A Neural Influence Diffusion Model for Social Recommendation

Le Wu
Hefei University of Technology

Peijie Sun*
sun.hfut@gmail.com
Hefei University of Technology

Yanjie Fu
Missouri University of Science and Technology

Richang Hong
Hefei University of Technology

Xiting Wang
Microsoft Research

Meng Wang
Hefei University of Technology

ABSTRACT

Precise user and item embedding learning is the key to building a successful recommender system. Traditionally, Collaborative Filtering (CF) provides a way to learn user and item embeddings from the user-item interaction history. However, the performance is limited due to the sparseness of user behavior data. With the emergence of online social networks, social recommender systems have been proposed to utilize each user's local neighbors' preferences to alleviate the data sparsity for better user embedding modeling. We argue that, for each user of a social platform, her potential embedding is influenced by her trusted users, with these trusted users are influenced by the trusted users' social connections. As social influence recursively propagates and diffuses in the social network, each user's interests change in the recursive process. Nevertheless, the current social recommendation models simply developed static models by leveraging the local neighbors of each user without simulating the recursive diffusion in the global social network, leading to suboptimal recommendation performance. In this paper, we propose a deep influence propagation model to stimulate how users are influenced by the recursive social diffusion process for social recommendation. For each user, the diffusion process starts with an initial embedding that fuses the related features and a free user latent vector that captures the latent behavior preference. The key idea of our proposed model is that we design a layer-wise influence propagation structure to model how users' latent embeddings evolve as the social diffusion process continues. We further show that our proposed model is general and could be applied when the user (item) attributes or the social network structure is not available. Finally, extensive experimental results on two real-world datasets clearly show the effectiveness of our proposed model, with more than 13% performance improvements over the best baselines for top-10 recommendation on the two datasets.

CCS CONCEPTS

• **Information systems** → **Social recommendation**; **Personalization**; • **Human-centered computing** → *Social networks.*

KEYWORDS

graph neural networks, social recommendation, influence diffusion, personalization

ACM Reference Format:
Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A Neural Influence Diffusion Model for Social Recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3331184.3331214

1 INTRODUCTION

By providing personalized item suggestions for each user, recommender systems have become a cornerstone of the E-commerce shopping experience [2, 23, 41]. Among all recommendation algorithms, learning low dimensional user and item embeddigs is a key building block that have been widely studied [20, 31]. With the learned user and item embeddings, it is convenient to approximate the likelihood or the predicted preference that a user would give to an item by way of a simple inner product between the corresponding user embedding and item embedding.

Many efforts have been devoted to designing sophisticated models to learn precise user and item embeddings. In the typical collaborative filtering scenario with user-item interaction behavior, the latent factor based approaches have received great success [20, 31, 32]. However, the recommendation performance is unsatisfactory due to the sparseness of user-item interaction data. As sometimes users and items are associated with features, factorization machines generalize most latent factor models with an additional linear regression function of user and item features [31]. Recently, researchers also designed more advanced neural models based on latent factor models and FMs [9, 12]. E.g., NeuMF extends over latent factor based models by modeling the complex relationships between user and item embedding with a neural architecture [12]. These deep embedding models advance the performance of previous shallow embedding models. Nevertheless, the recommendation performance is still hampered by the sparse data.

Luckily, with the prevalence of online social networks, more and more people like to express their opinions of items in these social platforms. The social recommender systems have emerged as a promising direction, which leverage the social network among users to alleviate the data sparsity issue and enhance recommendation performance [7, 16, 25, 35]. These social recommendation approaches are based on the social influence theory that states connected people would influence each other, leading to the similar

*Corresponding author.

SIGIR '19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331214

interests among social connections [3, 4, 22]. E.g., social regularization has been empirically proven effective for social recommendation, with the assumption that connected users would share similar latent embeddings [15, 16, 25]. TrustSVD++ extended the classic latent factor based models by incorporating each user's trusted friends' feedbacks to items as the auxiliary feedback of the active user [7]. Despite the performance improvement by considering the first-order local neighbors of each user, we argue that, for each user, instead of the interest diffusion from a user's neighbors to this user at one time, the social diffusion presents a dynamic recursive effect to influence each user's embedding. In detail, as the social influence propagation process begins (i.e., the diffusion iteration $k = 1$), each user's first latent embedding is influenced by the initial embeddings of her trusted connections. With the recursive influence diffuses over time, each user's latent embedding at k-th iteration is influenced by her trusted neighbors at the $(k-1)$-th iteration. Therefore, the social influence recursively propagates and diffuses in the social network. Correspondingly, each user's interests change in the recursive process. Precise simulating the recursive diffusion process in the global social network would better model each user's embedding, thus improve the social recommendation performance.

In this paper, we propose DiffNet: an Influence *Diff*usion neural network based model to stimulate the recursive social influence propagation process for better user and item embedding modeling in social recommendation. The key idea behind the proposed model is a carefully designed layer-wise influence diffusion structure for users, which models how users' latent embeddings evolve as the social diffusion process continues. Specifically, the diffusion process starts with an initial embedding for each user on top of the fusion of each user's features and a a free user latent vector that captures the latent behavior preference. For the item side, as items do not propagate in the social network, each item's embedding is also fused by the free item latent embedding and the item features. With the influence diffuses to a predefined K-th diffusion step, the K-th layer user interest embedding is obtained. In fact, with the learned user and item embedddings, DiffNet can be seamlessly incorporated into classical CF models, such as BPR and SVD++, and efficiently trained using SGD.

We summarize the contributions of this paper as follows:

- We propose a DiffNet model with a layer-wise influence propagation structure to model the recursive dynamic social diffusion in social recommendation. Besides, DiffNet has a fusion layer such that each user and each item could be represented as an embedding that encompasses both the collaborative and the feature content information.
- We show that the proposed DiffNet model is time and storage efficient in comparison to most embedding based recommendation models. The proposed model is a generalization of many related recommendation models and it is flexible when user and item attributes are not available.
- Experimental results on two real-world datasets clearly show the effectiveness of our proposed model. DiffNet outperforms more than **13.5**% on *Yelp* and **15.5**% on *Flickr* for top-10 recommendation compared to the the baselines with the best performance.

2 PRELIMINARIES

In a social recommender system, there are two sets of entities: a user set U ($|U| = M$), and an item set V ($|V| = N$). As the implicit feedback (e.g., watching an movie, purchasing an item, listening to a song) are more common in, we also consider the recommendation scenario with implicit feedback [32]. Let $\mathbf{R} \in \mathbb{R}^{M \times N}$ denote users' implicit feedback based rating matrix, with $r_{ai} = 1$ if user a is interested in item i, otherwise it equals 0. The social network can be represented as a user-user directed graph $\mathcal{G} = [U, \mathbf{S} \in \mathbb{R}^{M \times M}]$, with U is the user set and $\mathbf{S}$ represents the social connections between users. If user a trusts or follows user b, $s_{ba} = 1$, otherwise it equals 0. If the social network is undirected, then user a connects to user b denotes a follows b, and b also follows a, i.e., $s_{ab} = 1 \wedge s_{ba} = 1$. Then, each user a's ego social network, i.e., is the i-th column ($\mathbf{s}_a$) of $\mathbf{S}$. For notational convenience, we use S_a to denote the userset that a trusts, i.e., $S_a = [b|s_{ba} = 1]$.

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 $\mathbf{Y} \in \mathbb{R}^{d2 \times N}$. Then, the social recommendation problem can be defined as:

*Definition 2.1 (**SOCIAL RECOMMENDATION**).* Given a rating matrix $\mathbf{R}$, a social network $\mathbf{S}$, and associated real-valued feature matrix $\mathbf{X}$ and $\mathbf{Y}$ of users and items, our goal is to predict users' unknown preferences to items as: $\hat{R} = f(\mathbf{R}, \mathbf{S}, \mathbf{X}, \mathbf{Y})$, where $\hat{R} \in \mathbb{R}^{M \times N}$ denotes the predicted preferences of users to items.

Given the problem definition, we introduce the preliminaries that are closely related to our proposed model.

Classical Embedding Models. Given the user-item rating matrix $\mathbf{R}$, the latent embedding based models embed both users and items in a low 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[20, 31, 32]:

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

where $\mathbf{u}_a$ is the embedding of 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 $\mathbf{V}$.

SVD++ is an enhanced version of the latent factor based models that leveraged the rated history items of each user for better user embedding modeling [19]. In SVD++, each user's embedding is composed of a free embedding as classical latent factor based models, as well as an auxiliary embedding that is summarized from her rated items. Therefore, the predicted preference is modeled as:

$$\hat{r}_{ai} = \mathbf{v}_i^T (\mathbf{u}_a + \frac{1}{|R_a|} \sum_{j \in R_a} \mathbf{y}_j) \tag{2}$$

where $R_a = [j : r_{aj} = 1]$ is the itemset that a shows implicit feedback, and $\mathbf{y}_j$ is an implicit factor vector.

As sometimes users and items are associated with attributes, the feature enriched embedding models give the predicted preference $\hat{r}_{ai}$ of user a to item i is:

$$\hat{r}_{ai} = \mathbf{w}^T [\mathbf{x_a}, \mathbf{y_i}] + \mathbf{v}_i^T \mathbf{u}_a, \tag{3}$$

where the first term captures the bias terms with the feature engineering, and the second term models the second-order interaction between users and items.

Different embedding based models vary in the embedding matrix formulation and the optimization function. E.g., Bayesian Personalized Ranking (BPR) is one of the most successful pair-wise based optimization function for implicit feedback [32]. In BPR, it assumes that the embedding matrices $\mathbf{U}$ and $\mathbf{V}$ are free embeddings that follow a Gaussian prior, which is equivalent to adding a L2-norm regularization in the optimization function as:

$$\min_{[\mathbf{W},\mathbf{U},\mathbf{V}]} \mathcal{L} = \sum_{a=1}^{M} \sum_{(i,j)\in D_a} \sigma(\hat{r}_{ai} - \hat{r}_{aj}) + \lambda(||\mathbf{U}||_F^2 + ||\mathbf{V}||_F^2), \quad (4)$$

where $\sigma(x) = \frac{1}{1+exp(-x)}$ is a logistic function that transforms the input into range $(0, 1)$. $D_a = \{(i, j)|i \in R_a \wedge j \in V - R_a\}$ is the training data for a with R_a the itemset that a positively shows feedback, and $j \in V - R_a$ denotes the items that a does not show feedback in the training data.

Social Recommendation Models Social influence occurs when a persons's emotions, opinions or behaviors are affected by others [1, 24, 34]. In a social platform, social scientists have converged that social influence is a natural process for users to disseminate their preferences to the followers in the social network, such that the action (preference, interest) of a user changes with the influence from his/her trusted users [13, 17, 21, 22]. Therefore, as the social diffusion process continues, the *social correlation* phenomenon exits, with each user's preference and behavior are similar to her social connections [3, 22].

The social influence and social correlation among users' interests are the foundation for building social recommender systems [7, 25, 34]. 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 [15, 16, 25, 42] and the user behavior enhancement based approaches [7, 8]. Specifically, the social regularization based approaches assumed that connected users would show similar embeddings under the social influence diffusion. As such, besides the classical collaborative filtering based loss function (e.g, Eq.(4)), an additional social regularization term is incorporated in the overall optimization function as:

$$\sum_{i=1}^{M}\sum_{j=1}^{M} s_{ij}||\mathbf{u}_i - \mathbf{u}_j||_F^2 = \mathbf{U}(\mathbf{D} - \mathbf{S})\mathbf{U}^T, \quad (5)$$

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

Instead of the social regularization term, some researchers argued that the social network provides valuable information to enhance each user's behavior. TrustSVD is such a model that shows state-of-the-art performance [7, 8]. By assuming the implicit feedbacks of a user's social neighbors' on items could be regarded as the auxiliary feedback of this user, TrustSVD models the predicted preference as:

$$\hat{r}_{ai} = \mathbf{v}_i^T(\mathbf{u}_a + \sum_{b\in S_a} \frac{\mathbf{u}_b}{|S_a|}) \quad (6)$$

where $\mathbf{u}_b$ 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.

We notice that nearly all of the current social recommendation models leveraged the observed social connections (each user's social neighbors) for recommendation with a static process at once. However, the social influence is not a static but a recursive process, with each user is influenced by the social connections as time goes on. At each time, users need to balance their previous preferences with the influences from social neighbors to form their updated latent interests. Then, as the current user interest evolves, the influences from social neighbors changes. The process is recursively diffused in the social network. Therefore, current solutions neglected the iterative social diffusion process for social recommendation. What's worse, when the user features are available, these social recommendation models need to be redesigned to leverage the feature data for better correlation modeling between users [16]. In the following section, we would show how to tackle the above issues with our proposed DiffNet model.

3 THE PROPOSED MODEL

3.1 Model Architecture

We show the overall neural architecture of DiffNet in Fig 1. By taking an user-item pair $< a, i >$ as input, it outputs the probability $\hat{r}_{ai}$ that u would like item i. The overall neural architecture of DiffNet contains four main parts: the embedding layer, the fusion layer, the layer-wise influence diffusion layers, and the prediction layer. Specifically, by taking related inputs, the embedding layer outputs free embeddings of users and items. For each user (item), the fusion layer generates a hybrid user (item) embedding by fusing both a user's (an item's) free embedding and the associated features. The fused user embedding is then sent to the influence diffusion layers. The influence diffusion layers are built with a layer-wise structure to model the recursive social diffusion process in the social network, which is the key idea of the DiffNet. After the influence diffusion process reaches stable, the output layer generates the final predicted preference. We detail each part as follows:

Embedding Layer. Let $\mathbf{P} \in \mathbb{R}^{D\times M}$ and $\mathbf{Q} \in \mathbb{R}^{D\times N}$ represent the free embeddings of users and items. These free embeddings capture the collaborative latent representations of users and items. Given the one hot representations of user a and item i, the embedding layer performs an index operation and outputs the free user latent vector $\mathbf{p}_a$ and free item latent vector $\mathbf{q}_i$ from user free embedding matrix $\mathbf{P}$ and 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 $\mathbf{h}_a^0$ that captures the user's initial interests from different kinds of input data. We model the fusion layer as a one-layer fully connected neural network as:

$$\mathbf{h}_a^0 = g(\mathbf{W}^0 \times [\mathbf{x}_a, \mathbf{p}_a]), \quad (7)$$

where $\mathbf{W}^0$ is a transformation matrix, and $g(x)$ is a non-linear function. Without confusion, we omit the bias term in a fully-connected neural network for notational convenience. This fusion layer could generalize many typical fusion operations, such as the concatenation operation as $\mathbf{h}_a^0 = [\mathbf{x}_a, \mathbf{p}_a]$ by setting $\mathbf{W}^0$ as an identity matrix.

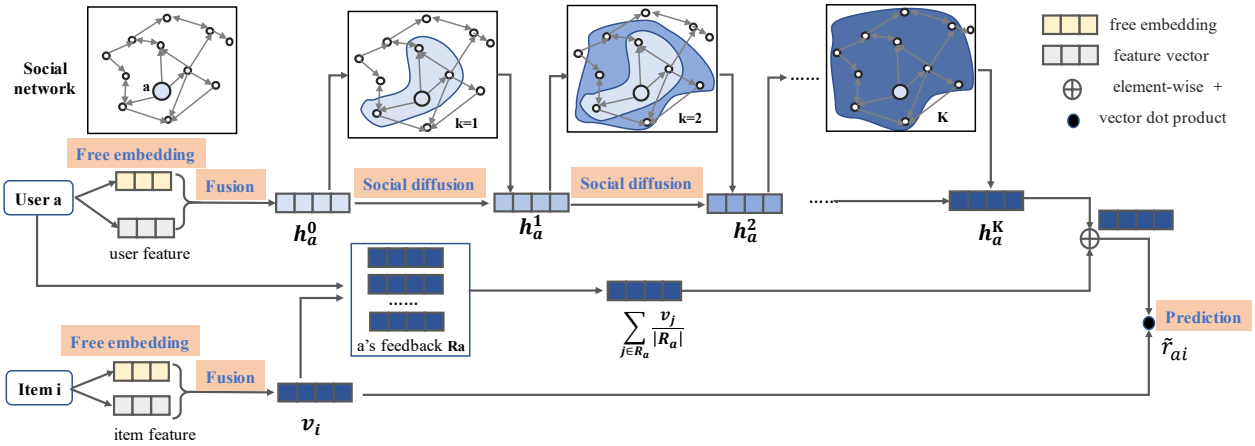


Figure 1: The overall architecture of our proposed model. The four parts of DiffNet are shown with orange background.

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

$$\mathbf{v}_i = \sigma(\mathbf{F} \times [\mathbf{q}_i, \mathbf{y}_i]). \tag{8}$$

Influence Diffusion Layers. By feeding the output of each user a's fusion embedding $\mathbf{h}_a^0$ from the fusion layer into the influence diffusion part, the influence diffusion layers model the dynamics of users' latent preference diffusion in the social network $\mathbf{S}$. As information diffuses in the social network from time to time, the influence diffusion part is analogously built with a multi-layer structure. Each layer k takes the users' embeddings from the previous layers as input, and output users' updated embeddings after the current social diffusion process finishes. Then, the updated user embeddings are sent to the $(k+1)$-th layer for the next diffusion process. This process is recursively operated until it the social diffusion process reaches stable.

For each user a, let $\mathbf{h}_a^k$ denote her latent embedding in the k-th layer of the influence diffusion part. By feeding the output of the k-th layer into the $(k+1)$-th layer, the influence diffusion operation at the $(k+1)$-th social diffusion layer updates each user a's latent embedding into $\mathbf{h}_a^{k+1}$. Specifically, the updated embedding $\mathbf{h}_a^{k+1}$ is composed of two steps: diffusion influence aggregation (AGG) from a's trusted users from the k-th layer, which transforms all the social trusted users' influences into a fixed length vector $\mathbf{h}_{Sa}^{k+1}$:

$$\mathbf{h}_{Sa}^{k+1} = Pool(\mathbf{h}_b^k | b \in S_a), \tag{9}$$

where the *Pool* function could be defined as an average pooling that performs a mean operation of all the trusted users' latent embedding at the k-th layer. The *Pool* can also be defined as a max operation that select the maximum element of all the trusted users' latent embedding at the k-th layer to form $\mathbf{h}_{Sa}^{k+1}$.

Then, a-th updated embedding $\mathbf{h}_a^{(k+1)}$ is a combination of her latent embedding $\mathbf{h}_a^k$ at the k-th layer and the influence diffusion embedding aggregation $\mathbf{h}_{Sa}^{k+1}$ from her trusted users. Since we do not know how each user balances these two parts, we use a non-linear neural network to model the combination as:

$$\mathbf{h}_a^{k+1} = s^{(k+1)}(\mathbf{W}^k \times [\mathbf{h}_{S_a}^{k+1}, \mathbf{h}_a^k]), \tag{10}$$

where $s^k(x)$ is non-linear transformation function.

With a predefined diffusion depth K, for each user a, the influence diffusion layer starts with the layer-0 user embedding $\mathbf{h}_a^0$ (Eq.(7)), i.e., the output of the fusion layer, and the layer-wise influence diffusion process then diffuses to layer 1, followed by layer 1 diffuses to layer 2. This influence diffusion step is repeated for K steps to reach the diffusion depth K, where each user a's latent embedding at the K-th layer is $\mathbf{h}_a^K$.

Please note that, DiffNet only diffuses users' latent vectors in the influence diffusion part without any item vector diffusion modeling. This is quite reasonable as item latent embeddings are static and do not propagate in the social network.

Prediction Layer. Given each user a's embedding $\mathbf{h}_a^K$ at the K-th layer after the iterative diffusion process, each item i's fusion vector $\mathbf{v}_i$, we model the predicted preference of user a to item i as:

$$\mathbf{u}_a = \mathbf{h}_a^K + \sum_{i \in R_a} \frac{\mathbf{v}_i}{|R_a|}, \tag{11}$$

$$\hat{r}_{ai} = \mathbf{v}_i^T \mathbf{u}_a, \tag{12}$$

where R_a is the itemset that a likes. In this equation, each user's final latent representation $\mathbf{u}_a$ is composed of two parts: the embeddings from the output of the social diffusion layers as $\mathbf{h}_a^K$, and the preferences from her historical behaviors as: $\sum_{i \in R_a} \frac{\mathbf{v}_i}{|R_a|}$. Specifically, the first term captures the user's interests from the recursive social diffusion process in the social network structure. The second term resembles the SVD++ model that leveraged the historical feedbacks of the user to alleviate the data sparsity of classical CF models [19], which has shown better performance over the classical latent factor based models. Thus, the final user embedding part is more representative with the recursive social diffusion modeling and the historical feedbacks of the user. After that, the final predicted rating is still measured by the inner produce between the corresponding user final latent vector and item latent vector.

3.2 Model Training

As we focus on implicit feedbacks of users, similar to the widely used ranking based loss function in BPR [32], we also design a pair-wise ranking based loss function for optimization:

$$\min_{\Theta} \mathcal{L}(\mathbf{R}, \hat{\mathbf{R}}) = \sum_{a=1}^{M} \sum_{(i,j)\in D_a} \sigma(\hat{r}_{ai} - \hat{r}_{aj}) + \lambda ||\Theta_1||^2 \tag{13}$$

where $\sigma(x)$ is a sigmoid function. $\Theta = [\Theta_1, \Theta_2]$, with $\Theta_1 = [\mathbf{P}, \mathbf{Q}]$, and $\Theta_2 = [\mathbf{F}, [\mathbf{W}^k]_{k=0}^{K-1}]$. λ is a regularization parameter that controls the complexity of user and item free embedding matrices. $D_a = \{(i,j)|i \in R_a \wedge j \in V - R_a\}$ denotes the pairwise training data for a with R_a represents the itemset that a positively shows feedback.

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. We split the mini-batch according to the userset, i.e., each user's training records are ensured in the same mini-batch. This mini-batch splitting procedure avoids the repeated computation of each user a's latent embedding $\mathbf{h}_a^K$ in the iterative influence diffusion layers.

As we could only observe positive feedbacks of users with huge missing unobserved values, similar as many implicit feedback works, for each positive feedback, we randomly sample 10 missing unobserved feedbacks as pseudo negative feedbacks at each iteration in the training process [39, 40]. As each iteration the pseudo negative samples change, each missing value gives very weak negative signal.

3.3 Discussion

3.3.1 Complexity. **Space complexity.** As shown in Eq.(13), the model parameters are composed of two parts: the user and item free embeddings $\Theta_1 = [\mathbf{P}, \mathbf{Q}]$, and the parameter set $\Theta_2 = [\mathbf{F}, [\mathbf{W}^k]_{k=0}^{K-1}]$. 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. 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. Since our proposed loss function resembles the BPR model with the pair-wise loss function that is designed for implicit feedback, we compare the time complexity of DiffNet with BPR. The main additional time cost lies in the layer-wise influence diffusion process. The dynamic diffusion process costs $O(MKL)$, where M is the number of users, and K denotes the diffusion depth and L denotes the average social neighbors of each user. Similarly, the additional time complexity of updating parameters is $O(MKL)$. Therefore, the additional time complexity is $O(MKL)$. In fact, as shown in the empirical findings as well as our experimental results, DiffNet reaches the best performance when $K = 2$. Also, the average social neighbors per user are limited with $L \ll M$. Therefore, the additional time complexity is acceptable and the proposed DiffNet could be applied to real-world social recommender systems.

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

3.3.2 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 utilized the social network structure $\mathbf{S}$ to model how users' latent preferences are dynamically influenced from the recursive social 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.(8), each item's latent embedding $\mathbf{v}_i$ degenerates to $\mathbf{q}_i$. Similarly, each user's initial layer-0 latent embedding $\mathbf{h}^0 = \mathbf{p}_a$ (Eq.(7)). Similarly, when either the user attributes or the item attributes do not exist, the corresponding fusion layer of user or item degenerates.

The key idea of our proposed model is the carefully designed social diffusion layers with the input social network $\mathbf{S}$. When the recommender system does not contain any social network information, the social diffusion layers disappear with $\mathbf{h}_a^K = \mathbf{h}_a^0$. Under this circumstances, as shown in Eq. (12) our proposed model degenerates to an enhanced SVD++ model [19] for recommendation, with the user and item latent embeddings contain the fused free embeddings and the associated user and item features.

3.3.3 Comparisons to Graph Convolutional based Models. In our proposed DiffNet, the designed layer-wise diffusion part (Eq.(9) and Eq.(10)) presents similar idea as the Graph Convolutional Networks (GCN), which are state-of-the-art representation learning techniques of graphs [11, 18, 36]. GCNs generate node embeddings in a recursive message passing or information diffusion manner of a graph, where the representation vector of a node is computed recursively from aggregation features in neighbor nodes. GCNs has shown theoretical elegance as simplified version of spectral based graph models [18]. Besides, GCNs are time efficient and achieve better performance in many graph-based tasks.

Due to the success of GCNs, several models have attempted to transfer the idea of GCNs for the recommendation tasks. By transferring these models to the recommendation scenario, the main components are how to construct a graph and further exploit the uniqueness of recommendation properties. Among them, the most closely related works are GC-MC [36] and PinSage [36].

GC-MC: It is one of the first few attempts that directly applied the graph convolutions for recommendation. GC-MC defines a user-item bipartite graph from user-item interaction behavior [36]. Then, each user embedding is convolved as the aggregation of the embeddings of her rated items. Similarly, each item embedding is convolved as the aggregation of the embeddings of the rated users' embeddings. However, the graph convolution is only operated with one layer of the observed links between users and items, neglecting the layer-wise diffusion structure of the graph.

PinSage: It is designed for similar item recommendation from a large recommender system. By constructing an item-item correlation graph from users' behaviors, a data-efficient GCN algorithm PinSage is developed [44]. PinSage could incorporate both the item correlation graph as well as node features to generate item embeddings. The main contribution lies in how to design efficient sampling techniques to speed up the training process. Instead of

Table 1: The statistics of the two datasets.

Dataset	*Yelp*	*Flickr*
Users	17237	8358
Items	38342	82120
Total Links	143765	187273
Ratings	204448	314809
Link Density	0.048%	0.268%
Rating Density	0.031%	0.046%

message passing on item-item graph, our work performs the recursive information diffusion of the social network, which is more realistic to reflect how users are dynamically influenced by the social influence diffusion. Applying GCNs for social recommendation is quite natural and to the best of our knowledge, has not been studied before.

4 EXPERIMENTS

In this section, we conduct experiments to evaluate the performance of DiffNet on two datasets.

4.1 Experimental Settings

Datasets. *Yelp* is an online location-based social network. Users make friends with others and express their experience through the form of reviews and ratings. As each user give ratings in the range $[0, 5]$, similar to many works, we transform the ratings that are larger than 3 as the liked items by this user. As the rich reviews are associated with users and items, we use the popular gensim tool[2] to learn the embedding of each word with Word2vec model [26]. Then, we get the feature vector of each user (item) by averaging all the learned word vectors of the user(item).

Flickr is a who-trust-whom online image based social sharing platform. Users follow other users and share their preferences to images to their social followers. Users express their preferences through the upvote behavior. For research purpose, we crawl a large dataset from this platform. Given each image, we have a ground truth classification of this image on the dataset. We send images to a VGG16 convolutional neural network and treat the 4096 dimensional representation in the last connected layer in VGG16 as the feature representation of the image [33]. For each user, her feature representation is the average of the image feature representations she liked in the training data.

In the data preprocessing step, for both datasets, we filtered out users that have less than 2 rating records and 2 social links, and removed the items which have been rated less than 2 times. We randomly select 10% of the data for the test. In the remaining 90% data, to tune the parameters, we select 10% from the training data as the validation set. The detailed statistics of the data after preprocessing is shown in Table 1.

Baselines and Evaluation Metrics. We compare DiffNet with various state-of-the-art baselines, including the classical pair-wise based recommendation model *BPR* [32], feature enhanced latent factor model *FM* [30], a state-of-the-art social recommendation model *TrustSVD* [7], a context-aware social recommendation model *ContextMF* that utilized the same input as our proposed model for recommendation [16]. Besides, we also compare our proposed model with two graph convolutional based recommendation models: *GC-MC* [36] and *PinSage* [44]. As the original PinSage focuses on generating high-quality embeddings of items, we generalize this model by constructing a user-item bipartite for recommendation [44]. Both of these two convolutional recommender models utilized the user-item bipartite and the associated features of users and items for recommendation.

As we focus on recommending top-N items for each user, we use two widely adopted ranking based metrics: Hit Ratio (HR) and Normalized Discounted Cumulative Gain(NDCG) [34]. Specifically, HR measures the number of items that the user likes in the test data that has been successfully predicted in the top-N ranking list. And NDCG considers the hit positions of the items and gives a higher score if the hit items in the top positions. For both metrics, the larger the values, the better the performance. Since there are too many unrated items, in order to reduce the computational cost, for each user, we randomly sample 1000 unrated items at each time and combine them with the positive items the user likes in the ranking process. We repeat this procedure 10 times and report the average ranking results.

Parameter Setting. For all the models that are based on the latent factor models, we initialize the latent vectors with small random values. In the model learning process, we use Adam as the optimizing method for all models that relied on the gradient descent based methods with an initial learning rate of 0.001. And the batch size is set as 512. In our proposed DiffNet model, we try the regularization parameter λ in the range $[0.0001, 0.001, 0.01, 0.1]$, and find $\lambda = 0.001$ reaches the best performance. For the aggregation function in Eq.(9), we have tried the max pooling and average pooling. We find the average pooling usually shows better performance. Hence, we set the average pooling as the aggregation function. Similar to many GCN models [18, 44], we set the depth parameter $K = 2$. With the user and item free embedding size D, in the fusion layer and the following influence diffusion layers, each layer's output is also set as D dimension. For the non-linear function $g(x)$ in the fusion layer, we use a sigmoid function that transforms each value into range $(0, 1)$. And we set the non linear functions of $[s^k(x)]_{k=0}^{K-1}$ with the ReLU function to avoid the vanishing gradient problem. After the training of each layer, we use batch normalization to avoid the internal covariate shift problem [14]. There are several parameters in the baselines, we tune all these parameters to ensure the best performance of the baselines for fair comparison. Please note that as generating user and item features are not the focus of our paper, we use the feature construction techniques as mentioned above. However, our proposed model can be seamlessly incorporated with more advanced feature engineering techniques.

4.2 Overall Comparison

In this section, we compare the overall performance of all models on two datasets. Specifically, Table 2 shows the HR@10 and NDCG@10 results for both datasets with varying latent dimension size D. Among all the baselines, BPR only considered the user-item rating information for recommendation, FM and TrustSVD improve over BPR by leveraging the node features and social network information. PinSage takes the same kind of input as FM and shows better performance than FM, showing the effectiveness of modeling the information passing of a graph. ContextMF is the baseline that uses the user and item features, as well as the social network

[2] https://radimrehurek.com/gensim/

Table 2: HR@10 and NDCG@10 comparisons for different dimension size D.

Models	*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	0.2443	0.2632	0.2617	0.1471	0.1575	0.155	0.0851	0.0832	0.0791	0.0679	0.0661	0.0625
SVD++	0.2581	0.2727	0.2831	0.1545	0.1632	0.1711	0.0821	0.0934	0.1054	0.0694	0.0722	0.0825
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
TrustSVD	0.2853	0.2880	0.2915	0.1704	0.1723	0.1738	0.1372	0.1367	0.1427	0.1062	0.1047	0.1085
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
GC-MC	0.2876	0.2902	0.2937	0.1657	0.1686	0.174	0.1123	0.1155	0.1182	0.0883	0.0945	0.0956
PinSage	0.2952	0.2958	0.3065	0.1758	0.1779	0.1868	0.1209	0.1227	0.1242	0.0952	0.0978	0.0991
DiffNet	**0.3366**	**0.3437**	**0.3477**	**0.2052**	**0.2095**	**0.2121**	**0.1575**	**0.1621**	**0.1641**	**0.1210**	**0.1231**	**0.1273**

Table 3: HR@N and NDCG@N comparisons for different top-N values.

Models	*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	0.1713	0.2632	0.3289	0.1243	0.1575	0.1773	0.0657	0.0851	0.1041	0.0607	0.0679	0.0737
SVD++	0.1868	0.2831	0.3492	0.1389	0.1711	0.1924	0.0827	0.1054	0.1257	0.0753	0.0825	0.0895
FM	0.1881	0.2835	0.3463	0.1359	0.1720	0.1895	0.0918	0.1233	0.1458	0.0845	0.0968	0.1046
TrustSVD	0.1906	0.2915	0.3693	0.1385	0.1738	0.1983	0.1072	0.1427	0.1741	0.0970	0.1085	0.1200
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
GC-MC	0.1932	0.2937	0.3652	0.1420	0.1740	0.1922	0.0897	0.1182	0.1392	0.0795	0.0956	0.1002
PinSage	0.2099	0.3065	0.3873	0.1536	0.1868	0.2130	0.0925	0.1242	0.1489	0.0842	0.0991	0.1036
DiffNet	**0.2276**	**0.3477**	**0.4232**	**0.1679**	**0.2121**	**0.2331**	**0.1210**	**0.1641**	**0.1952**	**0.1142**	**0.1273**	**0.1384**

structure. It performs better than most baselines. Our proposed model consistently outperforms ContextMF, showing the effectiveness of modeling the recursive social diffusion process in the social recommendation process.

When comparing the results of the two datasets, we observe that leveraging the social network structure and the social diffusion process contributes more on *Flickr* compared to *Yelp*. On both datasets, PinSAGE is the best baseline that leverages the node features without the social network information. E.g., DiffNet improves over PinSAGE about 13% on *Yelp*, and nearly 30% on *Flickr*. We guess a possible reason is that, the *Flickr* is a social based image sharing platform with a stronger social influence diffusion effect. In contrast, *Yelp* is a location based social network, and users' food and shopping preferences are not easily influenced in the social platform. Last but not least, we find the performance of all models does not increase as the latent dimension size D increases from 16 to 64. Some models reach the best performance when $D = 32$ (e.g., BPR) while other models reach the best performance when $D = 64$ (e.g., DiffNet). In the following experiment, we set the proper D for each model with the best performance in order to ensure fairness.

Table 3 shows the HR@N and NDCG@N on both datasets with varying top-N recommendation size N. From the results, we also find similar observations as Table 2, with our proposed model DiffNet always shows the best performance. Based on the overall experiment results, we could empirically conclude that our proposed DiffNet model outperforms all the baselines under different ranking metrics and different parameters.

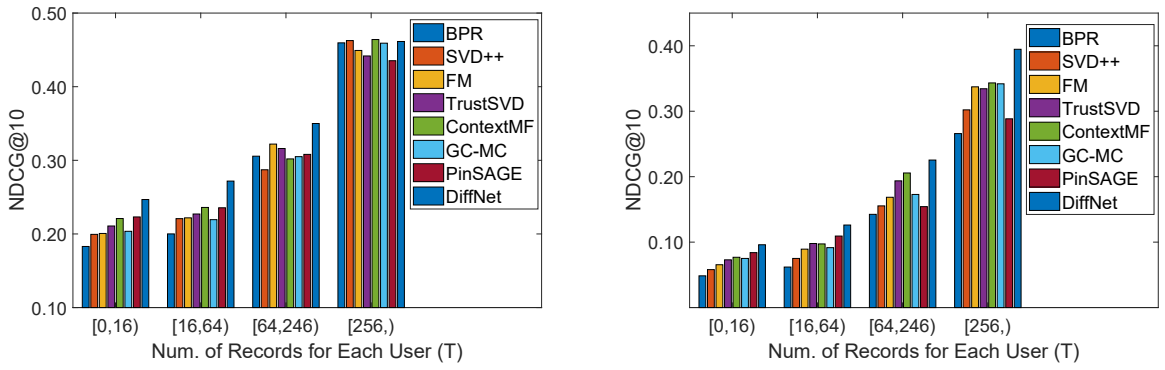
4.3 Performance under Different Data Sparsity

The data sparsity issue is a main challenge for most CF based recommender systems. In this part, we would like to show the performance of various models under different sparsity.

Specifically, we bin users into different groups based on the number of observed feedbacks in the training data. Then we show the performance of each group with different models on NDCG@10 in Fig 2. In this figure, the horizontal axis shows the user group information. E.g., $[16, 64)$ means for each user in this group, the training records satisfy $16 \leq |R_a| < 64$. As can be observed from this figure, for both datasets, with the increase of the user rating records, the performance increases quickly for all models. When the rating records of each user is less than 16, the BPR baseline could not work well as this model only exploited the very sparse user-item interaction behavior for recommendation. Under this situation, all improvement is significant by leveraging various kinds of side information. E.g., FM, SVD++, and ContextMF improves over BPR by 9.6%, 15.2% and 20.7% on Yelp, and 34.9%, 50.3%, 58.4% on Flickr. The improvement of Flick is much more significant than that of Yelp, as Flickr has much more items compared to Yelp. By considering the iteratively social diffusion process in social recommendation, our proposed model improves BPR by 34.8% and 97.1% on Flick and Yelp, which far exceeds the remaining models. With the increase of user rating records, the performance improvements of all models over BPR decrease, but the overall trend is that all models have better performance than BPR. We also observe that when users have more than 256 records, some methods have similar results as BPR or even a little worse than BPR. We guess a possible reason is that BPR could well learn the user interests from enough interaction data. Withe the additional side information, some noises are introduced to decrease the performance.

4.4 Detailed Model Analysis

We would analyze the recursive social diffusion depth K, and the impact of the fusion layer that combines the collaborative free embedding and associated entity feature vector.



(a) Yelp (b) Flickr

Figure 2: Performance under different sparsity (Better viewed in color.)

Table 4: HR@10 and NDCG@10 performance with different diffusion depth K.

Diffusion Depth K	*Yelp*				*Flickr*			
	HR	Improve.	NDCG	Improve.	HR	Improve.	NDCG	Improve.
K=2	**0.3477**	-	**0.2121**	-	**0.1641**	-	**0.1273**	-
K=0	0.3145	-9.54%	0.2014	-5.09%	0.1439	-12.27%	0.1148	-10.0%
K=1	0.3390	-2.50%	0.2981	-1.93%	0.1592	-2.96%	0.1257	-1.22%
K=3	0.3348	-3.72%	0.2005	-5.49%	0.1603	-2.34%	0.1246	-2.22%

Table 5: HR@10 and NDCG@10 of our simplified models on *Yelp* and *Flickr* with different fusion inputs. X=Y=0 denotes the user and item feature vector are not available. P=0 (Q=0) denotes we do not add the free base user (item) latent embedding.

Simplified models	*Yelp*				*Flickr*			
	HR	Improve.	NDCG	Improve.	HR	Improve.	NDCG	Improve.
DiffNet	**0.3477**	-	**0.2121**	-	**0.1641**	-	**0.1273**	-
$\mathbf{X} = 0$	0.3403	-2.11%	0.2072	-2.32%	0.1582	-3.58%	0.1232	-3.25%
$\mathbf{Y} = 0$	0.3271	-5.92%	0.1951	-8.06%	0.1423	-13.26%	0.1098	-13.73%
$\mathbf{X} = \mathbf{Y} = 0$	0.3196	-8.09%	0.1912	-9.89%	0.1360	-17.08%	0.1073	-15.69%
$\mathbf{P} = 0$	0.2461	-29.22%	0.1569	-26.05%	0.1056	-35.66%	0.0863	-32.17%
$\mathbf{Q} = 0$	0.1975	-43.19%	0.658	-69.00%	0.0334	-78.78%	0.022	-82.13%

Table 4 shows the results on DiffNet with different K values. The column of "Improve" show the performance changes compared to the best setting of DiffNet, i.e., $K = 2$. When $K = 0$, the layer-wise diffusion part disappears, and our proposed model degenerates to an enhanced SVD++ with entity feature modeling. As can be observed from this figure, as we leverage the layer wise diffusion process from $K = 0$ to $K = 1$, the performance increases quickly for for both datasets. For both datasets, the best performance reaches with two recursive diffusion depth, i.e., $K = 2$. When K continues to increase to 3, the performance drops for both datasets. We hypothesis that, considering the K-step recursive social diffusion process resembles the k-hop neighbors of each user. Since the social diffusion diminishes with time and the distance between each user and the k-hop neighbors, setting K with 2 is enough for social recommendation. In fact, other related studies have empirically find similar trends, with the best diffusion size is set as $K = 2$ or $K = 3$ [18, 44].

Table 4 shows the performance on DiffNet with different fusion inputs. As can be seen from this figure, the performance drops when the user and (or) item features are not available. We also notice that it is very important to add the free latent embeddings of users and items in the modeling process. As can be observed from this figure, the performance drops very quickly when either the user free latent embedding matrix $\mathbf{P}$ or the item free embedding matrix $\mathbf{Q}$ are not considered. E.g., the performance drops about 80% on Flickr when the item free embedding is not considered. The reason is that, the item (user) latent factors could not be captured by the item (user) features. Therefore, learning the collaborative effect between users and items with the free embeddings is very important for the recommendation task.

5 RELATED WORK

Collaborative Filtering. Given an user-item rating matrix $\mathbf{R}$, CF usually projected both users and items in a same low latent space for comparison [20]. In reality, compared to the explicit ratings, it is more common for users implicitly express their feedbacks through action or inaction, such as click, add to cart or consumption [32]. Bayesian Personalized Ranking (BPR) is a state-of-the-art latent factor based technique for dealing with implicit feedback. Instead

of directly predicting each user's point-wise explicit ratings, BPR modeled the pair-wise preferences with the assumption that users prefer the observed implicit feedbacks compared to the unobserved ones [32]. Despite the relatively high performance, the data sparsity issue is a barrier to the performance of these collaborative filtering models. To tackle the data sparsity issue, many models have been proposed by extending these classical CF models. E.g., SVD++ is proposed to combine users' implicit feedbacks and explicit feedbacks for modeling users' latent interests [19]. Besides, as users and items are associated with rich attributes, Factorization Machine (FM) is such a unified model that leverages the user and item attributes in latent factor based models [37]. Recently, some deep learning based models have been proposed to tackle the CF problem [12]. As sometimes the user and item features are sparse, many deep learning based models have been proposed to tackle how to model these sparse features [9, 37]. In contrast to these works, we do not consider the scenario of sparse features and put emphasis on how to model the recursive social diffusion process for social recommendation.

Social Recommendation Social recommendation leverages the social network among users to enhance recommendation performance [7, 16, 25]. In fact, social scientists have long converged that as information diffuses in the social networks, users are influenced by their social connections with the social influence theory, leading to the phenomenon of similar preferences among social neighbors [3, 4, 23, 29]. Social regularization has been empirically proven effective for social recommendation, with the assumption that similar users would share similar latent preferences under the popular latent factor based models [15, 25, 42]. SBPR model is proposed into the pair-wise BPR model with the assumption that users tend to assign higher ratings to the items their friends prefer [45]. By treating the social neighbors' preferences as the auxiliary implicit feedbacks of an active user, TrustSVD [7, 8] is proposed to incorporate the trust influence from social neighbors on top of SVD++ [19]. 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 [16]. Social recommendation has also been extended with social circles [28], temporal context [34], and visual item information [38]. Instead of simply considering the local social neighbors of each user, our work differs from these works in explicitly modeling the recursive social diffusion process to better model each user's latent preference in the global social network.

Graph Convolutional Networks and Applications. Our proposed model with recursive social diffusion process borrows the recent advances of graph convolutional networks (GCN) [11, 18, 36]. GCNs have shown success to extend the convolution operation from the regular Euclidean domains to non-Euclidean graph domains. Spectral graph convolutional neural network based approaches provide localized convolutions in the spectral domain [5, 6]. These spectral models usually handle the whole graph simultaneously, and are difficult to parallel or scale to large graphs. Recently, Kipf et al. designed a graph convolutional network (GCN) for semi-supervised learning on graph data, which can be motivated based on the spectral graph convolutional networks [10, 11, 18, 36]. The key idea of GCNs is to generate node embeddings in a message passing or information diffusion manner of a graph, which advanced previous spectral based models with much less computational cost.

Researchers also exploited the possibility of applying spectral models and GCNs to recommender systems. As in the collaborative setting, the user-item interaction could be defined as a bipartite graph, some works adopted the spectral graph theory for recommendation [27, 46]. Nevertheless, these models showed high computational cost and it is non-trivial to incorporate user (item) features in the modeling process. As GCNs showed improved efficiency and effectiveness over the spectral models [18], a few research works exploited GCNs for recommendation [36, 43, 44]. These models all share the commonality of applying the graph convolution operation that aggregates the information of the graph's first-order connections. GC-MC is one of the first few attempts that directly applied the graph convolutions on the user-item rating graph [36]. However, the graph convolution is only operated with one layer of the observed links between users and items, neglecting the higher order structure of the graph. GCMC is proposed for bipartite edge prediction with inputs of user-item interaction matrix [43]. This model is consisted of two steps: constructing a user-user graph and item-item graph from the user-item interaction matrix, then updating user and item vectors are based on the convolutional operations of the constructed graphs. Hence, the performance of GCMC relies heavily on the user-user and item-item construction process, and the two step process is not flexible compared to the end-to-end training process. By constructing an item correlation graph, researchers developed a data-efficient GCN algorithm PinSage, which combines efficient random walks and graph convolutions to generate embeddings of nodes that incorporate both graph structure as well as node feature information [44]. Our work differs from them as we leverage the graph convolution operation for the recursive social diffusion in the social networks, which is quite natural. Besides, our proposed model is general and could be applied when the user (item) attributes or the social network structure is not available.

6 CONCLUSIONS

In this paper, we proposed a DiffNet neural model for social recommendation. Our main contribution lies in designing a layer-wise influence diffusion part to model how users' latent preferences are recursively influenced by the her trusted users. We showed that the proposed DiffNet model is time and storage efficient. It is also flexible when the user and item attributes are not available. The experimental results clearly showed the flexibility and effectiveness of our proposed models. E.g., DiffNet improves more than 15% over the best baseline with NDCG@10 on *Flickr* dataset. In the future, we would like to extend our model for temporal social recommendation, where the temporal changes of users' interests are implicitly reflected from their temporal feedback patterns.

ACKNOWLEDGEMENTS

This work was supported in part by grants from the National Natural Science Foundation of China(Grant No. 61725203, 61722204, 61602147, 61732008, 61632007), the Anhui Provincial Natural Science Foundation(Grant No. 1708085QF155), and the Fundamental Research Funds for the Central Universities(Grant No. JZ2018HGTB0230).

REFERENCES

[1] [n. d.]. Wikipedia explanation of social influence. https://en.wikipedia.org/wiki/Social_influence. Accessed Jan 10, 2019.

[2] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *TKDE* 17, 6 (2005), 734–749.

[3] Aris Anagnostopoulos, Ravi Kumar, and Mohammad Mahdian. 2008. Influence and correlation in social networks. In *SIGKDD*. 7–15.

[4] Robert M Bond, Christopher J Fariss, Jason J Jones, Adam DI Kramer, Cameron Marlow, Jaime E Settle, and James H Fowler. 2012. A 61-million-person experiment in social influence and political mobilization. *Nature* 489, 7415 (2012), 295–298.

[5] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. 2014. Spectral networks and locally connected networks on graphs. In *ICLR*.

[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*. 3844–3852.

[7] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. 2015. TrustSVD: Collaborative Filtering with Both the Explicit and Implicit Influence of User Trust and of Item Ratings.. In *AAAI*. 123–125.

[8] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. 2016. A novel recommendation model regularized with user trust and item ratings. *TKDE* 28, 7 (2016), 1607–1620.

[9] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. In *IJCAI*. 1725–1731.

[10] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*. 1024–1034.

[11] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *TCDE* 40, 3 (2017), 52–74.

[12] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

[13] Herminia Ibarra and Steven B Andrews. 1993. Power, social influence, and sense making: Effects of network centrality and proximity on employee perceptions. *ASQ* 38, 2 (1993), 277–303.

[14] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML* (2015), 448–456.

[15] Mohsen Jamali and Martin Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In *RecSys*. 135–142.

[16] Meng Jiang, Peng Cui, Fei Wang, Wenwu Zhu, and Shiqiang Yang. 2014. Scalable recommendation with social contextual information. *TKDE* 26, 11 (2014), 2789–2802.

[17] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *SIGKDD*. ACM, 137–146.

[18] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[19] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *SIGKDD*. 426–434.

[20] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[21] Adam DI Kramer, Jamie E Guillory, and Jeffrey T Hancock. 2014. Experimental evidence of massive-scale emotional contagion through social networks. *PNAS* 111, 24 (2014), 8788–8790.

[22] Kevin Lewis, Marco Gonzalez, and Jason Kaufman. 2012. Social selection and peer influence in an online social network. *PNAS* 109, 1 (2012), 68–72.

[23] Qi Liu, Enhong Chen, Hui Xiong, Chris HQ Ding, and Jian Chen. 2011. Enhancing collaborative filtering by user interest expansion via personalized ranking. *TKDE* 42, 1 (2011), 218–233.

[24] Qi Liu, Biao Xiang, Enhong Chen, Hui Xiong, Fangshuang Tang, and Jeffrey Xu Yu. 2014. Influence maximization over large-scale social networks: A bounded linear approach. In *CIKM*. ACM, 171–180.

[25] Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. 2011. Recommender systems with social regularization. In *WSDM*. 287–296.

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. 3111–3119.

[27] Federico Monti, Michael Bronstein, and Xavier Bresson. 2017. Geometric matrix completion with recurrent multi-graph neural networks. In *NIPS*. 3697–3707.

[28] Xueming Qian, He Feng, Guoshuai Zhao, and Tao Mei. 2014. Personalized recommendation combining user interest and social circle. *TKDE* 26, 7 (2014), 1763–1777.

[29] JZ Qiu, Jian Tang, Hao Ma, YX Dong, KS Wang, and J Tang. 2018. DeepInf: Modeling influence locality in large social networks. In *SIGKDD*. 2110–2119.

[30] Steffen Rendle. 2010. Factorization machines. In *ICDM*. 995–1000.

[31] Steffen Rendle. 2012. Factorization machines with libfm. *TIST* 3, 3 (2012), 57.

[32] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.

[33] Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *ICLR*.

[34] Peijie Sun, Le Wu, and Meng Wang. 2018. Attentive Recurrent Social Recommendation. In *SIGIR*. 185–194.

[35] Jiliang Tang, Xia Hu, and Huan Liu. 2013. Social recommendation: a review. *SNAM* 3, 4 (2013), 1113–1133.

[36] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph Convolutional Matrix Completion. *ICLR* (2017).

[37] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *SIGKDD*. ACM, 1235–1244.

[38] Le Wu, Lei Chen, Richang Hong, Yanjie Fu, Xing Xie, and Meng Wang. 2019. A Hierarchical Attention Model for Social Contextual Image Recommendation. *TKDE* (2019).

[39] Le Wu, Yong Ge, Qi Liu, Enhong Chen, Richang Hong, Junping Du, and Meng Wang. 2017. Modeling the evolution of usersâĂŹ preferences and social links in social networking services. *TKDE* 29, 6 (2017), 1240–1253.

[40] Le Wu, Yong Ge, Qi Liu, Enhong Chen, Bai Long, and Zhenya Huang. 2016. Modeling Users' Preferences and Social Links in Social Networking Services: a Joint-Evolving Perspective. In *AAAI*. 279–286.

[41] Le Wu, Qi Liu, Enhong Chen, Nicholas Jing Yuan, Guangming Guo, and Xing Xie. 2016. Relevance meets coverage: A unified framework to generate diversified recommendations. *TIST* 7, 3 (2016), 39.

[42] Le Wu, Peijie Sun, Richang Hong, Yong Ge, and Meng Wang. 2019. Collaborative Neural Social Recommendation. *TSMC: Systems* (2019).

[43] Yuexin Wu, Hanxiao Liu, and Yiming Yang. 2018. Graph Convolutional Matrix Completion for Bipartite Edge Prediction. In *KDIR*. 51–60.

[44] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *SIGKDD*. 974–983.

[45] Tong Zhao, Julian McAuley, and Irwin King. 2014. Leveraging social connections to improve personalized ranking for collaborative filtering. In *CIKM*. 261–270.

[46] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S Yu. 2018. Spectral collaborative filtering. In *RecSys*. 311–319.