrating the user-item interaction data to improve the estimation of users' personal preferences.

More recently, [7] developed an Attentive Group Recommendation system (AGREE) by combining a standard attention network with the neural collaborative filtering method (NCF). Compared with our SIGR, AGREE has two serious drawbacks. First, it does not consider the data sparsity issue of the group-item interaction data in learning user weights. Second, its good performance heavily depends on the direct learning of group preference embedding from the group's interaction data and thus cannot make good recommendations for cold-start groups without any interaction record. Our work falls

TABLE II: Impact of Parameter η.

η	Hits@1	Hits@5	Hits@10	Hits@15	Hits@20	MRR
0.1	0.0462	0.1183	0.1671	0.1985	0.2224	0.0902
0.2	0.0473	0.1251	0.1828	0.2211	0.2575	0.0927
0.3	0.0481	0.1328	0.1934	0.2344	0.2601	0.0953
0.4	0.0487	0.1404	0.1969	0.2365	0.2688	0.1005
0.5	**0.0496**	**0.1521**	**0.2227**	**0.2744**	**0.3080**	**0.1082**
0.8	0.0471	0.1388	0.1931	0.2554	0.2922	0.0976
1.0	0.0460	0.1324	0.1977	0.2377	0.2700	0.0967

TABLE III: Impact of Parameter $1/\rho_S^2$.

$1/\rho_S^2$	Hits@1	Hits@5	Hits@10	Hits@15	Hits@20	MRR
0.001	0.0424	0.1247	0.1755	0.2183	0.2472	0.0887
0.01	0.0473	0.1299	0.1844	0.2280	0.2594	0.0967
0.05	**0.0496**	**0.1521**	**0.2227**	**0.2744**	**0.3080**	**0.1082**
0.1	0.0461	0.1312	0.1894	0.2288	0.2643	0.0956
0.5	0.0438	0.1243	0.1812	0.2256	0.2619	0.0893
1.0	0.0425	0.1150	0.1764	0.2135	0.2446	0.0877

TABLE IV: Impact of Parameter N (the number of iterations).

N	Hits@1	Hits@5	Hits@10	Hits@15	Hits@20	MRR
1m	0.0448	0.1312	0.1949	0.2429	0.2716	0.0951
2m	0.0468	0.1408	0.2115	0.2595	0.2846	0.0956
3m	0.0472	0.1465	0.2167	0.2716	0.2896	0.0967
5m	**0.0496**	**0.1521**	**0.2227**	**0.2744**	**0.3080**	**0.1082**
7m	0.0496	0.1521	0.2227	0.2745	0.3081	0.1082
9m	0.0496	0.1522	0.2228	0.2745	0.3081	0.1083

TABLE V: Impact of Parameter M (the number of negative edges sampled for per positive edge).

M	Hits@1	Hits@5	Hits@10	Hits@15	Hits@20	MRR
2	0.0471	0.1493	0.2143	0.2680	0.3063	0.1073
4	0.0489	0.1517	0.2199	0.2736	0.3004	0.1079
6	**0.0496**	**0.1521**	**0.2227**	**0.2744**	**0.3080**	**0.1082**
8	0.0496	0.1521	0.2227	0.2744	0.3081	0.1082
10	0.0496	0.1523	0.2229	0.2746	0.3082	0.1083

into the category of early aggregation-based approaches and focuses on overcoming the data sparsity issue and limitations of the group-item interaction data in learning both users' personal preferences and their social influences. Several novel techniques are proposed to seamless integrate both user-item interaction data and user social network features to address these challenges.

IX. Conclusions

In this paper, we focused on the problem of making recommendations to occasional groups. We proposed a novel Social Influence-based Group Recommender (SIGR) with the powerful BGEM and the attention mechanism as building blocks, which is able to learn both user embedding and user social influences from data in a unified way. To overcome the sparsity and limitations of the group-item interaction data, we developed a novel SDAE-based deep learning component to exploit and integrate both global and local social network structure features to improve the estimation of user social influences, and proposed two model optimization approaches to leverage the user-item interaction data to improve the learning of user embedding. To evaluate the performance of group recommender systems in making recommendations to occasional groups, we created two large-scale benchmark datasets and conducted extensive experiments on them. The experimental results validated the superiority of our solutions by comparing with the state-of-the-art techniques.

Acknowledgement

This work was supported by ARC Discovery Project (Grant No.DP190101985, DP170103954) and ARC Discovery Early Career Researcher Award (Grant No. DE160100308). It was also partially supported by National Natural Science Foundation of China (Grant No.61532018, 61836007, 61832017, 61632016) and Natural Science Research Project of Jiangsu Higher Education Institution (No. 17KJA520003).

References

[1] I. Alina Christensen and S. Schiaffino. Social influence in group recommender systems. *Online Information Review*, 38(4):524–542, 2014.

[2] S. Amer-Yahia, S. B. Roy, A. Chawlat, G. Das, and C. Yu. Group recommendation: Semantics and efficiency. *Proc. VLDB Endow.*, 2(1):754–765, Aug. 2009.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] L. Baltrunas, T. Makcinskas, and F. Ricci. Group recommendations with rank aggregation and collaborative filtering. In *RecSys*, pages 119–126, 2010.

[5] L. Boratto and S. Carta. State-of-the-art in group recommendation and new approaches for automatic identification of groups. In *Information retrieval and mining in distributed environments*, pages 1–20. 2010.

[6] F. Brandt, V. Conitzer, and U. Endriss. Computational social choice. *Multiagent systems*, pages 213–283, 2012.

[7] D. Cao, X. He, L. Miao, Y. An, C. Yang, and R. Hong. Attentive group recommendation. In *SIGIR*, pages 645–654, 2018.

[8] H. Chen, H. Yin, W. Wang, H. Wang, Q. V. H. Nguyen, and X. Li. Pme: Projected metric embedding on heterogeneous networks for link prediction. In *KDD*, pages 1177–1186, 2018.

[9] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys*, pages 39–46, 2010.

[10] A. Grover and J. Leskovec. Node2vec: Scalable feature learning for networks. In *KDD*, pages 855–864, 2016.

[11] J. Guo, Y. Zhu, A. Li, Q. Wang, and W. Han. A social influence approach for group user modeling in group recommendation systems. *IEEE Intelligent Systems*, 31(5):40–48, Sept 2016.

[12] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *WWW*, pages 173–182, 2017.

[13] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*, pages 549–558, 2016.

[14] B. Hu and M. Ester. Spatial topic modeling in online social media for location recommendation. In *RecSys*, pages 25–32, 2013.

[15] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and W. Cao. Deep modeling of group preferences for group-based recommendation. In *AAAI*, pages 1861–1867, 2014.

[16] X. Liu, Y. Tian, M. Ye, and W.-C. Lee. Exploring personal impact for group recommendation. In *CIKM*, pages 674–683, 2012.

[17] J. F. McCarthy and T. D. Anagnost. Musicfx: An arbiter of group preferences for computer supported collaborative workouts. In *CSCW*, pages 363–372, 1998.

[18] K. McCarthy, M. Salamó, L. Coyle, L. McGinty, B. Smyth, and P. Nixon. Cats: A synchronous approach to collaborative group recommendation. In *FLAIRS*, pages 86–91, 2006.

[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[20] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *KDD*, pages 701–710, 2014.

[21] T. Pessemier, S. Dooms, and L. Martens. Comparison of group recommendation algorithms. *Multimedia Tools Appl.*, 72(3):2497–2541, Oct. 2014.

[22] E. Quintarelli, E. Rabosio, and L. Tanca. Recommending new items to ephemeral groups using contextual user influence. In *RecSys*, pages 285–292, 2016.

[23] B. Recht, C. Re, and S. Wright. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, pages 693–701, 2011.

[24] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[25] I. Ronen, I. Guy, E. Kravi, and M. Barnea. Recommending social media content to community owners. In *SIGIR*, pages 243–252, 2014.

[26] S. B. Roy, S. Thirumuruganathan, S. Amer-Yahia, G. Das, and C. Yu. Exploiting group recommendation functions for flexible preferences. In *ICDE*, pages 412–423, 2014.

[27] A. Said, S. Berkovsky, and E. W. De Luca. Group recommendation in context. In *CAMRa*, pages 2–4, 2011.

[28] A. Salehi-Abari and C. Boutilier. Preference-oriented social networks: Group recommendation and inference. In *RecSys*, pages 35–42, 2015.

[29] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, Dec. 2010.

[30] L. Xiao, Z. Min, Z. Yongfeng, G. Zhaoquan, L. Yiqun, and M. Shaoping. Fairness-aware group recommendation with pareto-efficiency. In *RecSys*, pages 107–115, 2017.

[31] M. Xie, H. Yin, H. Wang, and F. Xu. Learning graph-based poi embedding for location-based recommendation. In *CIKM*, 2016.

[32] M. Ye, X. Liu, and W.-C. Lee. Exploring social influence for recommendation: A generative model approach. In *SIGIR*, pages 671–680, 2012.

[33] H. Yin, Z. Hu, X. Zhou, H. Wang, K. Zheng, Q. V. H. Nguyen, and S. Sadiq. Discovering interpretable geo-social communities for user behavior prediction. In *ICDE*, pages 942–953, 2016.

[34] H. Yin, X. Zhou, B. Cui, H. Wang, K. Zheng, and Q. V. H. Nguyen. Adapting to user interest drift for poi recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 28(10):2566–2581, 2016.

[35] H. Yin, L. Zou, Q. V. H. Nguyen, Z. Huang, and X. Zhou. Joint event-partner recommendation in event-based social networks. In *ICDE*, pages 929–940, 2018.

[36] Q. Yuan, G. Cong, and C.-Y. Lin. Com: A generative model for group recommendation. In *KDD*, pages 163–172, 2014.

[37] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma. Collaborative knowledge base embedding for recommender systems. In *KDD*, pages 353–362, 2016.