A Graph Neural Network Framework for Social Recommendations

Wenqi Fan, Yao Ma, Qing Li, Jianping Wang, Jiliang Tang, and Dawei Yin

Abstract—Nowadays, most data about social networks, users shopping behaviors, and inter-item relationships etc., can be represented as graph structure. Graph Neural Networks (GNNs) have shown great success in learning meaningful representations for graph data by naturally integrating node information and topological structures. Data used in making social recommendations can also be represented as graph data in the form of user-user social graphs and user-item graphs. In addition, the relationships between items can be represented as graph data, denoted as item-item graph. GNNs provide an unprecedented opportunity to advance social recommendations, yet there are considerable challenges in building GNNs-based social recommendations based on this modelling framework in that (1) users (items) are simultaneously involved in the user-item graph and user-user social graph (item-item graph); (2) user-item graphs not only contain user-item interactions but also include users' opinions on items; and (3) the nature of social relations are heterogeneous among users. In this paper, we propose a novel graph neural network framework (**GraphRec+**) for social recommendations, which is able to coherently model graph data in order to learn better user and item representations. Specifically, we introduce a principled approach for jointly capturing interactions and opinions in the user-item graph and also propose an attention mechanism to differentiate the heterogeneous strengths of social relations. Comprehensive experiments on three real-world datasets show the effectiveness of the proposed framework.

Index Terms—Social Recommendation; Graph Neural Networks; Recommender Systems; Collaborative Filtering; Social Network; Neural Networks

1 Introduction

In this era of information proliferation, tools are required to assist users in filtering relevant information from very noisy data [1], [2]. Recommender systems are an effective way of mitigating the information overload problem, especially in many user-oriented online services, such as e-commerce (Amazon, Taobao), and social media (Facebook, Instagram) sites [3]. The goal of recommender systems is to suggest to the user a personalized list of items that are likely to be clicked on or purchased [3], [4]. Among existing techniques used in modern recommender systems, collaborative filtering (CF) is one of the most popular techniques used to model users' preference for items by utilizing the history of user-item interactions [2], [5].

In addition to the user-item interactions, integrating social relations into recommender systems has attracted a lot of attention in recent years [6], [7], [8]. These social recommender systems have been developed based on the phenomenon of *word-of-mouth marketing*, which is widely recognized as the most effective strategy for item recommendations. Specifically, users are able to acquire and disseminate information through people around them, such as classmates, friends, relatives, or colleagues. As such, the underlying social relations of users can play an important role in helping them filter information and in profiling users' preferences towards items [9]. Therefore, utilizing users' social relations has been used to greatly boost the performance of many recommender systems [4], [10].

In recent years, deep neural networks have achieved great success when applied to graph data [11]. These deep neural network architectures, known as Graph Neural Networks (GNNs), are proposed to learn meaningful representations from graph data based on node features and the graph structure [12], [13], [14], [15]. In particular, the main ideas are to iteratively aggregate feature information from local graph neighborhoods by using deep neural networks. Meanwhile, node information can be propagated through the graph structure after transformation and aggregation operations. Hence, GNNs naturally integrate the node information as well as the topological structure, and have been demonstrated as powerful methods for representation learning applied to graph data [11], [13], [14]. GNNs have been applied to wide variety of applications, including knowledge graphs [16], [17], computer vision [18], [19], and text analysis [20].

On the other hand, data in social recommendations can be naturally represented as graph data with two explicit graphs, one as a user-item graph, for denoting interactions between users and items (see Figure 1 (a)), and the other as a social graph for denoting the relationships between users (see Figure 1 (b)). These two types of user relationships can help infer better user preferences from different perspectives.

In addition to these explicit relationships in social recommendations, the relationships between items are also important for providing another stream of information that

- *W. Fan and J. Wang are with the Department of Computer Science, City University of Hong Kong, Hong Kong. E-mail: wenqifan03@gmail.com, jianwang@cityu.edu.hk.*
- *Y. Ma and J. Tang are with Data Science and Engineering (DSE) Lab, Michigan State University, USA. E-mail: mayao4@msu.edu, tangjili@msu.edu*
- *Q. Li is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. E-mail: csqli@comp.polyu.edu.hk*
- *D. Yin is with Data Science Lab, JD.com. E-mail: yindawei@acm.org.*

Manuscript received April 19, 2005; revised August 26, 2015.

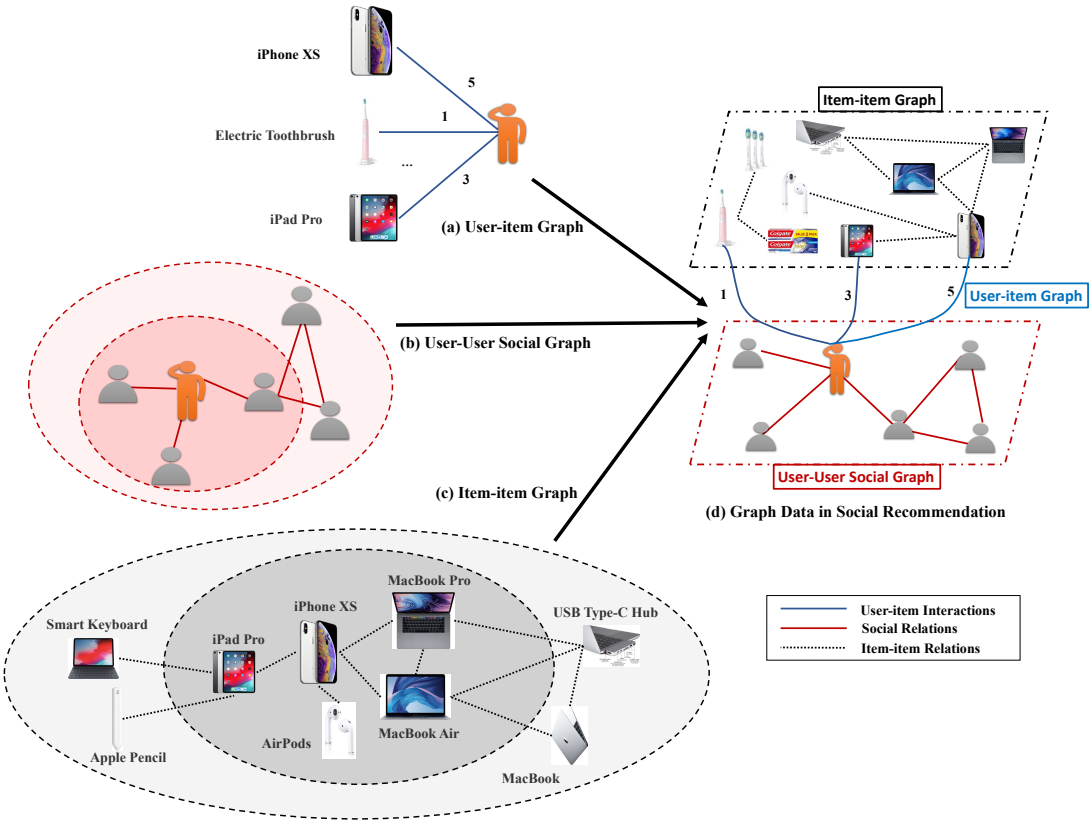


Figure 1: Graph Data in Social Recommendations. It contains three graphs including the user-item graph (a), the user-user social graph (b), and the item-item graph (c). Note that the number on the edges of the user-item graph denotes the opinions (or rating score) of the users on the items via the interactions.

can potentially be used to profile items. That is because items are not independent and are likely to be similar or related [21], [22]. For example, users who bought an Apple iPhone XS are likely to consider purchasing the Apple AirPods because they are associated with the same attribute (i.e., designed by Apple Inc.). Thus, when a user plans to purchase a notebook with MacOS, *Macbook Air, Macbook and Macbook Pro,* these items are bundled together for comparison because of their similar functionality. In fact, the relations between items have already facilitated many important applications, such as navigation between related items, discovery of new or previously unknown items, and identification of interesting item combinations. Specifically, Amazon allows users to navigate between items through relations [21], such as 'users who bought X also bought Y' and 'users who viewed X also viewed Y'. Apparently, these relations between items can also be represented as a graph, denoted as an item-item graph, as shown in Figure 1 (c). Therefore, it is desirable to consider the relations among items for enhancing the representation learning of items in social recommendations.

With these three graphs, data in social recommendations can be represented as a heterogeneous graph, as shown in Figure 1 (d). Note that users (who can be seen as bridges) are simultaneously involved in both explicit graphs (bridging the user-item graph and social graph), while items bridge the user-item graph and item-item graph. Moreover, instead of learning representations of items and users from the user-item graph only, it is important to incorporate the social graph into the process of learning better user representations, and to integrate the item-item graph into the process of learning better item representations. Learning representations of items and users are the key to building social recommender systems. Thus, given their advantages, GNNs provide unprecedented opportunities to advance social recommendations.

Nevertheless, building social recommender systems based on GNNs faces challenges. The social graph and the user-item graph in a social recommender system provide information about users from different perspectives, while the item-item graph and user-item graph contain valuable signals that are beneficial to infer the profiles of items. It is important to aggregate information from these three graphs to learn better user and item representations from different perspectives. Thus, the first challenge is how to inherently combine these graphs. Moreover, the user-item graph not only contains interactions between users and items, but also includes users' opinions on items. These opinions can be explicitly given by the user as a rating score, generally within a certain numerical scale (e.g., 5-star). For example, as shown in Figure 1 (a), the user interacts with the items of "iPhone XS" and "Electric Toothbrush", and the user likes the "iPhone XS" while disliking the "Electric Toothbrush". These items' opinions given by users provide valuable signals associated with both the users and items, since these opinions can capture users preferences on items while reflecting the varying characteristics of items, as provided by the users. Therefore,

the second challenge is how to capture interactions and opinions between users and items jointly. In addition, the low cost of link information in online worlds can result in networks with varied tie strengths (e.g., strong and weak ties are mixed together) [23], with users being more likely to share similar tastes among strong ties than weak ties. Considering social relations equally may be insufficient to model the complicated relationships between users, and could lead to degradation in recommendation performance. Hence, the third challenge is how to distinguish social relations with heterogeneous strengths.

In this paper, to tackle these challenges simultaneously, we aim to build social recommender systems based on graph neural networks. Specifically, we propose a novel graph neural network **GraphRec+** for social recommendations, which is an improved model of the existing GraphRec [24]. Our major contributions are summarized as follows:

- We propose a novel graph neural network GraphRec+, which can coherently model graph data to learn better user and item representations in social recommendations.
- We provide a principled approach to jointly capture interactions and opinions in the user-item graph.
- We introduce a method to consider heterogeneous strengths of social relations mathematically.
- We demonstrate the effectiveness of the proposed framework on various real-world datasets.

The remainder of this article is organized as follows. We introduce the proposed framework in Section 2. In Section 3, we conduct experiments on three real-world datasets to illustrate the effectiveness of the proposed method. In section 4, we review existing research related to our work. Finally, we conclude our work with future directions in Section 5.

2 The Proposed Framework

In this section, we first introduce the definitions and notations used in this paper, then give an overview of the proposed framework and detail each of the model components. Finally, we discuss how the model parameters are learned.

2.1 Definitions and Notations

Let $U = \{u_1, u_2, ..., u_n\}$ and $V = \{v_1, v_2, ..., v_m\}$ be the sets of users and items, respectively, where n is the number of users, and m is the number of items. We assume that $\mathbf{R} \in \mathbb{R}^{n \times m}$ is the user-item rating matrix, which is also called the user-item graph. If u_i gives a rating to v_j, r_{ij} is the rating score; otherwise we employ 0 to represent the unknown rating from u_i to v_j, i.e., $r_{ij} = 0$. The observed rating score r_{ij} can be seen as user u_i's opinion on the item v_j. Let $\mathcal{O} = \{\langle u_i, v_j \rangle | r_{ij} \neq 0\}$ be the set of known ratings and $\mathcal{T} = \{\langle u_i, v_j \rangle | r_{ij} = 0\}$ be the set of unknown ratings.

In addition, users can establish social relations with each other. We use $\mathrm{T} \in \mathbb{R}^{n \times n}$ to denote the user-user social graph, where $\mathrm{T}_{ij} = 1$ if u_j has a relation to u_i and zero otherwise. Let $N(i)$ be the set of users whom u_i is directly connected with in the user-user social graph T, $C(i)$ be the set of items with which u_i has interacted with in $\mathbf{R}$, and $B(j)$ be the set of users who have interacted with v_j in $\mathbf{R}$. Meanwhile, the item-item graph is denoted as $\mathrm{I}_G = (V, E_V)$, where V is the set of items and E_V is the set of edges connecting related items. $M(j)$ denotes the set of items that are similar/related to item j.

Table 1: Notation

Symbols	Definitions and Descriptions
r_{ij}	the rating value of item v_j by user u_i
$\mathbf{q}_j$	the embedding of item v_j
$\mathbf{p}_i$	the embedding of user u_i
$\mathbf{e}_r$	the opinion embedding for the rating level r, such as 5-star rating, $r \in \{1, 2, 3, 4, 5\}$
d	the length of embedding vector
$C(i)$	the set of items which user u_i interacted with
$N(i)$	the set of social friends who user u_i directly connected with
$B(j)$	the set of users who have interacted the item v_j
$M(j)$	the set of items which are similar/related to the item v_j
$\mathbf{h}_i^I$	the item-space user latent factor from item set $C(i)$ of user u_i
$\mathbf{h}_i^S$	the social-space user latent factor from the social friends $N(i)$ of user u_i
$\mathbf{h}_i$	the user latent factor of user u_i, combining from item space $\mathbf{h}_i^I$ and social space $\mathbf{h}_i^S$
$\mathbf{h}_j^U$	the user-space item latent factor from user set $B(j)$ of item v_j
$\mathbf{h}_j^V$	the item2item-space item latent factor from the similar/related items set $M(j)$ of item v_j
$\mathbf{z}_j$	The item latent factor of item v_j
$\mathbf{x}_{ia}$	the opinion-aware interaction representation of item v_a for user u_i
$\mathbf{f}_{jt}$	the opinion-aware interaction representation of user u_t for item v_j
α_{ia}	the item attention of item v_a in contributing to $\mathbf{h}_i^I$
β_{io}	the social attention of neighboring user u_o in contributing to $\mathbf{h}_i^S$
μ_{jt}	the user attention of user u_t in contributing to $\mathbf{h}_j^U$
κ_{jk}	the item attention of item v_t in contributing to $\mathbf{h}_j^V$
r'_{ij}	the predicted rating value of item v_j by user u_i
$\oplus$	the concatenation operator of two vectors
T	the user-user social graph
I_G	the item-item graph
$\mathbf{R}$	the user-item rating matrix (user-item graph)
$\mathbf{W}, \mathbf{b}$	the weight and bias in neural network

Given the user-item graph $\mathbf{R}$, social graph T, and item-item graph I_G, we aim to predict the missing rating value in $\mathbf{R}$. Following [3], [25], we use an embedding vector $\mathbf{p}_i \in \mathbb{R}^d$ to denote a user u_i, and an embedding vector $\mathbf{q}_j \in \mathbb{R}^d$ to represent an item v_j, where d is the length of embedding vector. More details about these embedding vectors will be provided in the following subsections. The mathematical notations used in this paper are summarized in Table 1.

2.2 An Overview of the Proposed Framework

The architecture of the proposed model is shown in Figure 2. The model consists of three main components: user modeling, item modeling, and rating prediction.

The first component, user modeling aims to learn latent factors of the users. As users in social recommender systems are simultaneously involved in two different graphs, i.e., a social graph and a user-item graph, we are provided with a great opportunity to learn user representations from different perspectives. Therefore, two aggregations are introduced to process these two different graphs respec-

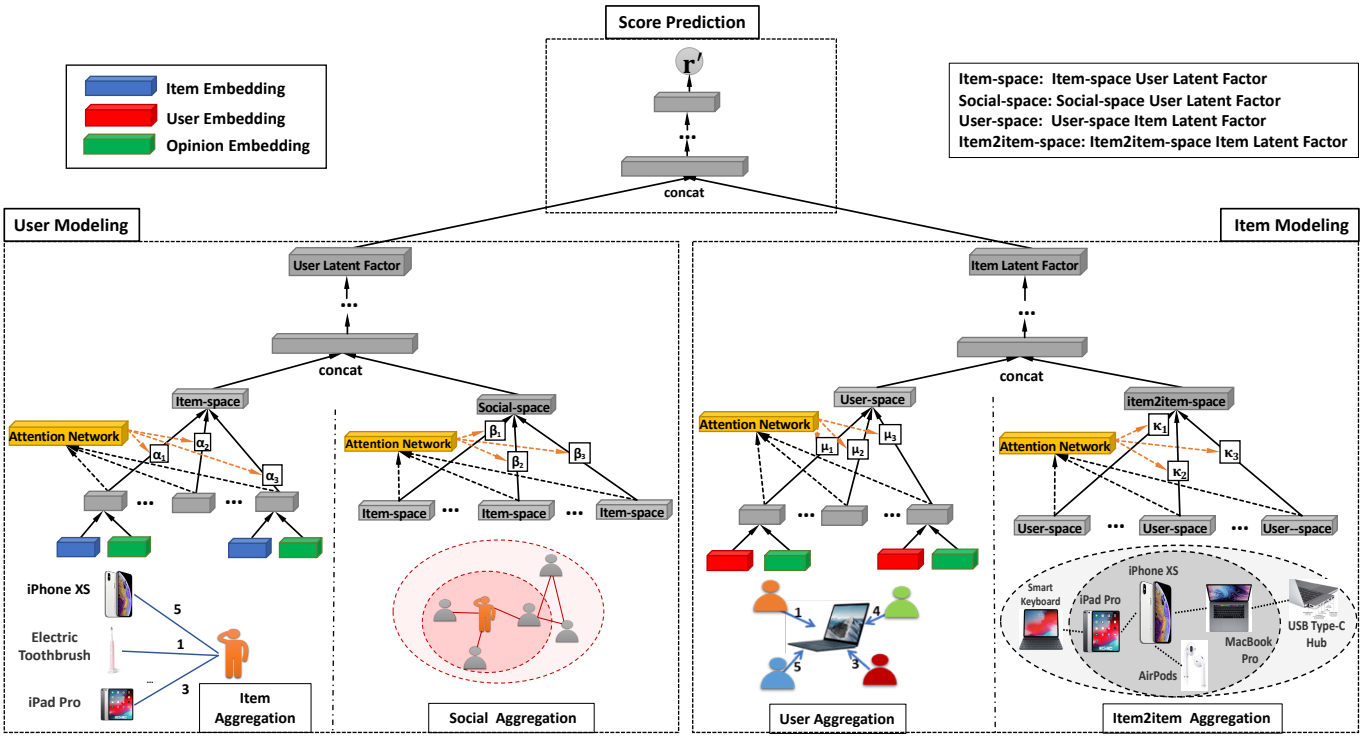


Figure 2: The overall architecture of the proposed model. It contains three major components: user modeling, item modeling, and score prediction.

tively. **The item aggregation** can be utilized to understand the users via interactions with items in the user-item graph (or item-space). **The social aggregation** can be used to understand the relationships between users in the social graph, which can help model users from the social perspective (or social-space). It is then intuitive to obtain user latent factors by combining information from both the item space and the social space.

The second component, item modeling aims to learn latent factors of the items. The item representations can be learned from the user perspectives in the user-item graph, while relationships between items in the item-item graph are also helpful in enhancing item representation learning. In particular, we introduce two aggregations for extracting useful information from these two graph types. One is **user aggregation**, which is utilized to model the representation of items via the interactions with users in the user-item graph (or user-space). The other one is **item2item aggregation**, which aims to enhance item representations from the item-item graph (or item2item-space). Information from both the user space and the item2item space can be combined to profile the final item latent factors.

The third component, rating prediction aims to learn the model parameters via prediction by integrating user and item modeling components. Next, we will detail each model component.

2.3 User Modeling

User modeling aims to learn user latent factors, denoted as $\mathbf{h}_i \in \mathbb{R}^d$ for user u_i. The challenge is how to inherently combine the user-item graph and social graph. To address this challenge, we first use two types of aggregations to learn factors from the two graphs, as shown in the left part of Figure 2. The first aggregation, denoted as the item aggregation, is utilized to learn item-space user latent factor $\mathbf{h}_i^I \in \mathbb{R}^d$ from the user-item graph. The second aggregation is social aggregation, where social-space user latent factor $\mathbf{h}_i^S \in \mathbb{R}^d$ is learned from the social graph. These two factors are then combined to form the final user latent factors $\mathbf{h}_i$. Next, we will introduce item aggregation, social aggregation and how to combine user latent factors from both item-space and social-space.

2.3.1 *Item Aggregation: Item-space User Latent Factor*

As the user-item graph contains not only interactions between users and items, but also users' opinions (or rating scores) on items, we provide a principled approach to jointly capture interactions and opinions in the user-item graph for learning item-space user latent factors $\mathbf{h}_i^I$ for a user u_i, which is used to model user latent factors via interactions in the user-item graph.

The purpose of item aggregation is to learn item-space user latent factor $\mathbf{h}_i^I$ by considering items a user u_i has interacted with and users' opinions on these items. To mathematically represent this aggregation, we use the following function:

$$\mathbf{h}_i^I = \sigma(\mathbf{W} \cdot AGG_{items}(\{\mathbf{x}_{ia}, \forall a \in C(i)\}) + \mathbf{b}) \quad (1)$$

where $C(i)$ is the set of items user u_i has interacted with (or u_i's neighbors in the user-item graph), $\mathbf{x}_{ia}$ is a representation vector to denote opinion-aware interaction between u_i and an item v_a, and AGG_{items} is the item aggregation function. In addition, σ denotes a non-linear activation function (i.e., a rectified linear unit), and $\mathbf{W}$ and $\mathbf{b}$ are the weight and bias of a neural network, respectively. Next

we discuss how to define the opinion-aware interaction representation $\mathbf{x}_{ia}$ and the aggregation function AGG_{items}.

A user can express his/her opinions (or rating scores), denoted as r, to items during user-item interactions. These opinions on items can capture users' preferences on items, which can help model item-space user latent factors. To model opinions, for each type of opinions r, we introduce an opinion embedding vector $\mathbf{e}_r \in \mathbb{R}^d$ that denotes each opinion r as a dense vector representation. For example, in a 5-star rating system, for each $r \in \{1, 2, 3, 4, 5\}$, we introduce an embedding vector $\mathbf{e}_r$. For an interaction between user u_i and item v_a with opinion r, we model the opinion-aware interaction representation $\mathbf{x}_{ia}$ as a combination of item embedding $\mathbf{q}_a$ and opinion embedding $\mathbf{e}_r$ via a Multi-Layer Perceptron (MLP). It can be denoted as g_v to fuse the interaction information with the opinion information, as shown in Figure 2. The MLP takes the concatenation of item embedding $\mathbf{q}_a$ and its opinion embedding $\mathbf{e}_r$ as input. The output of MLP is the opinion-aware representation of the interaction between u_i and v_a, $\mathbf{x}_{ia}$, as follows:

$$\mathbf{x}_{ia} = g_v([\mathbf{q}_a \oplus \mathbf{e}_r]) \tag{2}$$

where $\oplus$ denotes the concatenation operation between two vectors.

One popular aggregation function for AGG_{items} is the mean operator, where we take the element-wise mean of the vectors in $\{\mathbf{x}_{ia}, \forall a \in C(i)\}$. This mean-based aggregator is a linear approximation of a localized spectral convolution [11], as shown by the following function:

$$\mathbf{h}_i^I = \sigma(\mathbf{W} \cdot \left\{ \sum_{a \in C(i)} \alpha_i \mathbf{x}_{ia} \right\} + \mathbf{b}) \tag{3}$$

where α_i is fixed to $\frac{1}{|C(i)|}$ for all items in the mean-based aggregator. It assumes that all interactions contribute equally to understanding user u_i. However, this may not be optimal, due to the fact that the influence of interactions on users may vary dramatically. Hence, we should allow interactions to contribute differently to a user's latent factor by assigning each interaction a weight.

To alleviate the limitation of the mean-based aggregator, inspired by attention mechanisms [26], [27], an intuitive solution is to tweak α_i to be aware of the target user u_i, i.e., assigning an individualized weight for each (v_a, u_i) pair,

$$\mathbf{h}_i^I = \sigma(\mathbf{W} \cdot \left\{ \sum_{a \in C(i)} \alpha_{ia} \mathbf{x}_{ia} \right\} + \mathbf{b}) \tag{4}$$

where α_{ia} denotes the attention weight of the interaction with v_a in contributing to user u_i's item-space latent factor when characterizing user u_i's preference from the interaction history $C(i)$. In particular, we parameterize the *item attention* α_{ia} with a two-layer neural network, which we term as the *attention network*. The input to the attention network is the opinion-aware representation $\mathbf{x}_{ia}$ of the interaction and the target user u_i's embedding $\mathbf{p}_i$. Formally, the attention network is defined as:

$$\alpha_{ia}^* = \mathbf{w}_2^T \cdot \sigma(\mathbf{W}_1 \cdot [\mathbf{x}_{ia} \oplus \mathbf{p}_i] + \mathbf{b}_1) + b_2 \tag{5}$$

The final attention weights are obtained by normalizing the above attentive scores using the Softmax function, which can be interpreted as the contribution of the interaction to the item-space user latent factor of user u_i as:

$$\alpha_{ia} = \frac{\exp(\alpha_{ia}^*)}{\sum_{a \in C(i)} \exp(\alpha_{ia}^*)} \tag{6}$$

2.3.2 Social Aggregation: Social-space User Latent Factor

According to the social correlation theories [28], [29], users' preferences are similar to or influenced by their directly connected social friends. We should incorporate social information to further model user latent factors. Meanwhile, tie strengths between users can further influence users' behaviors from the social graph. In other words, the learning of social-space user latent factors should consider heterogeneity in the strength of social relations. Therefore, we introduce an attention mechanism to select social friends who are representative to characterize users' social information, and then aggregate their information.

In order to represent user latent factors from this social perspective, we propose social-space user latent factors to aggregate the item-space user latent factors of neighboring users from the social graph. In particular, the social-space user latent factor of u_i, $\mathbf{h}_i^S$, is used to aggregate the item-space user latent factors of users in u_i's social neighbors $N(i)$, as follows:

$$\mathbf{h}_i^S = \sigma(\mathbf{W} \cdot AGG_{social}(\left\{ \mathbf{h}_o^I, \forall o \in N(i) \right\}) + \mathbf{b}) \tag{7}$$

where AGG_{social} denotes the aggregation function on user's social neighbors.

One natural aggregation function for AGG_{social} is also the mean operator which takes the element-wise mean of the vectors in $\left\{ \mathbf{h}_o^I, \forall o \in N(i) \right\}$, as given by the following function:

$$\mathbf{h}_i^S = \sigma(\mathbf{W} \cdot \left\{ \sum_{o \in N(i)} \beta_i \mathbf{h}_o^I \right\} + \mathbf{b}) \tag{8}$$

where β_i is fixed to $\frac{1}{|N(i)|}$ for all neighbors for the mean-based aggregator. It assumes that all neighbors contribute equally to the representation of user u_i. However, as mentioned before, strong and weak ties are mixed together in a social network, and users are likely to share more similar tastes among strong ties than weak ties. Thus, we perform an attention mechanism with a two-layer neural network to extract these users who are important to influence u_i, and model their tie strengths, by relating *social attention* β_{io} with $\mathbf{h}_o^I$ and the target user embedding $\mathbf{p}_i$, as follows:

$$\mathbf{h}_i^S = \sigma(\mathbf{W} \cdot \left\{ \sum_{o \in N(i)} \beta_{io} \mathbf{h}_o^I \right\} + \mathbf{b}) \tag{9}$$

$$\beta_{io}^* = \mathbf{w}_2^T \cdot \sigma(\mathbf{W}_1 \cdot [\mathbf{h}_o^I \oplus \mathbf{p}_i] + \mathbf{b}_1) + b_2 \tag{10}$$

$$\beta_{io} = \frac{\exp(\beta_{io}^*)}{\sum_{o \in N(i)} \exp(\beta_{io}^*)} \tag{11}$$

where β_{io} can be seen as the strengths between users.

2.3.3 Learning User Latent Factor

In order to better learn user latent factors, item-space user latent factors and social-space user latent factors need to be considered together, since the social graph and the user-item graph provide information about users from different perspectives. We propose to combine these two latent factors as the final user latent factor via a standard MLP, where the item-space user latent factor $\mathbf{h}_i^I$ and the social-space user latent factor $\mathbf{h}_i^S$ are concatenated before feeding into MLP. Formally, the user latent factor $\mathbf{h}_i$ for user u_i is defined as:

$$\mathbf{c}_1^u = \left[\mathbf{h}_i^I \oplus \mathbf{h}_i^S\right] \tag{12}$$

$$\mathbf{c}_2^u = \sigma(\mathbf{W}_2 \cdot \mathbf{c}_1^u + \mathbf{b}_2) \tag{13}$$

$$\ldots$$

$$\mathbf{h}_i = \sigma(\mathbf{W}_l \cdot \mathbf{c}_{l-1}^u + \mathbf{b}_l) \tag{14}$$

where l is the index of a hidden layer.

2.4 Item Modeling

Item modeling is used to learn item latent factors, denoted as $\mathbf{z}_j$, for item v_j. As mentioned before, items are associated with not only users (consumers) in the user-item graph, but also related items in the item-item graph, and this provides different perspectives to learn item latent factors. To combine these two graphs inherently, we also adopt a similar method to that used in learning user latent factors in the user modeling. As shown in the right part of Figure 2, we first use two types of aggregation to learn two different item latent factors from these two graphs, respectively. The first aggregation, denoted as user aggregation, is utilized to learn user-space item latent factor $\mathbf{h}_j^U$ from the user-item graph. The second aggregation is the item2item aggregation where item2item-space item latent factor $\mathbf{h}_j^V$ can be learned from the item-item graph. The final item latent factors can then be formed by combining these two factors.

2.4.1 User Aggregation: User-space Item Latent Factor

We use a similar method as learning item-space user latent factors via item aggregation. For each item v_j, we need to aggregate information from the set of users who have interacted with v_j, denoted as $B(j)$.

Even for the same item, users might express different opinions during user-item interactions. These opinions from different users can capture the characteristics of the same item in different ways, as provided by users, which can help model item latent factors. For an interaction from u_t to v_j with opinion r, we introduce an opinion-aware interaction user representation $\mathbf{f}_{jt}$, which is obtained from the basic user embedding $\mathbf{p}_t$ and opinion embedding $\mathbf{e}_r$ via an MLP. We use g_u to denote the function used to fuse the interaction information with the opinion information:

$$\mathbf{f}_{jt} = g_u([\mathbf{p}_t \oplus \mathbf{e}_r]) \tag{15}$$

Then, to learn the user-space item latent factor $\mathbf{h}_j^U$, we propose to aggregate the opinion-aware interaction representation of users in $B(j)$ for item v_j. The user aggregation function is denoted as AGG_{users}, which aggregates the opinion-aware interaction representation of users in $\{\mathbf{f}_{jt}, \forall t \in B(j)\}$. Therefore,

$$\mathbf{h}_j^U = \sigma(\mathbf{W} \cdot AGG_{users}(\{\mathbf{f}_{jt}, \forall t \in B(j)\}) + \mathbf{b}) \tag{16}$$

In addition, we introduce an attention mechanism to differentiate the importance weight μ_{jt} of users with a two-layer neural attention network, taking $\mathbf{f}_{jt}$ and $\mathbf{q}_j$ as the inputs,

$$\mathbf{h}_j^U = \sigma(\mathbf{W} \cdot \left\{\sum_{t \in B(j)} \mu_{jt}\mathbf{f}_{jt}\right\} + \mathbf{b}) \tag{17}$$

$$\mu_{jt}^* = \mathbf{w}_2^T \cdot \sigma(\mathbf{W}_1 \cdot [\mathbf{f}_{jt} \oplus \mathbf{q}_j] + \mathbf{b}_1) + b_2 \tag{18}$$

$$\mu_{jt} = \frac{\exp(\mu_{jt}^*)}{\sum_{t \in B_{(j)}} \exp(\mu_{jt}^*)} \tag{19}$$

This *user attention* μ_{jt} is used to capture heterogeneous influences from user-item interactions for learning the user-space item latent factors.

2.4.2 Item2item Aggregation: Item2item-space Item Latent Factor

As items are not independent and are likely to be similar and related, it is desirable to further enrich the item latent factors from similar or related items. Thus, we introduce item2item aggregation operation to learn item2item-space item latent factors. More specifically, the item2item-space item latent factors of item v_j, denoted as $\mathbf{h}_j^V$, is to aggregate the user-space item latent factors of items in v_j's similar or related items $M(j)$, as follows:

$$\mathbf{h}_j^V = \sigma(\mathbf{W} \cdot AGG_{Item2item}(\left\{\mathbf{h}_k^U, \forall k \in M(j)\right\}) + \mathbf{b}) \tag{20}$$

where $AGG_{item2item}$ is the aggregation function on the item's 'neighbors' $M(\cdot)$ in the item-item graph I_G. Additionally, we also introduce an attention mechanism to differentiate the important weight κ_{jk} of related items with a two-layers neural attention network,

$$\mathbf{h}_j^V = \sigma(\mathbf{W} \cdot \left\{\sum_{k \in M(j)} \kappa_{jk}\mathbf{h}_k^U\right\} + \mathbf{b}) \tag{21}$$

$$\kappa_{jk}^* = \mathbf{w}_2^T \cdot \sigma(\mathbf{W}_1 \cdot [\mathbf{h}_k^U \oplus \mathbf{q}_j] + \mathbf{b}_1) + b_2 \tag{22}$$

$$\kappa_{jk} = \frac{\exp(\kappa_{jk}^*)}{\sum_{k \in M(j)} \exp(\kappa_{jk}^*)} \tag{23}$$

2.4.3 Learning Item Latent Factor

Given these different item representations (i.e., $\mathbf{h}_j^U$ and $\mathbf{h}_j^V$), we finally combine them together for the final item latent factor via a standard MLP with l hidden layers, as follows:

$$\mathbf{c}_1^v = \left[\mathbf{h}_j^U \oplus \mathbf{h}_j^V\right] \tag{24}$$

$$\mathbf{c}_2^v = \sigma(\mathbf{W}_2 \cdot \mathbf{c}_1^v + \mathbf{b}_2) \tag{25}$$

$$\ldots$$

$$\mathbf{z}_j = \sigma(\mathbf{W}_l \cdot \mathbf{c}_{l-1}^v + \mathbf{b}_l) \tag{26}$$

2.5 Rating Prediction

In this subsection, we design recommendation tasks to learn model parameters. There are various recommendation tasks, such as item ranking and rating prediction. In this work, we apply the proposed GraphRec+ model for the recommendation task of rating prediction. With the latent factors of users and items (i.e., $\mathbf{h}_i$ and $\mathbf{z}_j$), we can first concatenate them $[\mathbf{h}_i \oplus \mathbf{z}_j]$ and then feed it into the MLP for rating prediction, as follows:

$$\mathbf{g}_1 = [\mathbf{h}_i \oplus \mathbf{z}_j] \tag{27}$$

$$\mathbf{g}_2 = \sigma(\mathbf{W}_2 \cdot \mathbf{g}_1 + \mathbf{b}_2) \tag{28}$$

$$\dots$$

$$\mathbf{g}_{l-1} = \sigma(\mathbf{W}_l \cdot \mathbf{g}_{l-1} + \mathbf{b}_l) \tag{29}$$

$$r'_{ij} = \mathbf{w}^T \cdot \mathbf{g}_{l-1} \tag{30}$$

where l is the index of a hidden layer, and r'_{ij} is the predicted rating from u_i to v_j.

2.6 Model Training

To estimate the model parameters of GraphRec+, we need to specify an objective function to optimize. Since we focus here on the task of rating prediction, a commonly used objective function is formulated as follows:

$$Loss = \frac{1}{2\,|\mathcal{O}|} \sum_{i,j \in \mathcal{O}} (r'_{ij} - r_{ij})^2 \tag{31}$$

where $|\mathcal{O}|$ is the number of observed ratings, and r_{ij} is the ground truth rating assigned by user u_i on item v_j.

To optimize the objective function, we adopt the RMSprop [30] as the optimizer in our implementation, rather than the vanilla SGD. At each time, it randomly selects a training instance and updates each model parameter towards the direction of its negative gradient. There are three types of embeddings in our model, including item embedding $\mathbf{q}_j$, user embedding $\mathbf{p}_i$, and opinion embedding $\mathbf{e}_r$. They are randomly initialized and jointly learned during the training stage. We do not use one-hot vectors to represent each user and item, since the raw features are very large and highly sparse. By embedding high-dimensional sparse features into a low-dimensional latent space, the model can be easily trained [3], [25]. Opinion embedding matrix $\mathbf{e}$ depends on the rating scale of the system. For instance, for a 5-star rating system, opinion embedding matrix $\mathbf{e}$ contains five different embedding vectors to denote scores in $\{1, 2, 3, 4, 5\}$. Overfitting is a major problem in optimizing deep neural network models. To alleviate this issue, the dropout strategy [31] was applied to our model. The idea of the dropout strategy is to randomly drop some neurons during the training process. When updating parameters, only some of them will be updated. Moreover, as dropout is disabled during testing, the whole network is used for prediction.

Time Complexity Analysis. We analyze here the time complexity of the proposed model, GraphRec+. The time cost of GraphRec+ is reflected by the aggregation operations in the different graphs. Given n users, m items, suppose that each user directly connects with $\hat{n}_v$ items (user-item graph) and $\hat{n}_s$ social neighbors (social graph) on average, and that each item directly connects with $\hat{m}_u$ users (user-item graph) and $\hat{m}_k$ similar or related items (item-item graph) on average. For item aggregation, the major time cost lies in the first-order aggregation, as shown in Equation 4. At first, Multi-Layer Perceptron (e.g., two layers) is adopted to calculate the attention of the interaction, as shown in Equation 5, which costs about $O(n\hat{n}_v d)$. Then, the item aggregation is used to obtain item-space user latent factor, which also costs about $O(n\hat{n}_v d)$. Since the number of aggregation operations is much smaller than n and m, we can regard it as a constant. The overall time complexity is about $O(n(\hat{n}_v + \hat{n}_s)d + m(\hat{m}_u + \hat{m}_k)d)$. As $\hat{n}_v, \hat{n}_s, \hat{m}_u, \hat{m}_k \ll \min\{n, m\}$, the overall complexity is linear to the number of users and items. Therefore, the total time complexity is acceptable in practice.

3 Experiment

3.1 Experimental Settings

3.1.1 Datasets

In our experiments, three representative datasets related to social recommendation are utilized to validate the performance of the proposed methods for rating prediction.

- **Ciao and Epinions**[1]. These two datasets are taken from popular social networking websites, Ciao[2] and Epinions[3]. Each social networking service allows users to rate items, browse and write reviews, and add friends to their 'Circle of Trust'. Hence, they provide a large amount of rating and social information. The ratings scale is from 1 to 5.
- **Flixster**. Flixster is a popular social networking website[4] on which users can review movies they have watched. Users can add others to their friends list and create a social network. Thus, Flixster dataset [10] also contains rating and social information. The rating values of the Flixster dataset are 10 discrete numbers in the range $[0.5, 5]$, with a step size of 0.5.

In the social graph, the number of nodes corresponds to the number of users in the dataset, and the number of edges in the user-user graph corresponds to the number of social relations between users in the dataset. For these three datasets, we removed those users with no social relations and no ratings, and items with no ratings, because these are cold-start users and items, which is beyond the scope of this work. We would like to leave the cold-start problem as our future work. The statistics of these three datasets are presented in Table 2.

3.1.2 Item-item Graph Construction

As there is no prior information that explicitly expresses the relationships between items in the three datasets, one natural way to connect the similar/related items is based on the similarity or relatedness between two items. The basic idea is that two items v_i and v_j should be connected

1. Both Ciao and Epinions datasets are available at: http://www.cse.msu.edu/∼tangjili/trust.html
2. http://www.ciao.co.uk
3. http://www.epinions.com
4. https://www.flixster.com

Table 2: Statistics of three datasets

Datasets	Ciao	Epinions	Flixster
# of Users	7,317	18,088	58,470
# of Items	104,975	261,649	38,076
# of Ratings	283,319	764,352	3,619,736
Density of Interaction	0.0368%	0.0161%	0.1625%
# of Social Relations	111,781	355,813	667,313
Density of Social Relations	0.2087%	0.1087%	0.0195%

if their similarity or relatedness is high, and unconnected otherwise.

In this work, we empirically adopt Cosine-based Similarity to measure the similarity between two items. Different sources can be used to denote items such as textual descriptions, the visual content of items, user-item interactions, and knowledge graph. The popular way for calculating the similarity between two items is based on the common users who clicked on or rated the items [22], [32]. Here, we adopt the user-item interactions $\mathbf{R}$ to represent the items. Then, according to items' similarity, we propose to extract the top-k similar items, as the connected neighbors for each item, to build the item-item graph I_G. The number of edges in the item-item graph is $k \times$ the number of items in the corresponding dataset.

3.1.3 Evaluation Metrics

In order to evaluate the quality of the recommendation algorithms, two popular metrics are adopted to evaluate the predictive accuracy, namely, Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) [24].Smaller values of MAE and RMSE indicate better predictive accuracy. Note that in practice, small improvement in RMSE or MAE terms can have a significant impact on the quality of the top-few recommendations [33].

3.1.4 Baselines

To evaluate performance, we compared our GraphRec+ with three groups of methods, including traditional recommender systems, traditional social recommender systems, and deep neural network-based recommender systems. For each group, we select representative baselines, and these are listed below:

- **PMF** [34]: **P**robabilistic **M**atrix **F**actorization utilizes user-item rating matrix only and models latent factors of users and items by Gaussian distributions.
- **SoRec** [35]: **So**cial **Rec**ommendation performs co-factorization on the user-item rating matrix and user-user social relations matrix.
- **SoReg** [6]: **So**cial **Reg**ularization models social network information as regularization terms to constrain the matrix factorization framework.
- **SocialMF** [10]: It considers the trust information and propagation of trust information into the matrix factorization model for recommender systems.
- **TrustMF** [36]: This method adopts matrix factorization technique that maps users into two low-dimensional spaces: truster space and trustee space, by factorizing trust networks according to the directional property of trust.
- **NeuMF** [3]: This method is a state-of-the-art collaborative Filtering with neural network architecture. The original implementation is for recommendation ranking tasks and here we adjust its loss to the squared loss for rating prediction.
- **DeepSoR** [37]: This model employs a deep neural network to learn representations of each user from social relations, and then integrates this into the probabilistic matrix factorization for rating prediction.
- **GCMC+SN** [38]: This model is a state-of-the-art recommender system with graph neural network architecture. In order to incorporate social network information into GCMC, we utilize the $node2vec$ [39] to generate user embedding as user side information, instead of using the raw feature social connections ($\mathrm{T} \in \mathbb{R}^{n \times n}$) directly. The reason is that the raw feature input vectors is highly sparse and high-dimensional. Using the network embedding techniques can help compress the raw input feature vector to a low-dimensional and dense vector, which makes the model easier to train.
- **GraphRec** [24]: This model employs a graph neural network to model user and item representations in social recommendations for rating prediction.

PMF and NeuMF are pure collaborative filtering models that do not incorporate social network information in the rating prediction, while the other models incorporate social recommendation. In addition, we compared GraphRec+ with three state-of-the-art neural network-based social recommender systems, i.e., DeepSoR, GCMC+SN, and GraphRec.

3.1.5 Parameter Settings

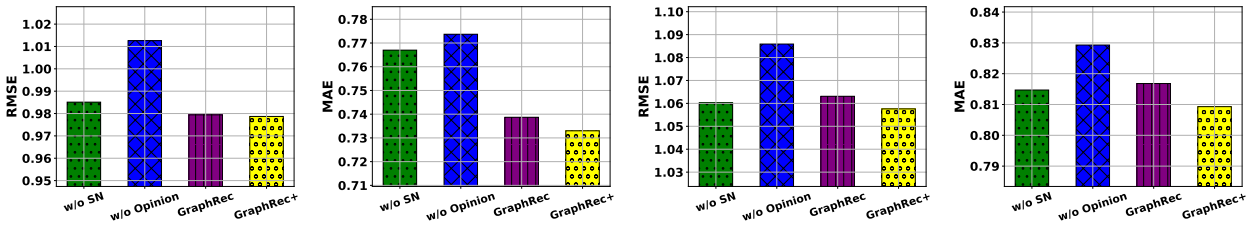
We have implemented our proposed method on the basis of PyTorch[5], a well-known Python library for neural networks. For each dataset, we used $x\%$ as a training set for learning the parameters, $(1 - x\%)/2$ as a validation set to tune the hyper-parameters, and $(1 - x\%)/2$ as a testing set for the final performance comparison, where x was varied as $\{80\%, 60\%\}$. For embedding size d, we tested the value of $\{8, 16, 32, 64, 128, 256\}$. The batch size and learning rate were searched in $\{32, 64, 128, 512\}$ and $\{0.0005, 0.001, 0.005, 0.01, 0.05, 0.1\}$, respectively. For building the item-item implicit network, the value of k for the most similar items was tested in $\{3, 7, 10, 15, 30, 50, 100\}$. Moreover, we empirically set the size of the hidden layer to the same as the embedding size, and set the activation function as ReLU. Without special mention, we employed three hidden layers for all the neural components. The early stopping strategy was performed, where we stopped training if the RMSE on validation set increased for 5 successive epochs. For all neural network methods, we randomly initialized the model parameters with a Gaussian distribution, with a mean and standard deviation of 0 and 0.1, respectively. The parameters for the baseline algorithms were initialized as in the corresponding papers and were then carefully tuned to achieve optimal performance.

5. https://pytorch.org/

Table 3: Performance comparison of different recommender systems

Datasets		Ciao (80%)		Ciao (60%)		Epinions (80%)		Epinions (60%)		Flixster (80%)		Flixster (60%)	
Metrics		RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
Algorithms	PMF	1.1238	0.9021	1.1967	0.9520	1.2128	0.9952	1.2739	1.0211	1.0042	0.7559	1.0793	0.8296
	SoRec	1.0652	0.8410	1.0738	0.8489	1.1437	0.8961	1.1563	0.9086	0.9693	0.7303	0.9974	0.7537
	SoReg	1.0848	0.8611	1.0947	0.8987	1.1703	0.9119	1.1936	0.9412	0.9727	0.7415	1.0372	0.7736
	SocialMF	1.0501	0.8270	1.0592	0.8353	1.1328	0.8837	1.1410	0.8965	0.9705	0.7378	1.0060	0.7647
	TrustMF	1.0479	0.7690	1.0543	0.7681	1.1395	0.8410	1.1505	0.8550	0.9380	0.7091	0.9978	0.7480
	NeuMF	1.0617	0.8062	1.0824	0.8251	1.1476	0.9072	1.1645	0.9097	0.9698	0.7420	1.0540	0.7663
	DeepSoR	1.0316	0.7739	1.0437	0.7813	1.0972	0.8383	1.1135	0.8520	0.9548	0.7190	0.9913	0.7462
	GCMC+SN	0.9931	0.7526	1.0221	0.7697	1.0711	0.8590	1.1004	0.8602	0.9613	0.7351	0.9914	0.7606
	GraphRec	0.9794	0.7387	1.0093	0.7540	1.0631	0.8168	1.0878	0.8441	0.9452	0.7175	0.9857	0.7366
	GraphRec+	0.9787	0.733	0.9962	0.7446	1.0576	0.8093	1.0819	0.8336	0.9303	0.7047	0.9669	0.7289
Improvement* (%)		0.07%	0.77%	1.30%	1.25%	0.52%	0.92%	0.54%	1.24%	0.82%	0.63%	1.91%	1.00%

* The value indicates the percentage of improvements gained by GraphRec+ compared to the strongest baseline.



(a) Ciao-RMSE (b) Ciao-MAE (c) Epinions-RMSE (d) Epinions-MAE

Figure 3: Effect of social network, user opinions, and item-item graph on Ciao and Epinions datasets.

3.2 Performance Comparison of Recommender Systems

We first compare the recommendation performance of all methods. Table 3 shows the overall rating prediction error $w.r.t.$ RMSE and MAE among the recommendation methods on Ciao, Epinions, and Flixster datasets. We have the following main findings:

- SoRec, SoReg, SocialMF, and TrustMF always outperform PMF. All of these methods are based on matrix factorization. SoRec, SoReg, SocialMF, and TrustMF leverage both the rating and social network information, whereas PMF only uses the rating information. These results support the notion that social network information is useful for, and complementary to, rating information for recommendations.
- NeuMF obtains much better performance than PMF. Both methods use only the rating information. However, NeuMF is based on neural network architecture, and this performance gain demonstrates the power of neural network models in recommender systems.
- DeepSoR and GCMC+SN perform better than SoRec, SoReg, SocialMF, and TrustMF. Each of these methods take advantage of both rating and social network information. However, DeepSoR and GCMC+SN are based on neural network architectures, and this performance gain indicates once more the power of neural network models in recommender systems.
- Among baselines, GCMC+SN shows quite strong performance. This implies that GNNs are powerful in representation learning for graph data, since it naturally integrates the node information as well as topological structure.
- Our method GraphRec+ consistently outperforms all the baseline methods. Compared to DeepSoR and GCMC+SN, our model provides advanced model components for integrating rating and social network information. In addition, our model provides a way to consider both interactions and opinions in the user-item graph. In comparison to GraphRec, the GraphRec+ explicitly incorporates the relationships between items for modeling the item latent factors. In particular, GraphRec+ performs better than GraphRec when training models under the 60% dataset condition. In the following subsections, we investigate this model further in order to better understand the contributions of model components to the proposed framework.

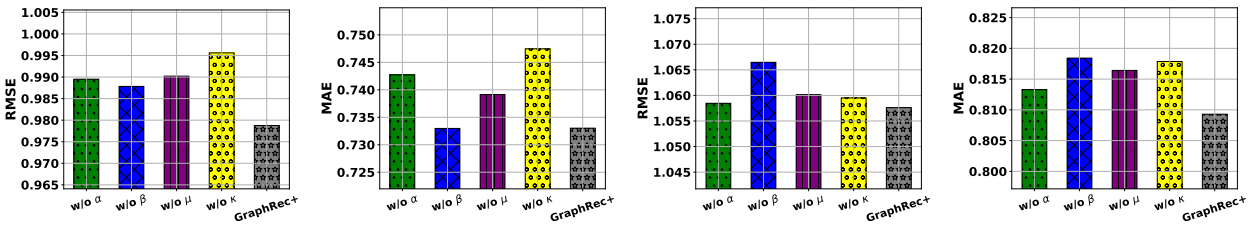
To sum up, the results of the model comparison suggest that: (1) social network information and items' neighbor information are helpful for recommendations; (2) graph neural network models can boost recommendation performance; and (3) the proposed framework outperforms representative baselines.

3.3 Model Analysis

In this subsection, we study the impact of model components and model hyper-parameters.

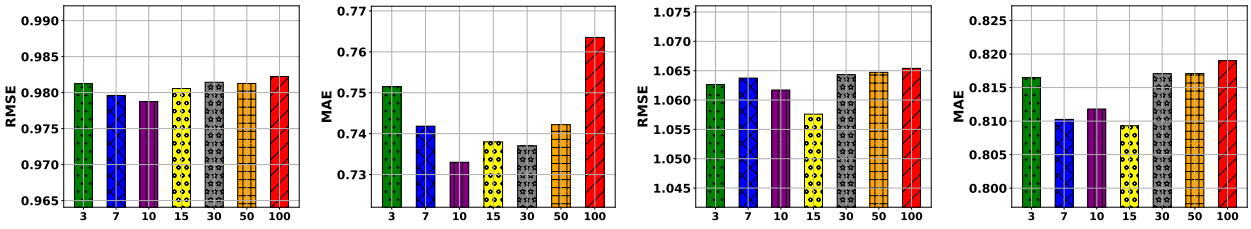
3.3.1 The Effect of Social Network, User Opinions, and Item-item Graph on Model Performance

In the last subsection, we have demonstrated the effectiveness of the proposed framework. The proposed framework provides model components to: (1) integrate social network information, (2) integrate item-item graphs; and (3) incorporate users' opinions about interactions with items. To understand the working of GraphRec+, we compare GraphRec+ with its three variants, which are GraphRec+ (w/o SN), GraphRec+ (w/o Opinion), and GraphRec. These three variants are defined as follows:



(a) Ciao-RMSE (b) Ciao-MAE (c) Epinions-RMSE (d) Epinions-MAE

Figure 4: Effect of attention mechanisms on Ciao and Epinions datasets.



(a) Ciao-RMSE (b) Ciao-MAE (c) Epinions-RMSE (d) Epinions-MAE

Figure 5: Effect of Top-k similar/related items on Ciao and Epinions datasets.

- GraphRec+ **(w/o SN)**: The social network information of GraphRec+ is removed. This variant only uses the item-space user latent factor $\mathbf{h}_i^I$ to represent user latent factors $\mathbf{h}_i$, while ignoring the social-space user latent factors $\mathbf{h}_i^S$.
- GraphRec+ **(w/o Opinion)**: For learning item-space user latent factor and item latent factor, the opinion embedding is removed during learning $\mathbf{x}_{ia}$ and $\mathbf{f}_{jt}$. This variant ignores the users' opinion on the user-item interactions.
- **GraphRec**: The item-item implicit graph of GraphRec+ is removed when modeling item latent factor $\mathbf{z}_j$. This variant only uses the user-space user latent factor $\mathbf{h}_j^U$ to represent item latent factors $\mathbf{h}_j$, while ignoring the item2item-space item latent factors $\mathbf{h}_j^V$.

Figure 3 shows the performance of GraphRec+ and its variants on the two datasets Ciao and Epinions. From the results, we have the following findings:

- **Social Network Information.** Our focus here is on analyzing the effectiveness of social network information. GraphRec+ (**w/o SN**) performs worse than GraphRec+. It verifies that social network information is important in learning user latent factors and for boosting the recommendation performance.
- **Opinions in Interaction.** We can see that without opinion information (**w/o Opinions**), the performance of rating prediction deteriorates significantly. It justifies our assumption that opinions on user-item interactions have useful information that can help learn user or item latent factors and improve the performance of recommendation.
- **Item-item Graph.** The GraphRec+ is proposed to consider the relationships among items via building item-item graphs. We can see that GraphRec+ can perform better than GraphRec. For example, on average, the relative improvement on Ciao and Epinions is 0.07% and 0.52% on RMSE, and 0.77% and 0.92% on MAE, respectively. These results provide support for the idea that the similar or related items are beneficial to profile items, and consequently, help improve recommendation performance.

3.3.2 The Effect of Attention Mechanisms on Model Performance

To get a better understanding of the proposed GraphRec+ model, we further evaluate the key components of GraphRec+: *attention mechanisms*. There are four different attention mechanisms during aggregation operation, including item attention α, social attention β, user attention μ, and item2item attention κ. We compare GraphRec+ with its four variants: GraphRec+(w/o α), GraphRec+(w/o β), GraphRec+(w/o μ), and GraphRec+(w/o κ). These four variants are defined as follows:

- GraphRec+ (**w/o** α) - Eliminating the effect of item attention α_{i*} by setting $\alpha_i = \frac{1}{|C(i)|}$ in Eq.5;
- GraphRec+ (**w/o** β) - Eliminating the effect of social attention β_{i*} by setting $\beta_i = \frac{1}{|N(i)|}$ in Eq.9;
- GraphRec+ (**w/o** μ) - Eliminating the effect of user attention μ_{j*} by setting $\mu_j = \frac{1}{|B(j)|}$ in Eq.17;
- GraphRec+ (**w/o** κ) - Eliminating the effect of related item attention κ_{j*} by setting $\kappa_j = \frac{1}{|M(j)|}$ in Eq.21;

The results of the different attention mechanisms on GraphRec+ are shown in Figure 4. From the results, we can see that the performance of GraphRec+ degrades when eliminating the effects of the different kinds of attention mechanisms. These results suggest that when performing the aggregation operation, differentiating the importance of each local neighbor in the graph can help boost the recommendation performance.

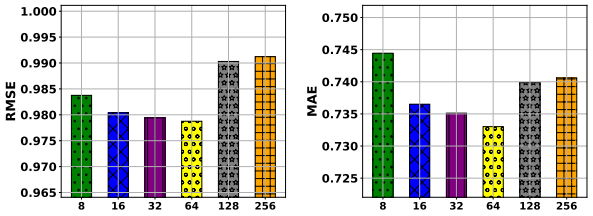
3.3.3 The Effect of Top-k Similar/Related Items on Model Performance

The relations between items provide another stream of potential information for profiling items. We empirically

Table 4: Performance comparison of different recommender systems on ranking metrics (NDCG@K).

Datasets		Ciao			Epinions			Flixster		
Metrics		NDCG@5	NDCG@10	NDCG@20	NDCG@5	NDCG@10	NDCG@20	NDCG@5	NDCG@10	NDCG@20
Algorithms	PMF	0.3044	0.3067	0.3648	0.3073	0.3627	0.3847	0.5112	0.5255	0.5321
	SoRec	0.3592	0.3690	0.3831	0.4015	0.4380	0.4568	0.5407	0.5601	0.5683
	SoReg	0.3670	0.3772	0.3951	0.4368	0.4684	0.5044	0.5504	0.5713	0.5800
	SocialMF	0.3784	0.3891	0.4119	0.4438	0.4620	0.4691	0.5565	0.5789	0.5880
	TrustMF	0.3776	0.3886	0.4103	0.4325	0.4630	0.4982	0.5662	0.5911	0.6011
	NeuMF	0.3980	0.4096	0.4342	0.4246	0.4429	0.4814	0.5344	0.5638	0.5718
	DeepSoR	0.3806	0.3938	0.4160	0.4421	0.4728	0.5046	0.5536	0.5754	0.5845
	GCMC+SN	0.3814	0.4307	0.4611	0.4475	0.4776	0.5022	0.5707	0.5928	0.6039
	GraphRec	0.3972	0.4410	0.4618	0.4580	0.4914	0.5124	0.5948	0.6245	0.6360
	GraphRec+	0.4351	0.4738	0.4966	0.5169	0.5442	0.5684	0.6385	0.6669	0.6748
Improvement* (%)		9.53	7.44	7.52	12.88	10.74	10.92	7.34	6.8	6.11

* The value indicates the percentage of improvements gained by GraphRec+ compared to the strongest baseline.



(a) Ciao-RMSE (b) Ciao-MAE

Figure 6: Effect of embedding size on Ciao dataset.

choose the top-k similar/related items for each item to build the item-item graph I_G. In this subsection, we investigate how the value of k affects the performance of the proposed method. Figure 5 shows the performance with varied values of k on Ciao and Epinions datasets. We first note that the best performance among the tested k values is $k = 10$ for Ciao and $k = 15$ for Epinions. When the value of k increases, the performance tends to increase initially. This indicates that including the most similar/related items is helpful. However, when the value of k becomes too large, the performance degrades as it may introduce too many noises with the similar/related items into the item-item graph.

3.3.4 The Effect of Embedding Size on Model Performance

We now analyze the effect of the embedding size of the user embedding $\mathbf{p}$, item embedding $\mathbf{q}$, and opinion embedding $\mathbf{e}$, on the performance of our model.

Figure 6 presents the performance comparison with respect to the length of embedding of our proposed model on Ciao dataset. In general, with an increase in the embedding size, the performance first increases and then decreases. When the embedding size is increased from 8 to 64, the performance improves significantly. However, when the embedding size reaches 128, GraphRec+ degrades the performance. This demonstrates that using a large embedding size has a substantial effect on model performance. Nevertheless, if the length of the embedding is too large, the complexity of our model will significantly increase. Therefore, we need to find a suitable length of embedding in order to balance the trade-off between the performance and the complexity of the model.

3.4 Validation on Ranking Metrics

In this subsection, we utilize one popular ranking metric to further evaluate the performance of different recommender systems applied to the top-k recommendation task: Normalized Discounted Cumulative Gain (NDCG@K). NDCG@K accounts for the position of the hit by assigning a higher score to hit at the top positions [3], [40]. We set K as 5, 10, and 20 in this experiment. Higher values of NDCG@K indicate better predictive performance.

As the ranking task is too time-consuming to rank all the items for all the users, we randomly sample 100 items with which the user did not interact and then rank the test items among them. As these three datasets provide users explicit ratings on items, we convert them into 1 as the implicit feedback. This processing method is widely used in previous works on recommendations with implicit feedback [41]. We randomly split the user-item interactions of each dataset into a training set (80%) to learn the parameters, a validation set (10%) to tune the hyper-parameters, and a test set (10%) for the final performance comparison. For the fair comparison between all recommender systems, we perform the point-wise prediction with sigmoid cross entropy loss using negative sampling, where we sample one negative instance per positive instance [3].

Table 4 presents the NDCG performance of different recommender systems. The overall trend is the same previously found in section 3.2. GraphRec+ consistently yields the best performance on all the datasets. In particular, GraphRec+ improves over the strongest baselines $w.r.t.$ NDCG@5 by 9.53%, 12.88%, and 7.34% in Ciao, Epinions, and Flixster, respectively. This verifies the importance of considering the relations among items for enhancing the representation learning of items for social recommendations.

4 Related Work

In this section, we briefly review some related work focusing on social recommendations, deep neural network techniques employed for recommendations, and advanced graph neural networks.

Exploiting social relations for recommendations has attracted significant attention in recent years [7], [36], [42]. One common assumption about these models is that a user's preference is similar to or influenced by the people around him/her (nearest neighbors), which is demonstrated by social correlation theories [28], [29]. Along with this line, SoRec [35] proposed a co-factorization method, which shares a common latent user-feature matrix factorized by ratings and social relations. TrustMF [36] modeled mutual influence between users and mapped users into two

low-dimensional spaces, truster space and trustee space, by factorizing social trust networks. SoDimRec [8] first adopted a community detection algorithm to partition users into several clusters, and then exploited the heterogeneity of social relations and weak dependency connections for recommendation. Comprehensive overviews on social recommender systems can be found in surveys [4].

In recent years, deep neural network models have had a great impact on learning effective feature representations in various fields, such as speech recognition [43], Computer Vision (CV) [44], [45] and Natural Language Processing (NLP) [46]. Some recent efforts have applied deep neural networks to recommendation tasks and shown promising results, but most of them used deep neural networks to model audio features of music [47], [48], textual description of items [26], [49], [50], and visual content of images [44], [51]. NeuMF [3] presented a Neural Collaborative Filtering framework to learn the non-linear interactions between users and items.

However, the application of deep neural networks in social recommender systems has been rare, until very recently. In particular, NSCR [52] extended the NeuMF [3] model to cross-domain social recommendations, that is, recommending items of information domains to potential users of social networks. This method produces a neural social collaborative ranking recommender system. However, NSCR requires that users have one or more social network accounts (e.g., Facebook, Twitter, Instagram), which limits its data collocation and application in practice. SMR-MNRL [5] developed a socially-aware movie recommendation using social media from the viewpoint of learning a multimodal heterogeneous network representation for ranking. They exploited the recurrent neural network and convolutional neural network to learn the representation of movies' textual description and poster image, and adopted a random-walk based learning method into multimodal neural networks. In all these works [52] [5], they addressed the task of cross-domain social recommendations for ranking metric, which is different from traditional social recommender systems.

Task with neural networks, included DLMF [53] and DeepSoR [37], are of greatest relevance to our work. DLMF [53] uses an auto-encoder on ratings to learn representations for initializing an existing matrix factorization. A two-phase trust-aware recommendation process is proposed that utilizes deep neural networks in matrix factorization initialization to synthesize the user's interests and their trust friends' interests, together with the impact of community effects, based on matrix factorization for recommendations. DeepSoR [37] utilized neural networks to learn social information about the users' k-nearest neighbors, and then integrated this into a probabilistic matrix factorization in order to make predictions.

More recently, Graph Neural Networks (GNNs) have been shown as capable of learning on graph structured data [11], [13], [14], [54], [55]. In the task of recommender systems, the user-item interaction contains the ratings given to items by users, and this is representative of typical graph data. Therefore, GNNs have been proposed to solve the recommendation problem [38], [56], [57], [58]. In sR-MGCNN [56], GNNs were adopted to extract graph embeddings for users and items, which were then combined with recurrent neural network to perform a diffusion process. GCMC [38] proposed a graph auto-encoder framework, which produced latent features of users and items through a form of differentiable message passing on the user-item graph. PinSage [57] extended GraphSage [12] to learn the embedding of nodes in web-scale graphs for item recommendation. NGCF [40] proposed to explicitly encode user/item signals in the form of high-order connectivities by graph propagation for item recommendations. MMGCN [58] employed information propagation on the modality-aware bipartite user-item graph to enhance micro-video recommendation. Despite the compelling success achieved by previous work, little attention has been paid to social recommendation with GNNs. Our work of developing a graph neural network for social recommendation as presented in this paper serves to fill this gap.

5 Conclusion

In this work, we have proposed a Graph Neural Network framework for social recommendations (**GraphRec+**). In particular, different aggregation operations are proposed to model the graph data to learn better user and item representations. Moreover, our model encodes the users' opinions towards items to enhance representation learning when modeling the user-item graph. In addition, we propose to utilize an attention mechanism to capture heterogeneous strengths of social relations when modeling social graphs. Finally, we propose to explicitly incorporate the relationships between items into our model in order to profile items. Our experiments show that items' relations can help enhance the performance of our proposed model. Comprehensive experiments on three real-world datasets show the effectiveness of our model.

In this work, we have only utilized the user-item interactions to build the item-item graph, yet rich side information is often associated with items, such as attributes and knowledge graph. Therefore, it would be interesting to investigate how to incorporate other side information for building item-item graphs to further enhance the recommender systems.

References

[1] X. Zhao, L. Xia, L. Zhang, Z. Ding, D. Yin, and J. Tang, "Deep reinforcement learning for page-wise recommendations," in *Proceedings of the 12th ACM Recommender Systems Conference*. ACM, 2018, pp. 95–103.

[2] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, 2009.

[3] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web, WWW*, 2017, pp. 173–182.

[4] J. Tang, X. Hu, and H. Liu, "Social recommendation: a review," *Social Network Analysis and Mining*, vol. 3, no. 4, pp. 1113–1133, 2013.

[5] Z. Zhao, Q. Yang, H. Lu, T. Weninger, D. Cai, X. He, and Y. Zhuang, "Social-aware movie recommendation via multimodal network learning," *IEEE Transactions on Multimedia*, vol. 20, no. 2, pp. 430–440, 2018.

[6] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *Proceedings of the fourth ACM international conference on Web Search and Data Mining*. ACM, 2011, pp. 287–296.

[7] X. Wang, W. Lu, M. Ester, C. Wang, and C. Chen, "Social recommendation with strong and weak ties," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2016, pp. 5–14.

[8] J. Tang, S. Wang, X. Hu, D. Yin, Y. Bi, Y. Chang, and H. Liu, "Recommendation with social dimensions," in *AAAI*, 2016, pp. 251–257.

[9] P. Resnick and H. R. Varian, "Recommender systems," *Communications of the ACM*, vol. 40, no. 3, pp. 56–58, 1997.

[10] M. Jamali and M. Ester, "A matrix factorization technique with trust propagation for recommendation in social networks," in *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010, pp. 135–142.

[11] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[12] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.

[13] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, 2016, pp. 3844–3852.

[14] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[15] Y. Ma, S. Wang, C. C. Aggarwal, and J. Tang, "Graph convolutional networks with eigenpooling," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019.

[16] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*. Springer, 2018, pp. 593–607.

[17] D. Nathani, J. Chauhan, C. Sharma, and M. Kaul, "Learning attention-based embeddings for relation prediction in knowledge graphs," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

[18] M. Kampffmeyer, Y. Chen, X. Liang, H. Wang, Y. Zhang, and E. P. Xing, "Rethinking knowledge graph propagation for zero-shot learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 487–11 496.

[19] C.-W. Lee, W. Fang, C.-K. Yeh, and Y.-C. Frank Wang, "Multi-label zero-shot learning with structured knowledge graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1576–1585.

[20] S. K. Sahu, F. Christopoulou, M. Miwa, and S. Ananiadou, "Inter-sentence relation extraction with document-level graph convolutional neural network," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

[21] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, no. 1, pp. 76–80, 2003.

[22] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 143–177, 2004.

[23] R. Xiang, J. Neville, and M. Rogati, "Modeling relationship strength in online social networks," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 981–990.

[24] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *The World Wide Web Conference*, ser. WWW '19. ACM, 2019, pp. 417–426.

[25] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang, "Irgan: A minimax game for unifying generative and discriminative information retrieval models," in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2017, pp. 515–524.

[26] C. Chen, M. Zhang, Y. Liu, and S. Ma, "Neural attentional rating regression with review-level explanations," in *Proceedings of the 27th International Conference on World Wide Web*, 2018, pp. 1583–1592.

[27] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1480–1489.

[28] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, vol. 27, no. 1, pp. 415–444, 2001.

[29] P. V. Marsden and N. E. Friedkin, "Network studies of social influence," *Sociological Methods & Research*, vol. 22, no. 1, pp. 127–151, 1993.

[30] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop, coursera: Neural networks for machine learning," *University of Toronto, Technical Report*, 2012.

[31] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[32] B. M. Sarwar, G. Karypis, J. A. Konstan, J. Riedl *et al.*, "Item-based collaborative filtering recommendation algorithms." *WWW*, vol. 1, pp. 285–295, 2001.

[33] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 426–434.

[34] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *21th Conference on Neural Information Processing Systems*, vol. 1, no. 1, 2007, pp. 2–1.

[35] H. Ma, H. Yang, M. R. Lyu, and I. King, "Sorec: social recommendation using probabilistic matrix factorization," in *Proceedings of the 17th ACM conference on Information and Knowledge Management*. ACM, 2008, pp. 931–940.

[36] B. Yang, Y. Lei, J. Liu, and W. Li, "Social collaborative filtering by trust," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 8, pp. 1633–1647, 2017.

[37] W. Fan, Q. Li, and M. Cheng, "Deep modeling of social relations for recommendation," in *AAAI*, 2018.

[38] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*, 2017.

[39] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 855–864.

[40] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 2019, pp. 165–174.

[41] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 2009, pp. 452–461.

[42] S. Purushotham, Y. Liu, and C.-C. J. Kuo, "Collaborative topic regression with social matrix factorization for recommendation systems," in *Proceedings of the 24th International Conference on Machine Learning*, 2012.

[43] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[44] Z. Li, J. Tang, and T. Mei, "Deep collaborative embedding for social image understanding," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 9, pp. 2070–2083, 2018.

[45] N. Wang and D.-Y. Yeung, "Learning a deep compact image representation for visual tracking," in *Advances in Neural Information Processing Systems*, 2013, pp. 809–817.

[46] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.

[47] A. Van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Advances in neural Information Processing Systems*, 2013, pp. 2643–2651.

[48] X. Wang and Y. Wang, "Improving content-based and hybrid music recommendation using deep learning," in *Proceedings of the 22nd ACM International Conference on Multimedia*. ACM, 2014, pp. 627–636.

[49] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21th ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining. ACM, 2015, pp. 1235–1244.

[50] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 233–240.

[51] S. Wang, Y. Wang, J. Tang, K. Shu, S. Ranganath, and H. Liu, "What your images reveal: Exploiting visual contents for point-of-interest recommendation," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 391–400.

[52] X. Wang, X. He, L. Nie, and T.-S. Chua, "Item silk road: Recommending items from information domains to social users," in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2017, pp. 185–194.

[53] S. Deng, L. Huang, G. Xu, X. Wu, and Z. Wu, "On deep learning for trust-aware recommendations in social networks," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 5, pp. 1164–1177, 2017.

[54] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

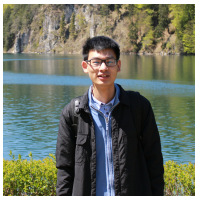
[55] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[56] F. Monti, M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 3700–3710.

[57] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *KDD '18*. ACM, 2018, pp. 974–983.

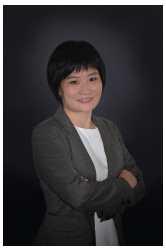
[58] Y. Wei, X. Wang, L. Nie, X. He, R. Hong, and T.-S. Chua, "Mmgcn: Multi-modal graph convolution network for personalized recommendation of micro-video," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 1437–1445.

Wenqi Fan is a third-year Ph.D. student of Computer Science at City University of Hong Kong (CityU). Currently, he is also a visiting student of Data Science and Engineering (DSE) Lab at Michigan State University, under the supervision of Dr.Jiliang Tang. His research mainly focuses on recommender systems, especially on deep neural networks for social recommendation. He has published innovative papers in top-tier conferences such as WWW, IJCAI, RecSys, WSDM, and SDM. Updated information can be found at https://wenqifan03.github.io/.



Yao Ma is a Ph.D. student of Computer Science and Engineering at Michigan State University. He also works as a research assistant at the Data Science and Engineering lab (DSE lab) led by Dr. Jiliang Tang. His research interests include network embedding and graph neural networks for representation learning on graph-structured data. He has published innovative works in top-tier conferences such as WSDM, ASONAM, ICDM, SDM, WWW, KDD and IJCAI. Before joining Michigan State University, he obtained his masters degree from Eindhoven University of Technology and bachelors degree from Zhejiang University. Updated information can be found at http://cse.msu.edu/∼mayao4/.

Qing Li received the B.Eng. degree from Hunan University, Changsha, China, and the M.Sc. and Ph.D. degrees from the University of Southern California, Los Angeles, all in computer science. He is currently a Chair Professor at the Department of Computing, the Hong Kong Polytechnic University. His research interests include object modeling, multimedia databases, social media, and recommender systems. He is a Fellow of IET, a senior member of IEEE, a member of ACM SIGMOD and IEEE Technical Committee on Data Engineering. He is the chairperson of the Hong Kong Web Society and is a steering committee member of DASFAA, ICWL, and WISE Society.



Jianping Wang received the BS and the MS degrees in computer science from Nankai University, Tianjin, China in 1996 and 1999, respectively, and the PhD degree in computer science from the University of Texas at Dallas in 2003. She is a professor in the Department of Computer Science at the City University of Hong Kong. Her research interests include dependable networking, optical networks, cloud computing, service oriented networking, and data center networks.

Jiliang Tang is assistant professor in the computer science and engineering department at Michigan State University since Fall@2016. Before that, he was a research scientist in Yahoo Research and got his PhD from Arizona State University in 2015. His research interests including social computing, data mining and machine learning and their applications in education. He was the recipients of 2019 NSF Career Award, the 2015 KDD Best Dissertation runner up and 6 best paper awards (or runner-ups) including WSDM2018 and KDD2016. He serves as conference organizers (e.g., KDD, WSDM and SDM) and journal editors (e.g., TKDD). He has published his research in highly ranked journals and top conference proceedings, which received thousands of citations and extensive media coverage.

Dawei Yin is Senior Director at JD.com. He is managing the recommendation engineering team, building the recommender systems of JD.com. He also founded Data Science Lab, leading the science efforts of recommendation, search, metrics and knowledge graph, etc.. Prior to joining JD.com, he was Senior Research Manager at Yahoo Labs, leading relevance science team and in charge of Core Search Relevance of Yahoo Search. He obtained Ph.D. (2013), M.S. (2010) from Lehigh University and B.S. (2006) from Shandong University. From 2007 to 2008, he was an M.Phil. student in The University of Hong Kong. His research interests include data mining, applied machine learning, information retrieval and recommender system. He published more than 70 research papers in premium conferences and journals, and was the recipients of WSDM2016 Best Paper Award, KDD2016 Best Paper Award, WSDM2018 Best Student Paper Award, and ICHI 2019 Best Paper Honorable Mention.