DiffNet++: A Neural Influence and Interest Diffusion Network for Social Recommendation

Le Wu *Member, IEEE*, Junwei Li, Peijie Sun, Richang Hong, *Member, IEEE*, Yong Ge, *Member, IEEE*, Meng Wang, *Senior Member, IEEE*

Abstract—Social recommendation has emerged to leverage social connections among users for predicting users' unknown preferences, which could alleviate the data sparsity issue in collaborative filtering based recommendation. Early approaches relied on utilizing each user's first-order social neighbors' interests for better user modeling, and failed to model the social influence diffusion process from the global social network structure. Recently, we propose a preliminary work of a neural influence *Diff*usion *Net*work (i.e., DiffNet) for social recommendation [37]. DiffNet models the recursive social diffusion process for each user, such that the influence diffusion hidden in the higher-order social network is captured in the user embedding process. Despite the superior performance of DiffNet, we argue that, as users play a central role in both user-user social network and user-item interest network, only modeling the influence diffusion process in the social network would neglect the latent collaborative interests of users hidden in the user-item interest network. To this end, in this paper, we propose DiffNet++, an improved algorithm of DiffNet that models the neural influence diffusion and interest diffusion in a unified framework. By reformulating the social recommendation as a heterogeneous graph with social network and interest network as input, DiffNet++ advances DiffNet by injecting both the higher-order user latent interest reflected in the user-item graph and higher-order user influence reflected in the user-user graph for user embedding learning. This is achieved by iteratively aggregating each user's embedding from three aspects: the user's previous embedding, the influence aggregation of social neighbors from the social network, and the interest aggregation of item neighbors from the user-item interest network. Furthermore, we design a multi-level attention network that learns how to attentively aggregate user embeddings from these three aspects. Finally, extensive experimental results on two real-world datasets clearly show the effectiveness of our proposed model.

Index Terms—recommender systems, social recommendation, influence diffusion, interest diffusion

1 Introduction

Collaborative Filtering (CF) based recommender systems learn user and item embeddings by utilizing user-item interest behavior data, and have attracted attention from both the academia and industry [32], [27]. However, as most users have limited behavior data, CF suffers from the data sparsity issue [1]. With the development of social networks, users build social relationships and share their item preferences on these platforms. As well supported by the social influence theory, users in a social network would influence each other, leading to similar preferences [10], [2]. Therefore, social recommendation has emerged, which focuses on exploiting social relations among users to alleviate data sparsity and enhancing recommendation performance [17], [18], [12], [37].

In fact, as users play a central role in social platforms with user-user social behavior and user-item interest behavior, the key to social recommendation relies on learning user embeddings with these two kinds of behaviors. For a long time, by treating the user-item interest network as a user-item matrix, CF based models resort to matrix factorization to project both users and items into a low latent space [32], [27], [31]. Most social based recommender systems advance these CF models by leveraging the user-user matrix to enhance each user's embedding learning with social neighbors' records, or regularizing the user embedding learning process with social neighbors [17], [26], [19], [13]. For example, SocialMF [17] and SR [26] added social regularization terms based on social neighbors in the optimization function, and TrustSVD incorporated influences of social neighbors' decisions as additional terms for modeling a user's embedding [13]. In summary, these models leveraged the first-order social neighbors for recommendation, and partially alleviated the data sparsity issue in CF.

Despite the performance improvement of these social recommendation models, we argue that the current social recommendation models are still far from satisfactory. In fact, as shown in Fig. 1, users play a central role in two kinds of behavior networks: the user-user social network and the user-item interest network. On one hand, users naturally form a social graph with a global recursive social diffusion process. Each user is not only influenced by the direct first-order social neighbors, but also the higher-order ego-centric social network structure. E.g., though user $u1$ does not follow $u5$, $u1$ may be largely influenced by $u5$ in the social recommendation process as there are two second-order paths: $u1 \rightarrow u2 \rightarrow u5$ and $u1 \rightarrow u4 \rightarrow u5$. Simply reducing the social network structure to the first-order social neighbors would not well capture these higher-order social influence effect in the recommendation process. On the other hand, given the user-item bipartite interest graph, CF relies on the assumption that "similar users

- *L. Wu, J. Li, P. Sun, R. Hong, M. Wang are with the School of Computer and Information, Hefei University of Technology, Hefei, Anhui 230009, China.*
  Emails: {lewu.ustc, lijunwei.edu, sun.hfut, hongrc.hfut, eric.mengwang}@gmail.com.
- *Y. Ge is with Management Information Systems Department, The University of Arizona, Tucson, Arizona, USA. Email:yongge@email.arizona.edu.*

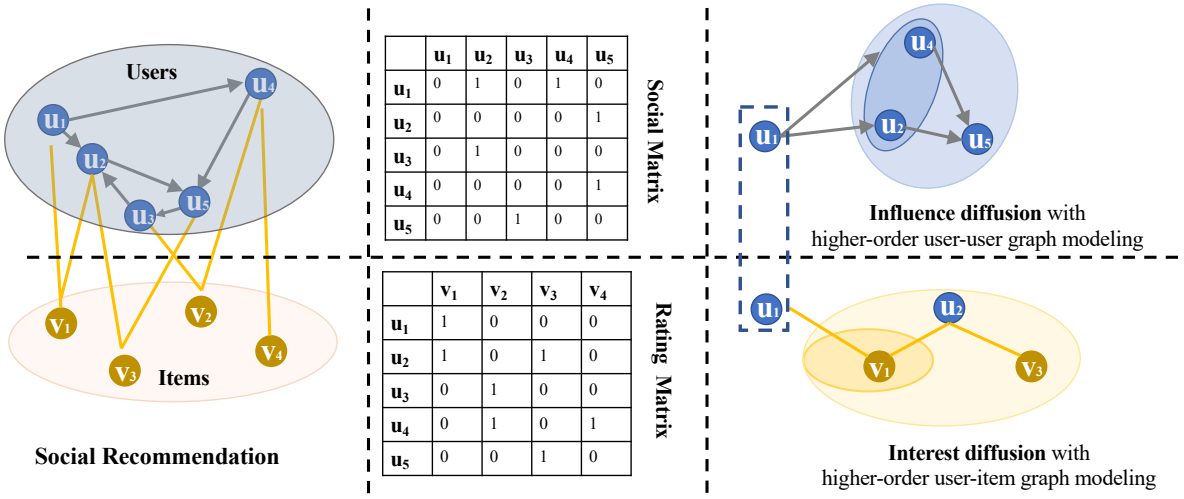


Fig. 1. An overall illustration of social recommendation. The second column shows how traditional models treat this problem with matrix representations of users' two kinds of behaviors. In this paper, we try to model both the influence diffusion and interest diffusion with graph representation of users' two kinds of behaviors.

show similar item interests". Therefore, each user's latent collaborative interests are not only reflected by her rated items but also influenced by similar users' interests from items. E.g, though $u1$ does not show interests for $v3$ with a direct edge connection, the similar user $u2$ (as they have common item interests of $v1$) shows item interest for $v3$ as: $u1 \leftrightarrow v1 \leftrightarrow u2 \leftrightarrow v3$. Therefore, $v3$ is also useful for learning $u1$'s embedding to ensure the collaborative signals hidden in the user-item graph are injected for user embedding learning. To summarize, previous CF and social recommendation models only considered the observed first-order structure of the two graphs, leaving the higher-order structures of users under explored.

To this end, we reformulate users' two kinds of behaviors as a heterogeneous network with two graphs, i.e, a user-user social graph and a user-item interest graph, and propose how to explore the heterogeneous graph structure for social recommendation. In fact, Graph Convolutional Networks (GCNs) have shown huge success for learning graph structures with theoretical elegance, practical flexibility and high performance [5], [7], [20]. GCNs perform node feature propagation in the graph, which recursively propagate node features by iteratively convolutional aggregations from neighborhood nodes, such that the up to K-th order graph structure is captured with K iterations [41]. By treating user-item interactions as a bipartite interest graph and user-user social network as a social graph, some works have applied GCNs separately on these two kinds of graphs [45], [35], [42], [37]. On one hand, given the user-item interest graph, NGCF is proposed to directly encode the collaborative information of users by exploring the higher-order connectivity patterns with embedding propagation [35]. On the other hand, in our previous work, we propose a *Diff*usion neural *Net*work (DiffNet) to model the recursive social diffusion process in the social network, such that the higher-order social structure is directly modeled in the recursive user embedding process [37]. These graph based models showed superior performance compared to the previous non-graph based recommendation models by modeling either graph structure. Nevertheless, how to design a unified model for better user modeling of these two graphs remains under explored.

In this paper, we propose to advance our preliminary DiffNet structure, and jointly model the two graph structure (user-item graph and user-user graph) for social recommendation. While it seems intuitive to perform message passing on both each user's social network and interest network, it is not well designed in practice as these two kinds of graphs serve as different sources to reflect each user's latent preferences. Besides, different users may have different preferences in balancing these two graphs, with some users are likely to be swayed by social neighbors, while others prefer to remain their own tastes. To this end, we propose DiffNet++, an improved algorithm of DiffNet that models the neural influence diffusion and interest diffusion in a unified framework. Furthermore, we design a multi-level attention network structure that learns how to attentively aggregate user embeddings from different nodes in a graph, and then from different graphs. In summary, our main contributions are listed as follows:

- Compared to our previous work of DiffNet [37], we revisit the social recommendation problem as predicting the missing edges in the user-item interest graph by taking both user-item interest graph and user-user social graph as input.
- We propose DiffNet++ that models both the higher-order social influence diffusion in the social network and interest diffusion in the interest network in a unified model. Besides, we carefully design a multi-level attention network to attentively learn how the users prefer different graph sources.
- Extensive experimental results on two real-world datasets clearly show the effectiveness of our proposed DiffNet++ model. Compared to the baseline with the best performance, DiffNet++ outperforms it more than 14% on Yelp and 21% on Flickr for top-10 recommendation.

2 Problem Definition and Related Work

2.1 Problem Definition

In a social recommender system, there are two sets of entities: a user set U ($|U| = M$), and an item set V ($|V| = N$). Users form two kinds of behaviors in the social platforms: making social connections with other users and showing item interests. These two kinds of behaviors could be defined as two matrices: a user-user social connection matrix $\mathbf{S} \in \mathbb{R}^{M \times M}$, and a user-item interaction matrix $\mathbf{R} \in \mathbb{R}^{M \times N}$. In the social matrix $\mathbf{S}$, if user a trusts or follows user b, $s_{ba} = 1$, otherwise it equals 0. We use S_a to represent the user set that user a follows, i.e., $S_a = [b|s_{ba} = 1]$. The user-item matrix $\mathbf{R}$ shows users' rating preferences and interests to items. As some implicit feedbacks (e.g., watching movies, purchasing items, listening to songs) are more common in real-world applications, we also consider the recommendation scenario with implicit feedback [32]. Let $\mathbf{R}$ denote users' implicit feedback based rating matrix, with $r_{ai} = 1$ if user a is interested in item i, otherwise it equals 0. We use R_a represents the item set that user a has consumed, i.e., $R_a = [i|r_{ai} = 1]$, and R_i denotes the user set which consumed the item i, i.e., $R_i = [a|r_{ia} = 1]$.

Given the two kinds of users' behaviors, the user-user social network is denoted as a user-user directed graph: $G_S =< U, \mathbf{S} >$, where U is the nodes of all users in the social network. If the social network is undirected, then user a connects to user b denotes a follows b, and b also follows a, i.e., $s_{ab} = 1 \wedge s_{ba} = 1$. The user interest network denotes users' interests for items, which could be constructed from the user-item rating matrix $\mathbf{R}$ as an undirected bipartite network: $G_I =< U \cup V, \mathbf{R} >$.

Besides, each user a is associated with real-valued attributes (e.g, user profile), denoted as $\mathbf{x}_a$ in the user attribute matrix $\mathbf{X} \in \mathbb{R}^{d1 \times M}$. Also, each item i has an attribute vector $\mathbf{y}_i$ (e.g., item text representation, item visual representation) in item attribute matrix $\mathbf{Y} \in \mathbb{R}^{d2 \times N}$. We formulate the graph based social recommendation problem as:

Definition 1 (Graph Based Social Recommendation). Given the user social network G_S and user interest network G_I, these two networks could be formulated as a heterogeneous graph that combines G_S and G_I as: $G = G_S \cup G_I =< U \cup V, \mathbf{X}, \mathbf{Y}, \mathbf{R}, \mathbf{S} >$. Then, the graph based social recommendation asks that: given graph G in the social network, our goal is to predict users' unknown preferences to items , i.e, the missing links in the graph based social recommendation as : $\hat{R} = f(G) = f(U \cup V, \mathbf{X}, \mathbf{Y}, \mathbf{R}, \mathbf{S})$, where $\hat{R} \in \mathbb{R}^{M \times N}$ denotes the predicted preferences of users to items.

2.2 Preliminaries and Related Work

In this subsection, we summarize the related works for social recommendation into three categories: classical social recommendation models, the recent graph based recommendation models, and attention modeling in the recommendation domain.

Classical Social Recommendation Models. By formulating users' historical behavior as a user-item interaction matrix $\mathbf{R}$, most classical CF models embed both users and items in a low dimension latent space, such that each user's predicted preference to an unknown item turns to the inner product between the corresponding user and item embeddings as [32], [27], [31]:

$$\hat{r}_{ai} = \mathbf{v}_i^T \mathbf{u}_a, \tag{1}$$

where $\mathbf{u}_a$ is the embedding of user a, which is the a-th column of the user embedding matrix $\mathbf{U}$. Similarly, $\mathbf{v}_i$ represents item i's embedding in the i-th column of item embedding matrix $\mathbf{V}$.

In fact, as various specialized matrix factorization models have been proposed for specific tasks, factorization machines is proposed as a general approach to mimic most factorization models with simple feature engineering [31]. Recently, some deep learning based models have been proposed to tackle the CF problem [15], [24]. These approaches advanced previous works by modeling the non-linear complex interactions between users, or the complex interactions between sparse feature input.

The social influence and social correlation among users' interests are the foundation for building social recommender systems[25], [33], [23], [22]. Therefore, the social network among users could be leveraged to alleviate the sparsity in CF and enhance recommendation performance [26], [33], [13]. Due to the superiority of embedding based models for recommendation, most social recommendation models are also built on these embedding models. These social embedding models could be summarized into the following two categories: the social regularization based approaches [17], [26], [19], [38] and the user behavior enhancement based approaches [12], [13]. Specifically, the social regularization based approaches assumed that connected users would show similar embeddings under the social influence diffusion. As such, besides the classical CF pair-wised loss function in BPR [32], an additional social regularization term is incorporated in the overall optimization function as:

$$\sum_{i=1}^{M} \sum_{j=1}^{M} s_{ij} ||\mathbf{u}_i - \mathbf{u}_j||_F^2 = \mathbf{U}(\mathbf{D} - \mathbf{S})\mathbf{U}^T, \tag{2}$$

where $\mathbf{D}$ is a diagonal matrix with $d_{aa} = \sum_{b=1}^{M} s_{ab}$.

Instead of the social regularization term, some researchers argued that the social network provides valuable information to enhance each user's behavior [44], [13]. TrustSVD is such a representative model that shows state-of-the-art performance [12], [13]. By assuming the implicit feedbacks of a user's social neighbors' on items could be regarded as the auxiliary feedback of this user, TrustSVD modeled the predicted preference as:

$$\hat{r}_{ai} = \mathbf{v}_i^T (\mathbf{u}_a + |R_a|^{\frac{-1}{2}} \sum_{i \in R_a} \mathbf{y}_i + |S_a|^{-\frac{1}{2}} \sum_{b \in S_a} \mathbf{u}_b) \tag{3}$$

where $R_a = [i|r_{ai} = 1]$ is the itemset that a shows implicit feedback, and $\mathbf{y}_i$ is an implicit factor vector. Therefore, these first two terms compose SVD++ model that explicitly builds each user's liked items in the user embedding learning process [21]. In the third term, $\mathbf{u}_b$ denotes the latent embedding of user b, who is trusted by a. As such, a's latent embedding is enhanced by considering the influence of her trusted users' latent embeddings in the social network.

As items are associated with attribute information (e.g., item description, item visual information), ContextMF is proposed to combine social context and social network under a collective matrix factorization framework with

carefully designed regularization terms [19]. Social recommendation has also been extended with social circles [29], temporal context [33], rich contextual information [36], and efficient training models without negative sampling [6]. All these previous works focused on how to explore the social neighbors, i.e., the observed links in the social network. Recently, CNSR is proposed to leverage the global social network in the recommendation process [38]. In CNSR, each user's latent embedding is composed of two parts: a free latent embedding (classical CF models), and a social network embedding that captures the global social network structure. Despite the relative improvement of CNSR, we argue that CNSR is still suboptimal as the global social network embedding process is modeled for the network based optimization tasks instead of user preference learning. In contrast to CNSR, our work explicitly models the recursive social diffusion process in the global social network for optimizing the recommendation task.

Graph Convolutional Networks and Applications in Recommendation. GCNs generalize the convolutional operations from the regular Euclidean domains to non-Euclidean graph and have empirically shown great success in graph representation learning [5], [7], [20]. Specifically, GCNs recursively perform message passing by applying convolutional operations to aggregate the neighborhood information, such that the K-th order graph structure is captured with K iterations [20]. By treating the user-item interaction as a graph structure, GCNs have been applied for recommendation [42], [45]. Earlier works relied on spectral GCNs, and suffered from huge time complexity [28], [45]. Therefore, many recent works focus on the spatial based GCNs for recommendation [42], [4], [43], [35]. PinSage is a GCN based content recommendation model by propagating item features in the item-item correlation graph [42]. GC-MC applied graph neural network for CF, with the first order neighborhood is directly modeled in the process [4]. NGCF extended GC-MC with multiple layers, such that the higher-order collaborative signals between users and items can be modeled in the user and item embedding learning process [35].

As the social structure among users could be naturally formulated as a user-user graph, recently we propose a preliminary graph based social recommendation model, DiffNet, for modeling the social diffusion process in recommendation [37]. DiffNet advances classical embedding based models with carefully designed influence diffusion layers, such that how users are influenced by the recursive influence diffusion process in the social network could be well modeled. Given each user a, the user embedding $\mathbf{u}_a$ is sent to the influence diffusion layers. Specifically, let K denote the depth of the influence diffusion layers and $\mathbf{h}_a^k$ is the user representation at the k-th layer of the influence diffusion part. For each user a, her updated embedding $\mathbf{h}_a^{k+1}$ is performed by social diffusion of the embeddings at the k-th layer with two steps: aggregation from her social neighbors at the k-th layer (Eq.(4)), and combination of her own latent embedding $\mathbf{h}_a^k$ at k-th layer and neighbors:

$$\mathbf{h}_{Sa}^{k+1} = Pool(\mathbf{h}_b^k | b \in S_a), \tag{4}$$

$$\mathbf{h}_a^{k+1} = s^{(k+1)}(\mathbf{W}^k \times [\mathbf{h}_{S_a}^{k+1}, \mathbf{h}_a^k]), \tag{5}$$

where the first equation is a pooling operation that transforms all the social trusted users influences into a fixed length vector $\mathbf{h}_{Sa}^{k+1}$, $s(x)$ is a transformation function and we use $s^{(k+1)}$ to denote the transformation function for $(k+1)$-th layer. As such, with a diffusion depth K, DiffNet could automatically models how users are influenced by the K-th order social neighbors in a social network for social recommendation. When $K = 0$, the social diffusion layers disappear and DiffNet degenerates to classical CF models.

In summary, all these previous GCN based models either considered the higher-order social network or the higher-order user interest network for recommendation. There are some recently works that also leverage the graph neural networks for social recommendation [9], [39]. Specifically, GraphRec is designed to learn user representations by fusing first order social and first-order item neighbors with non-linear neural networks [9]. Researchers also proposed deep learning techniques to model the complex interaction of dynamic and static patterns reflected from users' social behavior and item preferences [39]. Although these works relied on deep learning based models with users' two kinds of behaviors, they only modeled the first order structure of the social graph and interest graph. We differ from these works as we simultaneously fuse the higher-order social and interest network structure for better social recommendation.

Attention Models and Applications. As a powerful and common technique, attention mechanism is often adopted when multiple elements in a sequence or set would have an impact of the following output, such that attentive weights are learned with deep neural networks to distinguish important elements [16], [3], [40]. Given a user's rated item history, NAIS is proposed to learn the neural attentive weights for item similarity in item based collaborative filtering [14]. For graph structure data, researchers proposed graph attention networks to attentively learn weights of each neighbor node in the graph convolutional process [34]. In social recommendation, many attention models have been proposed to learn the social influence strength [30], [33], [11], [9]. E.g., with each user's direct item neighbors and social neighbors, GraphRec leverages attention modeling to learn the attentive weights for each social neighbor and each rated item for user modeling [9]. In social contextual recommender systems, users' preferences are influenced by various social contexual aspcect, and an attention network was proposed to learn the attention weight of each social contexutal aspect in the user decision process. Our work is also inspired by the applications of attention modeling, and apply it to fuse the social network and interest network for social recommendation.

3 The Proposed Model

In this section, we first show the overall architecture of our proposed model DiffNet++, followed by each component. After that, we will introduce the learning process of DiffNet++. Finally, we give a detailed discussion of the proposed model.

3.1 Model Architecture

As shown in the related work part, our preliminary work of DiffNet adopts the recursive influence diffusion process

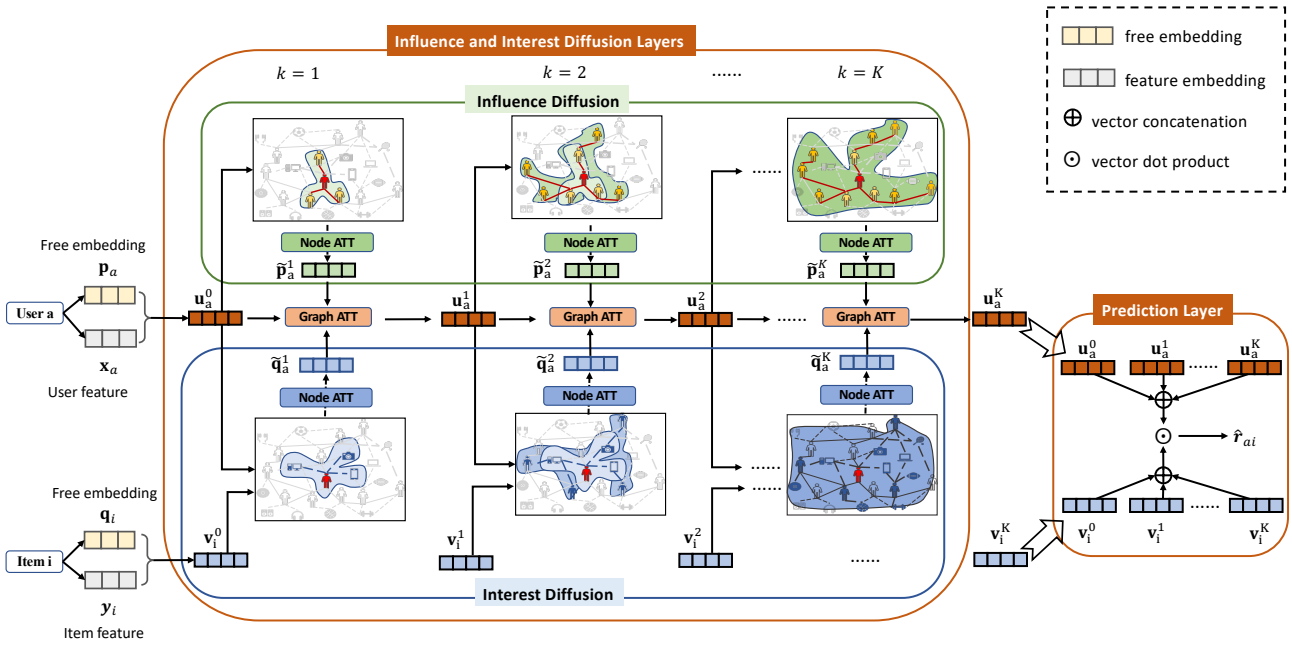


Fig. 2. The overall structure of the DiffNet++ model. As shown in the graph, we use *Node ATT* to denote the node level attention layer in each graph, and *Graph ATT* to denote the graph attention layer when fusing the interest graph representation and social graph representation.

for iterative user embedding learning, such that the up to K-th order social network structure is injected into the social recommendation process [37]. In this part, we propose DiffNet++, an enhanced model of DiffNet that fuses both influence diffusion in the social network G_S and interest diffusion in the interest network G_I for social recommendation. We show the overall neural architecture of DiffNet++ in Fig. 2. The architecture of DiffNet++ contains four main parts: an embedding layer, a fusion layer, the influence and interest diffusion layers, and a rating prediction layer. Specifically, by taking related inputs, the embedding layer outputs free embeddings of users and items, and the fusion layer fuses both the content features and free embeddings. In the influence and interest diffusion layers, we carefully design a multi-level attention structure that could effectively diffuse higher-order social and interest networks. After the diffusion process reaches stable, the output layer predicts the preference score of each unobserved user-item pair.

Embedding Layer. It encodes users and items with corresponding free vector representations. Let $\mathbf{P} \in \mathbb{R}^{M \times D}$ and $\mathbf{Q} \in \mathbb{R}^{N \times D}$ represent the free latent embedding matrices of users and items with D dimensions. Given the one hot representations of user a, the embedding layer performs an index selection and outputs the free user latent embedding $\mathbf{p}_a$, i.e., the transpose of a-th row from user free embedding matrix $\mathbf{P}$. Similarly, item i's embedding $\mathbf{q}_i$ is the transpose of i-th row of item free embedding matrix $\mathbf{Q}$.

Fusion Layer. For each user a, the fusion layer takes $\mathbf{p}_a$ and her associated feature vector $\mathbf{x}_a$ as input, and outputs a user fusion embedding $\mathbf{u}_a^0$ that captures the user's initial interests from different kinds of input data. We model the fusion layer as:

$$\mathbf{u}_a^0 = g(\mathbf{W}_1 \times [\mathbf{p}_a, \mathbf{x}_a,]), \tag{6}$$

where $\mathbf{W}_1$ is a transformation matrix, and $g(x)$ is a transformation function. Without confusion, we omit the bias term. This fusion layer could generalize many typical fusion operations, such as the concatenation operation $\mathbf{u}_a^0 = [\mathbf{x}_a, \mathbf{p}_a]$ by setting $\mathbf{W}_1$ as an identity matrix and $g(x)$ an identity function.

Similarly, for each item i, the fusion layer models the item embedding $\mathbf{v}_i^0$ as a function between its free latent vector $\mathbf{q}_i$ and its feature vector $\mathbf{y}_i$ as:

$$\mathbf{v}_i^0 = g(\mathbf{W}_2 \times [\mathbf{q}_i, \mathbf{y}_i]). \tag{7}$$

Influence and Interest Diffusion Layers. By feeding the output of each user a's fused embedding $\mathbf{u}_a^0$ and each item i's fused embedding $\mathbf{v}_i^0$ into the influence and interest diffusion layers, these layers recursively model the dynamics of this user's latent preference and the item's latent preference propagation in the graph $\mathcal{G}$ with layer-wise convolutions. In detail, at each layer $k+1$, by taking user a's embedding $\mathbf{u}_a^k$ and item i's embedding $\mathbf{v}_i^k$ from previous layer k as input, these layers recursively output the updated embeddings of $\mathbf{v}_i^{k+1}$ and $\mathbf{u}_a^{k+1}$ with diffusion operations. This iteration step starts at $k = 0$ and stops when the recursive process reaches a pre-defined depth K. As each item only appears in the user-item interest graph G_I, in the following, we would first introduce how to update item embeddings, followed by the user embedding with influence and interest diffusions.

For each item i, given its k-th layer embedding $\mathbf{v}_i^k$, we model the updated item embedding $\mathbf{v}_i^{k+1}$ at the $(k+1)$-th layer from G_I as:

$$\tilde{\mathbf{v}}_i^{k+1} = AGG_u(\mathbf{u}_a^k, \forall a \in R_i) = \sum_{a \in R_i} \eta_{ia}^{k+1} \mathbf{u}_a^k, \tag{8}$$

$$\mathbf{v}_i^{k+1} = \tilde{\mathbf{v}}_i^{k+1} + \mathbf{v}_i^k, \tag{9}$$

where $R_i = [a|r_{ia} = 1]$ is the userset that rates item i. $\mathbf{u}_a^k$ is the k-th layer embedding of user a. $\tilde{\mathbf{v}}_i^{k+1}$ is the item i's aggregated embedding from its neighbor users in the user-item interest graph G_I, with η_{ia}^{k+1} denotes the aggregation weight. After obtaining the aggregated embedding $\tilde{\mathbf{v}}_i^{k+1}$ from the k-th layer, each item's updated embedding $\mathbf{v}_i^{k+1}$ is a fusion of the aggregated neighbors' embeddings and the item's emebedding at previous layer k. In fact, we try different kinds of fusion functions, including the concatenation and the addition, and find the addition always shows the best performance. Therefore, we use the addition as the fusion function in Eq.(9).

In the item neighbor aggregation function, Eq.(8) shows the weight of user a to item i. A naive idea is to aggregate the embeddings from i's neighbor users with mean pooling operation, i.e., $\tilde{\mathbf{v}}_i^{k+1}$=$\sum_{a \in R_i} \frac{1}{|R_i|} \mathbf{u}_a^k$. However, it neglects the different interest weights from users, as the importance of different users vary in representing the item representation. Therefore, we use an attention network to learn the attentive weight η_{ia}^{k+1} in Eq.(8) as:

$$\eta_{ia}^{k+1} = MLP_1([\mathbf{v}_i^k, \mathbf{u}_a^k]), \tag{10}$$

where a MultiLayer Perceptrion (MLP) is used to learn the node attention weights with the related user and item embeddings at the k-th layer. After that, we normalize the attention weights with:

$$\eta_{ia}^{k+1} = \frac{exp(\eta_{ia}^{k+1})}{\sum_{b \in R_i} exp(\eta_{ib}^{k+1})}. \tag{11}$$

Specifically, the exponential function is used to ensure each attention weight is larger than 0.

For each user a, let $\mathbf{u}_a^k$ denote her latent embedding at the k-th layer. As users play a central role in both the social network G_S and the interest network G_I, besides her own latent embedding $\mathbf{u}_a^k$, her updated embedding $\mathbf{u}_a^{(k+1)}$ at $(k+1)$-th layer is influenced by two graphs: the influence diffusion in G_S and the interest diffusion in G_I. Let $\tilde{\mathbf{p}}_a^{k+1}$ denote the aggregated embedding of influence diffusion from the social neighbors and $\tilde{\mathbf{q}}_a^{k+1}$ represents the embedding of aggregated interest diffusion from the interested item neighbors at the $(k+1)$-th layer. Then, each user's updated embedding $\mathbf{u}_a^{k+1}$ is modeled as:

$$\mathbf{u}_a^{k+1} = \mathbf{u}_a^k + (\gamma_{a1}^{k+1} \tilde{\mathbf{p}}_a^{k+1} + \gamma_{a2}^{k+1} \tilde{\mathbf{q}}_a^{k+1}), \tag{12}$$

$$\tilde{\mathbf{p}}_a^{k+1} = \sum_{b \in S_a} \alpha_{ab}^{k+1} \mathbf{u}_b^k, \tag{13}$$

$$\tilde{\mathbf{q}}_a^{k+1} = \sum_{i \in R_a} \beta_{ai}^{k+1} \mathbf{v}_i^k, \tag{14}$$

where Eq.(12) shows how each user updates her latent embedding by fusing the influence diffusion aggregation $\tilde{\mathbf{p}}_a^{k+1}$ and interest diffusion aggregation $\tilde{\mathbf{q}}_a^{k+1}$, as well as her own embedding $\mathbf{u}_a^k$ at previous layer. Since each user appears in both the social graph and interest graph, Eq.(13) and Eq.(14) models the influence diffusion aggregation and interest diffusion aggregation from the two graphs respectively. Specifically, α_{ab}^{k+1} denotes the social influence of user b to a at the $(k+1)$-th layer in the social network, and β_{ai}^{k+1} denotes the attraction of item i to user a at the $(k+1)$-th layer in the interest network.

In addition to the user and item embeddings, there are three groups of weights in the above three equations. A naive idea is to directly set equal values of each kind of weights, i.e., $\gamma_{a1}^{(k+1)}$=$\gamma_{a2}^{(k+1)}$=$\frac{1}{2}$, $\alpha_{ab}^{(k+1)}$=$\frac{1}{|S_a|}$, and $\beta_{ai}^{(k+1)}$=$\frac{1}{|R_a|}$. However, this simple idea could not well model the different kinds of weights in the user decision process. In fact, these three groups of weights naturally present a two-layer multi-level structure. Specifically, the social influence strengths and the interest strengths could be seen as node-level weights, which model how each user balances different neighboring nodes in each graph. By sending the aggregations of node level attention into Eq.(12), γ_{al}^{k+1} is the graph level weight that learns to fuse and aggregate information from different graphs. Specifically, the graph layer weights are important as they model how each user balances the social influences and her historical records for user embedding. Different users vary, with some users are more likely to be swayed by the social network while the interests of others are quite stable. Therefore, the weights in the graph attention layer for eah user also need to be personally adapted.

As the three groups of weights represent a multi-level structure, we therefore use a multi-level attention network to model the attentive weights. Specifically, the graph attention network is designed to learn the contribution weight of each aspect when updating a's embedding with different graphs, i.e., $\tilde{\mathbf{p}}_a^{k+1}$ and $\tilde{\mathbf{q}}_a^{k+1}$ in Eq.(12), and the node attention networks are designed to learn the attentive weights in each social graph and each interest graph respectively. Specifically, the social influence score α_{ab}^{k+1} is calculated as follows:

$$\alpha_{ab}^{k+1} = MLP_2([\mathbf{u}_a^k, \mathbf{u}_b^k]). \tag{15}$$

In the above equation, the social influence strength α_{ab}^{k+1} takes the related two users' embeddings at the k-th layer as input, and sending these features into a MLP to learn the complex relationship between features for social influence strength learning. Without confusion, we omit the normalization step of all attention modeling in the following, as all of them share the similar form as shown in Eq.(11).

Similarly, we calculate the interest influence score β_{ai}^{k+1} by taking related user embedding and item embedding as input:

$$\beta_{ai}^{k+1} = MLP_3([\mathbf{u}_a^k, \mathbf{v}_i^k]) \tag{16}$$

After obtaining the two groups of the node attentive weights, the output of the node attention weights are sent to the graph attention network, and we could model the graph attention weights of $\gamma_{al}^{k+1} (l = 1, 2)$ as:

$$\gamma_{a1}^{k+1} = MLP_4([\mathbf{u}_a^k, \tilde{\mathbf{p}}_a^k]) \tag{17}$$

$$\gamma_{a2}^{k+1} = MLP_4([\mathbf{u}_a^k, \tilde{\mathbf{q}}_a^k]) \tag{18}$$

In the above equation, for each user a, the graph attention layer scores not only rely on the user's embedding ($\mathbf{u}_a^k$), but also the weighted representations that are learnt from the node attention network. For example, as shown in Eq.(12), $\gamma_{a1}^{(k+1)}$ denotes the influence diffusion weight for contributing to users' depth $(k+1)$ embedding , with additional input of the learned attentive combination of the influence diffusion aggregation in Eq.(13). Similarly,

$\gamma_{a2}^{(k+1)}$ denotes the interest diffusion weight for contributing to users' depth $(k+1)$ embedding , with additional input of the learned attentive combination of the interest diffusion aggregation in Eq.(14). As $\gamma_{a1}^{k+1} + \gamma_{a2}^{k+1} = 1$, larger γ_{a1}^{k+1} denotes higher influence diffusion effect with less interest diffusion effect. Therefore, the learned aspect importance scores are tailored to each user, which distinguish the importance of the influence diffusion effect and interest diffusion effect during the user's embedding updating process.

Prediction Layer. After the iterative diffusion process with K times, we obtain the embedding set of u and i with $\mathbf{u}_a^k$ and $\mathbf{v}_i^k$ for $k = [0, 1, 2, ..., K]$. Then, for each user a, her final embedding is denoted as: $\mathbf{u}_a^* = [\mathbf{u}_a^0||\mathbf{u}_a^1||...||\mathbf{u}_a^K]$ that concatenates her embedding at each layer. Similarly, each item i's final embedding is : $\mathbf{v}_i^* = [\mathbf{v}_i^0||\mathbf{v}_i^1||...||\mathbf{v}_i^K]$. After that, the predicted rating is modeled as the inner product between the final user and item embeddings:

$$\hat{r}_{ai} = [\mathbf{u}_a^0||\mathbf{u}_a^1||...||\mathbf{u}_a^K]^T[\mathbf{v}_i^0||\mathbf{v}_i^1||...||\mathbf{v}_i^K]. \tag{19}$$

3.2 Model Training

We use a pair-wise ranking based loss function for optimization, which is widely used for implicit feedback [32]:

$$L = \min_{\Theta} \sum_{(a,i)\in R^+ \cup (a,j)\in R^-} -ln\sigma(\hat{r}_{ai} - \hat{r}_{aj}) + \lambda||\Theta||^2 \ . \tag{20}$$

where R^+ denotes the set of positive samples (observed user-item pairs), and R^- denotes the set of negative samples (unobserved user-item pairs that randomly sampled from R). $\sigma(x)$ is sigmoid function. $\Theta = [\Theta_1, \Theta_2]$ is the regularization parameters in our model, with $\Theta_1 = [\mathbf{P}, \mathbf{Q}]$, and the parameter set in the fusion layer and the multi-level attention modeling, i.e., $\Theta_2 = [\mathbf{W}_1, \mathbf{W}_2, [MLP_i]_{i=1,2,3,4}]$. All the parameters in the above loss function are differentiable.

In practice, we implement the proposed model with TensorFlow[1] to train model parameters with mini-batch Adam. For All the trainable parameters, we initialize them with the Gaussian distribution with a mean value of 0 and a standard deviation of 0.01. Besides, we do not deliberately adjust the dimensions of each embedding size in the convolutional layer, all of them keep the same size. As for the several MLPs in the multi-level attention network, we use two-layer structure. In the experiment part, we will give more detail descriptions about the parameter setting.

3.3 Matrix Formulation of DiffNet++

The key idea of our proposed DiffNet++ model is the well designed interest and influence diffusion layers. In fact, this part could be calculated in matrix forms. In the following, we would like to show how to update user and item embedding from the k-th layer to the $(k+1)$-th layer with matrix operations. Let $\mathbf{H}^{(k+1)} = [\eta_{ia}^{k+1}] \in \mathbb{R}^{N\times M}$ denote the matrix representation of attentive item aggregation weigth in Eq.(10), we have:

$$\mathbf{H} = MLP_1(\mathbf{U}^k, \mathbf{V}^k). \tag{21}$$

1. https://www.tensorflow.org/

At the user side, given Eq.(12) ,let $\mathbf{A}^{(k+1)} = [\alpha_{ab}^{k+1}] \in \mathbb{R}^{M\times M}$, $\mathbf{B}^{(k+1)} = [\beta_{ia}^{k+1}] \in \mathbb{R}^{M\times N}$ denote the attentive weight matrices of social network (Eq.(13)) and interest network(Eq.(14)), i.e., the outputs of the node attention layer. We use $\mathbf{\Gamma}^{(k+1)} = [\gamma_{al}^{k+1}] \in \mathbb{R}^{M\times 2}$ to denote the attentive weight matrix of the multi-level networks in Eq.(17) and Eq.(18). All these three attention matrices can be calculated similarly as shown above.

After learning the attention matrices, we could update user and item embeddings at the $(k+1)$-th layer as:

$$\begin{bmatrix}\mathbf{U}^{(k+1)}\\ \mathbf{V}^{(k+1)}\end{bmatrix} = \begin{bmatrix}\mathbf{I}_1 & \mathbf{R}.*\mathbf{B}.*rm(\mathbf{\Gamma}(:,2),N)\\ \mathbf{R}^T.*\mathbf{H} & \mathbf{I}_2\end{bmatrix}\begin{bmatrix}\mathbf{U}^{(k)}\\ \mathbf{V}^{(k)}\end{bmatrix} \tag{22}$$

$$+ \begin{bmatrix}\mathbf{S}.*\mathbf{A}.*rm(\mathbf{\Gamma}(:,\mathbf{1}),M) & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{U}^{(k)}\\ \mathbf{V}^{(k)}\end{bmatrix} \tag{23}$$

$$= \begin{bmatrix}\mathbf{I}_1 + \mathbf{S}.*\mathbf{A}.*rm(\mathbf{\Gamma}(:,1),M) & \mathbf{R}.*\mathbf{B}.*rm(\mathbf{\Gamma}(:,2),N)\\ \mathbf{R}^T.*\mathbf{H} & \mathbf{I}_2\end{bmatrix}\begin{bmatrix}\mathbf{U}^{(k)}\\ \mathbf{V}^{(k)}\end{bmatrix}, \tag{24}$$

where $\mathbf{I}_1$ is an identity matrix with M rows, and $\mathbf{I}_2$ is an identity matrix with N rows. Moreover, $\mathbf{\Gamma}(:,1)$ and $\mathbf{\Gamma}(:,2)$ represent the first column and second column of matrix $\mathbf{\Gamma}$, $.*$ denotes the dot product and $rm(\mathbf{A}, r_1)$ denotes an array that containing r_1 copies of $\mathbf{A}$ in the column dimensions.

Based on the above matrix operations of the social and influence diffusion layers, DiffNet++ is easily implemented by current deep learning frameworks.

3.4 Discussion

Space complexity. As shown in Eq.(20), the model parameters are composed of two parts: the user and item free embeddings Θ_1=[$\mathbf{P}, \mathbf{Q}$], and the parameter set in the fusion layer and the attention modeling, i.e., Θ_2=[$\mathbf{W}_1, \mathbf{W}_2, [MLP_i]_{i=1,2,3,4}$]. Since most embedding based models (e.g., BPR [32], FM [31]) need to store the embeddings of each user and each item, the space complexity of Θ_1 is the same as classical embedding based models and grows linearly with users and items. For parameters in Θ_2, they are shared among all users and items, with the dimension of each parameter is far less than the number of users and items. In practice, we empirically find the two-layer MLP achieve the best performance. As such, this additional storage cost is a small constant that could be neglected. Therefore, the space complexity of DiffNet++ is the same as classical embedding models.

Time complexity. Compared to the classical matrix factorization based models, the additional time cost lies in the influence and interest diffusion layers. Given M users, N items and diffusion depth K, suppose each user directly connects to L_s users and L_i items on average, and each item directly connects to L_u users. At each influence and interest diffusion layer, we need to first calculate the two-level attention weight matrices as shown in Eq.(21), and then update user and item embeddings. Since in practice, MLP layers are very small (e.g., two layers), the time cost for attention modeling is about $O(M(L_s+L_i)D+NL_uD)$. After that, as shown in Eq.(24), the user and item update step also costs $O(M(L_s+L_i)D+NL_uD)$. Since there are K diffusion layers, the total additional time complexity for influence and interest diffusion layers are $O(K(M(L_s+L_i)+NL_u)D)$. In practice, as $L_s, L_i, L_u \ll min\{M, N\}$, the additional time

is linear with users and items, and grows linearly with diffusion depth K. Therefore, the total time complexity is acceptable in practice.

Model Generalization. The proposed DiffNet++ model is designed under the problem setting with the input of user feature matrix $\mathbf{X}$, item feature matrix $\mathbf{Y}$, and the social network $\mathbf{S}$. Specifically, the fusion layer takes users' (items') feature matrix for user (item) representation learning. The layer-wise diffusion layer utilizes the social network structure $\mathbf{S}$ and the interest network structure $\mathbf{R}$ to model how users' latent preferences are dynamically influenced from the recursive influence and interest diffusion process. Next, we would show that our proposed model is generally applicable when different kinds of data input are not available.

When the user (item) features are not available, the fusion layer disappears. In other words, as shown in Eq.(7), each item's latent embedding $\mathbf{v}_i^0$ degenerates to $\mathbf{q}_i$. Similarly, each user's initial layer-0 latent embedding $\mathbf{u}^0 = \mathbf{p}_a$ (Eq.(6)). Similarly, when either the user attributes or the item attributes do not exist, the corresponding fusion layer of user or item degenerates.

4 EXPERIMENTS

Datasets. We conduct experiments on two real-world datasets: *Yelp* and *Flickr*. Yelp[2] is a well-known online location based social network, where users could make friends with others and review restaurants. We use the Yelp dataset that is publicly available[3]. The original dataset contains the social network among users, and users' rating values to locations that range from $[1, 5]$. Flickr[4] is an online image based social sharing platform for users to follow others and share image preferences. In this paper, we use the social image recommendation dataset that is crawled and published by authors in [36], with both the social network structure and users' rating record of images.

For Yelp dataset, as the original ratings are presented with detailed values, we transform the original scores to binary values. If the rating value is larger than 3, we transform it into 1, otherwise it equals 0. For both datasets, we filter out users that have less than 2 rating records and 2 social links and remove items which have been rated less than 2 times. After that, we randomly select 5% records of each user as the test set. Then, 10% of the remaining data will be kept apart as the validation. After that, 85% of the original data is used for training. We show an overview of the characteristics of the two datasets in Table 1.

TABLE 1
The statistics of the two datasets.

Dataset	Yelp	Flickr
Number of Users	17,237	8,358
Number of Items	38,342	82,120
Number of Ratings	204,448	327,815
Number of Links	143,765	187,273
Rating Density	0.03%	0.05%
Link Density	0.05%	0.27%

2. http://www.yelp.com/
3. https://www.yelp.com/dataset
4. http://flickr.com/

TABLE 2
Comparison of the baselines, with "F" represents feature input and "S" denotes the social network input. For the modeling process, we use OI and OS to denote the observed first-order interest network and social network for user embedding learning. We use "HS" to denote the higher-order social information for embedding learning, and "HI" to denote higher-order interest information for embedding learning.

Model	Model Input		User Embedding Ability			
	F	S	OI	OS	HI	HS
BPR [32]	×	×	×	×	×	×
FM [31]	√	×	×	×	×	×
SocialMF [17]	×	√	×	√	×	×
TrustSVD [12]	×	√	√	√	×	×
ContextMF [19]	√	√	×	√	×	×
CNSR [38]	×	√	×	√	×	√
GraphRec [9]	×	√	√	√	×	×
PinSage [42]	√	×	√	×	√	×
NGCF [35]	×	×	√	×	√	×
DiffNet-nf [37]	×	√	×	√	×	√
DiffNet [37]	√	√	×	√	×	√
DiffNet++-nf	×	√	√	√	√	√
DiffNet++	√	√	√	√	√	√

Baselines and Evaluation Metrics. To illustrate the effectiveness of our method, we compare DiffNet++ with competitive baselines, including classical CF models (BPR [32], FM [31]), social based recommendation model (SocialMF [17], TrustSVD [12], ContextMF [19], CNSR [38]), as well as the graph based recommendation models of GraphRec [9], PinSage [42], NGCF [35]. Please note that, in PinSage, we take the user-item graph with both user and item features as input, in order to transform this model for the recommendation task. For our proposed models of DiffNet [37] and DiffNet++, since both models are flexible and could be reduced to simpler versions without user and item features, we use DiffNet-nf and DiffNet++-nf to represent reduced versions of DiffNet and DiffNet++ when removing user and item features. For better illustration, we list the main characteristics of all these models in Table 2, with our proposed models are listed with italic letters. Please note that, as BPR learns free user and item embeddings with the observed user-item ratings. Therefore, the first-order interest network is not learned in the emebedding modeling process. As can be seen from this paper, our proposed DiffNet++-nf and DiffNet++ are the only two models that consider both the higher-order social influence and higher-order interest network for social recommendation.

For the top-N ranking evaluation, we use two widely used metrics, Hit Ratio (HR) [8] and Normalized Discounted Cummulative Gain (NDCG) [8], [37]. Specifically, HR measures the percentage of hit items in the top-N list, and NDCG puts more emphasis on the top ranked items. As we focus on the top-N ranking performance with large itemset, similar as many other works [15], [37], to evaluate the performance, for each user, we randomly select 1000 unrated items that a user has not interacted with as negative samples. Then, we mix these pseudo negative samples and corresponding positive samples (in the test set) to select top-N potential candidates. To reduce the uncertainty in this process, we repeat this procedure 5 times and report the average results.

Parameter Setting. We implement the proposed DiffNet++ with Tensorflow. For the regularization parame-

TABLE 3
Overall comparison of HR@10 and NDCG@10 with different dimension size D.

model	Yelp						Flickr					
	HR			NDCG			HR			NDCG		
	D=16	D=32	D=64	D=16	D=32	D=64	D=16	D=32	D=64	D=16	D=32	D=64
BPR	0.2435	0.2616	0.2632	0.1468	0.1573	0.1554	0.0773	0.0812	0.0795	0.0611	0.0652	0.0628
FM	0.2768	0.2835	0.2825	0.1698	0.1720	0.1717	0.1115	0.1212	0.1233	0.0872	0.0968	0.0954
SocialMF	0.2571	0.2709	0.2785	0.1655	0.1695	0.1677	0.1001	0.1056	0.1174	0.0862	0.0910	0.0964
TrustSVD	0.2826	0.2854	0.2939	0.1683	0.1710	0.1749	0.1352	0.1341	0.1404	0.1056	0.1039	0.1083
ContextMF	0.2985	0.3011	0.3043	0.1758	0.1808	0.1818	0.1405	0.1382	0.1433	0.1085	0.1079	0.1102
CNSR	0.2702	0.2817	0.2904	0.1723	0.1745	0.1746	0.1146	0.1198	0.1229	0.0913	0.0942	0.0978
GraphRec	0.2873	0.2910	0.2912	0.1663	0.1677	0.1812	0.1195	0.1211	0.1231	0.0910	0.0924	0.0930
PinSage	0.2944	0.2966	0.3049	0.1753	0.1786	0.1855	0.1192	0.1234	0.1257	0.0937	0.0986	0.0998
NGCF	0.3050	0.3068	0.3042	0.1826	0.1844	0.1828	0.1110	0.1150	0.1189	0.0880	0.0895	0.0945
DiffNet-nf	0.3126	0.3156	0.3195	0.1854	0.1882	0.1928	0.1342	0.1317	0.1408	0.1040	0.1034	0.1089
DiffNet	0.3293	0.3437	0.3461	0.1982	0.2095	0.2118	0.1476	0.1588	0.1657	0.1121	0.1242	0.1271
DiffNet++-nf	0.3194	0.3199	0.3230	0.1914	0.1944	0.1942	0.1410	0.1480	0.1503	0.1100	0.1132	0.1169
DiffNet++	**0.3406**	**0.3552**	**0.3694**	**0.2070**	**0.2158**	**0.2263**	**0.1562**	**0.1678**	**0.1832**	**0.1213**	**0.1286**	**0.1420**

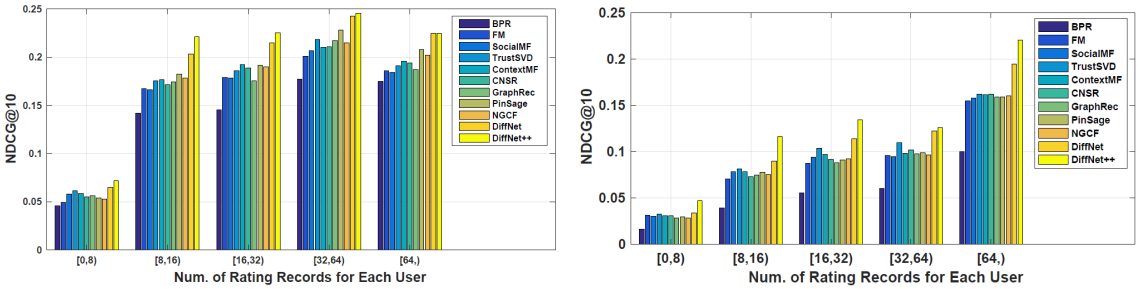
TABLE 4
Overall comparison of HR@N and NDCG@N with different top-N values (D=64).

model	Yelp						Flickr					
	HR			NDCG			HR			NDCG		
	N=5	N=10	N=15	N=5	N=10	N=15	N=5	N=10	N=15	N=5	N=10	N=15
BPR	0.1695	0.2632	0.3252	0.1231	0.1554	0.1758	0.0651	0.0795	0.1037	0.0603	0.0628	0.0732
FM	0.1855	0.2825	0.3440	0.1341	0.1717	0.1876	0.0989	0.1233	0.1473	0.0866	0.0954	0.1062
SocialMF	0.1739	0.2785	0.3365	0.1324	0.1677	0.1841	0.0813	0.1174	0.1300	0.0723	0.0964	0.1061
TrustSVD	0.1882	0.2939	0.3688	0.1368	0.1749	0.1981	0.1089	0.1404	0.1738	0.0978	0.1083	0.1203
ContextMF	0.2045	0.3043	0.3832	0.1484	0.1818	0.2081	0.1095	0.1433	0.1768	0.0920	0.1102	0.1131
CNSR	0.1877	0.2904	0.3458	0.1389	0.1746	0.1912	0.0920	0.1229	0.1445	0.0791	0.0978	0.1057
GraphRec	0.1915	0.2912	0.3623	0.1279	0.1812	0.1956	0.0931	0.1231	0.1482	0.0784	0.0930	0.0992
PinSage	0.2105	0.3049	0.3863	0.1539	0.1855	0.2137	0.0934	0.1257	0.1502	0.0844	0.0998	0.1046
NGCF	0.1992	0.3042	0.3753	0.1450	0.1828	0.2041	0.0891	0.1189	0.1399	0.0819	0.0945	0.0998
DiffNet-nf	0.2101	0.3195	0.3982	0.1535	0.1928	0.2164	0.1087	0.1408	0.1709	0.0979	0.1089	0.1192
DiffNet	0.2276	0.3461	0.4217	0.1679	0.2118	0.2307	0.1178	0.1657	0.1855	0.1072	0.1271	0.1301
DiffNet++-nf	0.2112	0.3230	0.3989	0.1551	0.1942	0.2176	0.1140	0.1503	0.1799	0.1021	0.1169	0.1256
DiffNet++	**0.2503**	**0.3694**	**0.4493**	**0.1841**	**0.2263**	**0.2497**	**0.1412**	**0.1832**	**0.2203**	**0.1269**	**0.1420**	**0.1544**

ter λ in Eq.(20), we empirically try it in the range of [0.0001, 0.001, 0.01, 0.1] and finally set $\lambda = 0.01$ to get the best performance. For the fusion layer in Eq.(6) and Eq.(7), we first transform the each user (item) feature vector to the same free embedding space, and calculate as: $\mathbf{u}_a^0 = \mathbf{W}_1 \times \mathbf{x}_a + \mathbf{p}_a$, and $\mathbf{v}_a^0 = \mathbf{W}_2 \times \mathbf{y}_i + \mathbf{q}_a$. For attention modeling, we resort to MLP with two layers. For our proposed model, we initialize all of them with a Gaussian distribution with a mean value of 0 and the standard deviation of 0.01. We use the Adam optimizer for with an initial learning rate of 0.001, and the training batch size is 512. In the training process, as there are much more unobserved items for each user, we randomly select 8 times pseudo negative samples for each user at each iteration. Since each iteration we change the pseudo negative samples, each unobserved item gives very weak signal. For all the baselines, we carefully tune the parameters to ensure the best performance.

4.1 Overall Performance Comparison

We show the overall performance of all models for top-10 recommendation with different embedding size D in Table 3. We notice that besides BPR, nearly all models show better performance with the increase of dimension D. All models improve over BPR, which only leverages the observed user-item rating matrix for recommendation, and suffer the data sparsity issue in practice. TrustSVD and SocialMF utilize social neighbors of each user as auxiliary information to alleviate this problem. GraphRec further improves over these traditional social recommendation models by jointly considering the first-order social neighbors and interest neighbors in the user embedding process. However, GraphRec only models the first-order relationships of two graphs for user embedding learning, with the higher-order graph structures are neglected. For GCN based models, PinSage and NGCF model the higher-order user-item graph structure, and DiffNet models the higher-order social structure. These graph neural models beat matrix based baselines by a large margin, showing the effectiveness in leveraging higher-order graph structure for recommendation. Our proposed DiffNet++ model always performs the best under any dimension D, indicating the effectiveness of modeling the recursive diffusion process in the social interest network. Besides, we observe DiffNet++ and DiffNet always show better performance compared to their counterparts that do not model the user and item features, showing the effectiveness of injecting both feature and latent embeddings in the fusion layer. We further compare the performance of different models with different top-N values in Table 4, and the



(a) Yelp dataset. (b) Flickr dataset.

Fig. 3. Performance under different rating sparsity on two datasets.

overall trend is the same as analyzed before. Therefore, we could empirically conclude the superiority of our proposed models. As nearly all models showed better performance at $D = 64$, we would use this setting to show the comparison of different models in the following analysis.

4.2 Performance Under Different Sparsity

In this part, we would like to investigate how different models perform under different rating sparsity. Specifically, we first group users into different interest groups based on the number of observed ratings of each user. E.g, $[8, 16)$ means each user has at least 8 rating records and less than 16 rating records. Then, we calculate the average performance of each interest group. The sparsity analysis on Yelp dataset and Flickr dataset are shown in Fig. 3(a) and Fig. 3(b) respectively. From both datasets, we observe that as users have more ratings, the overall performance increases among all models. This is quite reasonable as all models could have more user behavior data for user embedding modeling. Our proposed models consistently improve all baselines, and especially show larger improvements on sparser dataset. E.g., when users have less than 8 rating records, DiffNet++ improves 22.4% and 45.0% over the best baseline on Yelp and Flickr respectively.

4.3 Detailed Model Analysis

Diffusion Depth K. The number of layer K is very important, as it determines the diffusion depth of different graphs. We show the results of different K values for both datasets in Table 5. The column of "Improve" shows the performance changes compared to the best setting, i.e., $K = 2$. When K increases from 0 to 1, the performance increases quickly (DiffNet++ degenerates to BPR when $K = 0$), and it achieves the best performance when $K = 2$. However, when we continue to increase the layer to 3, the performance drops. We empirically conclude that 2-hop higher-order social interest graph structure is enough for social recommendation. And adding more layers may introduce unnecessary neighbors in this process, leading to performance decrease. Other related studies have also empirically found similar trends [20], [42].

The Effects of Multi-level Attention. A key characteristic of our proposed model is the multi-level attention modeling by fusing the social network and interest network for recommendation. In this subsection, we discuss the effects of different attention mechanisms. We show the results of different attention modeling combinations in in Table 6, with "AVG" means we directly set the equal attention weights without any attention learning process. As can be observed from this table, either the node level attention or the graph level attention modeling could improve the recommendation results, with the graph level attention shows better results. When combing both the node level attention and the graph level attention, the performance can be further improved. E.g., for the Flickr dataset, the graph level attention improves more than 4% compared to the results of the average attention, and combining the node level attention further improves about 2%. However, the improvement of attention modeling varies in different datasets, with the results of the Yelp dataset is not as significant as the Flickr dataset. This observation implies that the usefulness of considering the importance strength of different elements in the modeling process varies, and our proposed multi-level attention modeling could adapt to different datasets' requirements.

Attention Value Analysis. For each user a at layer k, the graph level attention weights of γ_{a1}^k and γ_{a2}^k denote the social influence diffusion weight and the interest diffusion weight. A larger value of γ_{a1}^k indicates the social influence diffusion process is more important to capture the user embedding learning with less influence from the interest network. In Table 7, we show the learned mean and variance of all users' attention weights at the graph level at each layer k. Since both datasets receive the best performance at $K = 2$, we show the attention weights at the first diffusion layer ($k = 1$) and the second diffusion layer $k = 2$. There are several interesting findings. First, we observe for both datasets, at the first diffusion layer with $k = 1$, the average value of the social influence strength γ_{a1}^1 are very high, indicating the first-order social neighbors play a very important role in representing each user's first layer representation. This is quite reasonable as users' rating behavior are very sparse, and leveraging the first order social neighbors could largely improve the recommendation performance. When $k = 2$, the average social influence strength γ_{a1}^2 varies among the two datasets, with the Yelp dataset shows a larger average social influence weight, while the Flickr dataset shows a larger interest influence weight with quite small value of average social influence weight. We guess a possible reason is that, as shown in Table 1, Flickr dataset shows denser social links compared to the Yelp dataset, with a considerable amount of

TABLE 5
HR@10 and NDCG@10 performance with different diffusion depth K ($D = 64$).

Depth K	Yelp				Flickr			
	HR	Improve	NDCG	Improve	HR	Improve	NDCG	Improve
K = 2	0.3694	-	0.2263	-	0.1832	-	0.1420	-
K = 0	0.2632	-28.32%	0.1554	-30.81%	0.0795	-55.21%	0.0628	-53.86%
K = 1	0.3566	-2.89%	0.2159	-3.87%	0.1676	-5.58%	0.1283	-5.73%
K = 3	0.3626	-1.25%	0.2215	-1.38%	0.1743	-1.80%	0.1347	-1.03%

TABLE 6
HR@10 and NDCG@10 performance with different attentional variants (D = 64).

Graph Attention	Node Attention	Yelp				Flickr			
		HR	Improve	NDCG	Improve	HR	Improve	NDCG	Improve
AVG	AVG	0.3631	-	0.2224	-	0.1733	-	0.1329	-
AVG	ATT	0.3657	+0.72%	0.2235	+0.49%	0.1792	+3.40%	0.1368	+2.93%
ATT	AVG	0.3662	+0.85%	0.2249	+1.12%	0.1814	+4.67%	0.1387	+4.36%
ATT	ATT	0.3694	+1.74%	0.2263	+1.75%	0.1832	+5.71%	0.1420	+6.85%

TABLE 7
Mean statistics of the graph level attention values ($K = 2$), with γ_{a1}^k is the social influence weight and γ_{a2}^k is the interest weight.

Layer k	Yelp		Flickr	
	Social γ_{a1}^k	Interest γ_{a2}^k	Social γ_{a1}^k	Interest γ_{a1}^k
k=1	0.7309	0.2691	0.8381	0.1619
k=2	0.6888	0.3112	0.0727	0.9273

directed social links at the first diffusion layer, the average weight of the second layer social neighbors decreases.

5 Conclusions and Future Work

In this paper, we presented a neural social and interest diffusion based model, i.e., DiffNet++, for social recommendation. We argued that, as users play a central role in social network and interest network, jointly modeling the higher-order structure of these two networks would mutually enhance each other. By formulating the social recommendation as a heterogeneous graph, we recursively learned the user embedding from convolutions on user social neighbors and interest neighbors, such that both the higher-order social structure and higher-order interest network are directly injected in the user modeling process. Furthermore, we designed a multi-level attention network to attentively aggregate the graph and node level representations for better user modeling. Experimental results on two real-world datasets clearly showed the effectiveness of our proposed model. In the future, we would like to explore the graph reasoning models to explain the paths for users' behaviors.

References

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *TKDE*, (6):734–749, 2005.

[2] A. Anagnostopoulos, R. Kumar, and M. Mahdian. Influence and correlation in social networks. In *SIGKDD*, pages 7–15. ACM, 2008.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[4] R. v. d. Berg, T. N. Kipf, and M. Welling. Graph convolutional matrix completion. In *SIGKDD*, 2018.

[5] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. In *ICLR*, 2014.

[6] C. Chen, M. Zhang, C. Wang, W. Ma, M. Li, Y. Liu, and S. Ma. An efficient adaptive transfer neural network for social-aware recommendation. In *SIGIR*, pages 225–234, 2019.

[7] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, pages 3844–3852, 2016.

[8] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *TOIS*, 22(1):143–177, 2004.

[9] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin. Graph neural networks for social recommendation. In *WWW*, pages 417–426, 2019.

[10] N. E. Friedkin. *A structural theory of social influence*, volume 13. Cambridge University Press, 2006.

[11] L. Gong and Q. Cheng. Adaptive edge features guided graph attention networks. *ArXiv*, abs/1809.02709, 2018.

[12] G. Guo, J. Zhang, and N. Yorke-Smith. Trustsvd: collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In *AAAI*, pages 123–129, 2015.

[13] G. Guo, J. Zhang, and N. Yorke-Smith. A novel recommendation model regularized with user trust and item ratings. *TKDE*, 28(7):1607–1620, 2016.

[14] X. He, Z. He, J. Song, Z. Liu, Y.-G. Jiang, and T.-S. Chua. Nais: Neural attentive item similarity model for recommendation. *TKDE*, 2018.

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

[16] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *TPAMI*, (11):1254–1259, 1998.

[17] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Recsys*, pages 135–142, 2010.

[18] M. Jiang, P. Cui, R. Liu, Q. Yang, F. Wang, W. Zhu, and S. Yang. Social contextual recommendation. In *CIKM*, pages 45–54, 2012.

[19] M. Jiang, P. Cui, F. Wang, W. Zhu, and S. Yang. Scalable recommendation with social contextual information. *TKDE*, 26(11):2789–2802, 2014.

[20] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[21] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *SIGKDD*, pages 426–434, 2008.

[22] A. D. Kramer, J. E. Guillory, and J. T. Hancock. Experimental evidence of massive-scale emotional contagion through social networks. *PNAS*, 111(24):8788–8790, 2014.

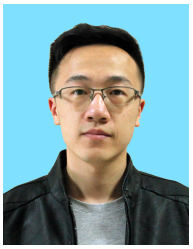
[23] K. Lewis, M. Gonzalez, and J. Kaufman. Social selection and peer influence in an online social network. *PNAS*, 109(1):68–72, 2012.

[24] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *SIGKDD*, pages 1754–1763, 2018.

[25] Q. Liu, B. Xiang, E. Chen, H. Xiong, F. Tang, and J. X. Yu. Influence maximization over large-scale social networks: A bounded linear approach. In *CIKM*, pages 171–180, 2014.

[26] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *WSDM*, pages 287–296, 2011.
[27] A. Mnih and R. R. Salakhutdinov. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264, 2008.
[28] F. Monti, M. Bronstein, and X. Bresson. Geometric matrix completion with recurrent multi-graph neural networks. In *NIPS*, pages 3697–3707, 2017.
[29] X. Qian, H. Feng, G. Zhao, and T. Mei. Personalized recommendation combining user interest and social circle. *TKDE*, 26(7):1763–1777, 2014.
[30] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, and J. Tang. Deepinf: Social influence prediction with deep learning. In *SIGKDD*, pages 2110–2119, 2018.
[31] S. Rendle. Factorization machines. In *ICDM*, pages 995–1000, 2010.
[32] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.
[33] P. Sun, L. Wu, and M. Wang. Attentive recurrent social recommendation. In *SIGIR*, pages 185–194, 2018.
[34] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. In *ICLR*, 2018.
[35] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua. Neural graph collaborative filtering. In *SIGIR*, pages 165–174, 2019.
[36] L. Wu, L. Chen, R. Hong, Y. Fu, X. Xie, and M. Wang. A hierarchical attention model for social contextual image recommendation. *TKDE*, 2019.
[37] L. Wu, P. Sun, Y. Fu, R. Hong, X. Wang, and M. Wang. A neural influence diffusion model for social recommendation. In *SIGIR*, pages 235–244, 2019.
[38] L. Wu, P. Sun, R. Hong, Y. Ge, and M. Wang. Collaborative neural social recommendation. *TSMC: Systems*, pages 1–13, 2018.
[39] Q. Wu, H. Zhang, X. Gao, P. He, P. Weng, H. Gao, and G. Chen. Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. In *WWW*, pages 2091–2102, 2019.
[40] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, pages 2048–2057, 2015.
[41] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.
[42] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *SIGKDD*, pages 974–983, 2018.
[43] J. Zhang, X. Shi, S. Zhao, and I. King. STAR-GCN: stacked and reconstructed graph convolutional networks for recommender systems. In *IJCAI*, pages 4264–4270, 2019.
[44] T. Zhao, J. McAuley, and I. King. Leveraging social connections to improve personalized ranking for collaborative filtering. In *CIKM*, pages 261–270, 2014.
[45] L. Zheng, C.-T. Lu, F. Jiang, J. Zhang, and P. S. Yu. Spectral collaborative filtering. In *Recsys*, pages 311–319, 2018.

Le Wu (M'18) is currently an associate professor at the Hefei University of Technology (HFUT), China. She received the Ph.D. degree from the University of Science and Technology of China (USTC). Her general area of research interests is data mining, recommender systems and social network analysis. She has published more than 40 papers in referred journals and conferences. Dr. Le Wu is the recipient of the Best of SDM 2015 Award, and the Distinguished Dissertation Award from China Association for Artificial Intelligence (CAAI) 2017.



Junwei Li is currently working towards the Ph.D. degree at Hefei University of Technology, China. His research interests include data mining and recommender systems.

Peijie Sun is a Ph.D. student with the Hefei University of Technology. He received the master degree from the same university in 2018. He has published several papers in leading conferences and journals, including SIGIR and IEEE Trans. on SMC: Systems. His current research interests include data mining and recommender systems.

Richang Hong (M'12) is currently a professor at HFUT. He received the Ph.D. degree from USTC, in 2008. He has co-authored over 60 publications in the areas of his research interests, which include multimedia question answering, video content analysis, and pattern recognition. He is a member of the Association for Computing Machinery. He was a recipient of the best paper award in the ACM Multimedia 2010.

Yong Ge is an assistant professor of Management Information Systems in University of Arizona. He received the Ph.D. degree in information technology from Rutgers, The State University of New Jersey in 2013. His research interests include data mining and business analytics. He received the ICDM-2011 Best Research Paper Award.. He has published prolifically in refereed journals and conference proceedings, such as IEEE Transactions on Knowledge and Data Engineering, ACM Transactions on Information Systems, ACM Transactions on Knowledge Discovery from Data, ACM SIGKDD, SIAM SDM, IEEE ICDM, and ACM RecSys. He also was the Program Committee member at ACM SIGKDD, IEEE ICDM etc.

Meng Wang(SM'17) is a professor at the Hefei University of Technology, China. He received his B.E. degree and Ph.D. degree in the Special Class for the Gifted Young and the Department of Electronic Engineering and Information Science from the University of Science and Technology of China (USTC), Hefei, China, in 2003 and 2008, respectively. His current research interests include multimedia content analysis, computer vision, and pattern recognition. He has authored more than 200 book chapters, journal and conference papers in these areas. He is the recipient of the ACM SIGMM Rising Star Award 2014. He is an associate editor of IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE), IEEE Transactions on Circuits and Systems for Video Technology (IEEE TCSVT), IEEE Transactions on Multimedia (IEEE TMM), and IEEE Transactions on Neural Networks and Learning Systems (IEEE TNNLS).