Enhancing Session-based Social Recommendation through Item Graph Embedding and Contextual Friendship Modeling

Pan Gu[a], Yuqiang Han[b,*], Wei Gao[b], Guandong Xu[c], Jian Wu[b]

[a] *College of Modern Science and Technology, China Jiliang University, Hangzhou, China*
[b] *College of Computer Science and Technology, Zhejiang University, Hangzhou, China*
[c] *Advanced Analytics Institute, University of Technology Sydney, Sydney, Australia*

Abstract

Recommender systems are designed to help users find matching items from plenty of candidates in online platforms. In many online platforms, such as Yelp and Epinions, users' behaviors are constantly recorded over time, and the users also can build connections with others and share their interests. Previous recommendation methods have either modeled the dynamic interests or the dynamic social influences. A few studies have focused on the modeling of both factors, but they still have several limitations: 1) they fail to consider the complex items transitions among all session sequences, which can be used as a local factor to boost the performance of recommendation methods, and 2) they ignore that a user and their friends only share the same preferences in certain sessions, by keeping the friend vector unchanged for all target users at time t, and 3) they do not consider that a user's long-term preference may change with the evolution of interests.

To overcome the above issues, in this paper, we propose an approach to incorporate item graph embedding and contextual friendship modeling into the recommendation task. Specifically, 1) we construct a directed item graph based on all historical session sequences and utilize a graph neural network to capture the rich local dependency between items, and 2) take a session-level attention

*Corresponding author
Email address: hyq2015@zju.edu.cn (Yuqiang Han)

mechanism to get each friend's representation according to the target user's current interests, and 3) apply max-pooling on the target user's historical session interests to learn the dynamics of his/her long-term interests. Extensive experiments on two real-world datasets show that our proposed model outperforms state-of-the-art methods consistently on various evaluation metrics.

Keywords: Session-based recommendation, Social recommendation, Graph convolutional networks

1. Introduction

With the emergence and popularity of online services, recommender systems have become fundamental for helping users find the matching items from plenty of candidates accumulated on the platforms, e.g., e-commerce, media streaming sites, search engines.

Most traditional methods, including the content-based and collaborative filtering methods [12, 15, 16], only capture the users' general preferences by modeling the static user-item interactions. In practice, users' profiles and activities are constantly recorded on most of the online platforms, and the sequential behaviors reflect the evolution of users' preferences over time. Therefore, the sequence-aware recommender systems have attracted increasing attention in recent years, such as session-based recommendation [42, 30, 10, 17]. Here, a session is a group of interactions that take place within a given time frame. Many e-commerce recommender systems and most of the news and media sites do not typically track the user number, and in certain domains the users often show time-varying preferences and session-based traits [31]. Thus subsequent sessions of the same user should be handled independently [10]. Moreover, many online services, such as Yelp and Epinions, enable users to build connections with others, and share their interests and experiences. Many social-aware methods have been proposed to take the social influences into consideration for tackling the data sparsity issue and further improving recommendation performance [19, 13, 27].

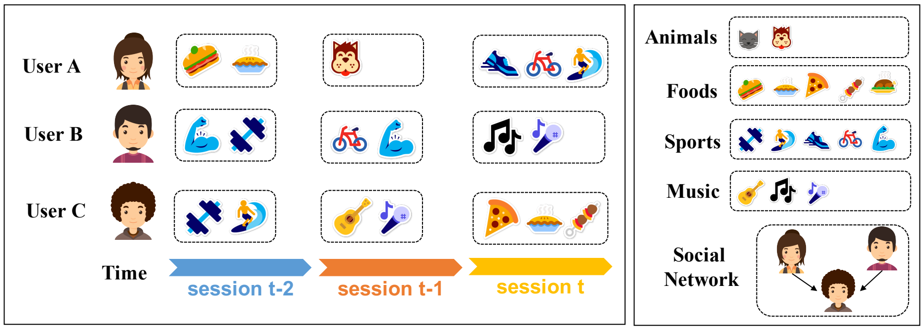


Figure 1: A possible illustration of user interests in social network. A session represents a user' specific interest. User C is a friend of user A and user B, but the common session preferences are not exactly the same. User A and user C share foods and sports; while user B and user C share sports and music. At session t, user A will pay the most attention to user C's session $t-2$, while user B will pay the most attention to user C's session $t-1$.

Although the methods above achieve satisfactory results, they model either users' dynamic interests or the social influences. However, the users' dynamic interests and the social influences can be complementary, and jointly modeling them can boost the performance of recommender systems. Recently, Song et al. [31] proposed an approach to model both users' session-based interests as well as dynamic social influences. They named the problem as *session-based social recommendation*, and it can be defined as follows: given a user's current behavior session, past behavior sessions, and the social relationships, predict the item s/he will interact with (e.g., purchase, visit) next within the session. Though achieving promising results, the study still has several limitations. **First**, it fails to consider the complex item transitions among all session sequences, which can be used as a local factor to boost the performance of session-based recommendation methods [34, 36]. **Second**, although considering the friend-level dynamic social influences, it keeps the friend vector unchanged when facing different target users at time t. However, target users and their friends only have the same preferences in certain sessions. For example, as shown in Figure 1, user C is a friend of user A and user B, but the common session preferences are not exactly the same. User A and user C share foods and sports; while user B and user C share sports and music. At session t, user A will pay the most attention to user C's session $t-2$, while user B will pay the most attention to user C's session $t-1$. Therefore, it is more reasonable to represent a friend dynamically corresponding to the target user's current interest. **Third**, it learns a static long-term preference representation for each user, but we argue that a user's long-term preference may change with the evolution of interests, for example, a user may be interested in a new sport s/he has never participated in. And it can also conduce to improving recommendation in addition to short-term interest and dynamic social influence.

To overcome the limitations mentioned above, in this paper, we propose an approach to solve the *session-based social recommendation* problem through item graph embedding and contextual friendship modeling. **Firstly**, we construct a directed item graph based on all historical session sequences before time t, and utilize GraphSAGE [9] to capture complex transitions of items to learn the latent vectors for all nodes (items). This layer maps the item ids into dense latent vectors. This also can mitigate the cold-start problem to some extent in recommender systems. **Secondly**, we use a Long Short Term Memory network(LSTM) [11] to model a user's short-term interest from the current session, and take the last hidden state of LSTM as the session interest representation. Then we apply max-pooling operation on his/her recent historical session interests to represent long-term interest, which can capture the dynamics of his/her long-term preference. **Thirdly**, to better characterize a user's preferences from the social aspect, we first take a session-level attention mechanism to get each friend's representation according to the target user's current interests, and then use GraphSAGE to aggregate informative friends to capture the dynamic social influences. That is, we compute the similarities between the target user's current interest and each friend's recent K session interests, and represent each friend as the weighted sum of the K interests with the similarities. The session-level attention mechanism is controlled by the target user's current interest and each friend's historical interests, making the dynamic friend's representation correspond to the shared interests between the target user and each friend. Then, we use GraphSAGE to automatically select informative friends. **Finally**, we combine the session-based short-term interest, long-term interest, and dynamic social influence by a fully connected layer for user preference modeling. In summary, the main contributions of this work are as follows:

- We capture the rich transitions among items by building an item graph based on all historical session sequences.
- We introduce the session-level attention mechanism to get a friend's different representations according to different target users' current interests at each step.
- We jointly model a user's session-based short-term interest, long-term interest, and dynamic social influence to infer his/her preference.
- We conduct extensive experiments for performance evaluation on two real-world datasets from Douban Movie and Delicious. The experimental results well justify the effectiveness and superiority of the proposed model.

Organization. The rest of the paper is organized as follows. In Section 2, we discuss the related work. In Section 3, we present the proposed model in detail. In Section 4, we present the experimental settings and the analysis of experimental results. In Section 5, we conclude the paper.

2. Related Work

In this section, we briefly review three lines of work closely related to ours: session-based recommendation, social recommendation, and graph convolution network.

2.1. Session-based Recommendation

Session-based recommendations are a typical task of recommender systems based on implicit feedback. Conventional methods are usually based on Markov chains that predict the next behavior given the previous ones. Zimdars et al. [42] applied Hidden Markov models(HMMS) to time-order information and achieve promising experimental results. Zimdars et al. [42] also investigated how to extract sequential patterns to learn the next state by using a standard decision tree. Mobasher et al. [22] studied sequential and non-sequential patterns in predictive tasks. They showed that sequential patterns extracted by kth-order Markov models are more suitable for predicting which item is accessed next by a user. Shani et al. [30] described a new model for recommender systems based on an Markov decision process (MDP)[25]. They experimented with several enhancements to the maximum-likelihood model, including skipping, clustering, and the use of finite mixture modeling. Chen et al. [3] presented Latent Markov Embedding (LME) for automatically embedding songs in Euclidean space, helping achieve the goal of automated playlist generation. However, the state space of Markov chains will quickly become unmanageable when trying to include all possible sequences of potential user's selections over all items.

Later on, RNNs have proved effective for sequential click prediction [40]. Hidasi et al. [10] firstly attempted to apply RNNs for solving the session-based recommendation. Since then, many methods have been presented to improve the performance of RNNs for session-based recommendations. Tan et al. [32] applied some popular approaches including data augmentation, dropout, batch normalization, and residual connections to enhance the performance of RNNs for session-based recommendations. Li et al. [17] introduced a neural attentive recommendation machine(NARM), which can capture both the user's sequential behavior and main purpose by incorporating an attention mechanism into RNN. Ren et al. [26] emphasized the repeat consumption phenomenon and proposed a model called RepeatNet which incorporates a repeat-explore mechanism into neural networks.

These RNNs based work revealed that the patterns in item transitions are important. But the complex transitions among distant items and the space structure of items are ignored. Thus, graph neural network [34, 36] has been employed in session-based recommendations. Wu et al. [34] proposed a method for session-based recommendations with graph neural networks(SR-GNN), which explored complex structure among items and generate accurate latent vectors of items by constructing the session graph. Based on this work, Xu et al. [36] provided an enhanced graph neural network for session-based recommendations by self-attention network. The self-attention mechanism is capable of capturing long-range dependencies among items regardless of the distance. Due to the benefits of model GNN, we also adopt it in our session-based model. SR-GNN constructed item graph from batch sessions, which means that the sparsity of the item graph depends on batch-size set in experiments. Differently, we construct an item graph in a more suitable method, which will be discussed in Section 3.

2.2. Social Recommendation

In recent years, considerable attention has been paid to users' social connections for improving recommendation performance [27]. Most existing social recommendation algorithms are based on matrix factorization methods [19, 24, 37], which factorize both the user-item network and the user-user network. This brings the latent preferences of connected users closer to each other. Specifically, Ma et al. [19] proposed a social regularization method (SoRec) to consider the constraint of social relationships. The idea is to share a common latent user feature matrix factorized by ratings and by trust. Ma et al. [20] then proposed a social trust ensemble method (RSTE), which achieved a balance between users' taste and trusted friends' favors. Jamali et al. [13] showed a matrix factorization based model in social rating networks(SocialMF) by reformulating the contributions of trusted users to the formation of the target users vector. That is, the feature vector of each user is dependent on the feature vectors of his direct neighbors in the social network. Guo et al. [8] proposed a novel trust-based recommendation model(TrustSVD), where both the explicit and implicit influence of user-item ratings are involved to generate predictions. Chaney et al. [2] developed a social poisson factorization model(SPF), accounting for the social aspect of how users consume items. Yang et al. [37] studied a matrix factorization based collaborative filtering method(TrustMF) that combines a truster model and a trustee model, that is, mapping users into two low-dimensional spaces, i.e.

truster space and trustee space. The common rationale behind these methods is that users' preferences are similar to their friends. In all of the researches, friends' influence vectors stay the same for different target users. In the real world, the interests of a user are usually dynamic and diverse, and the preferences of a user may partially match his/her friends' interests. While matching with different target users, we may concentrate on the most relevant parts of friends' interests. our model is a fundamentally different approach compared to researches above, which adjusts friends' vector representation corresponding to different target users.

2.3. Graph Convolutional Network

Graph convolutional networks(GCNs) inherit from convolution neural networks, which have been applied to computer vision successfully. GCNs can naturally integrate node information and topological structure, have proved extremely powerful in graph-structured data [4, 5]. Different from conventional random walks based network embedding algorithms [23], Duvenaud et al. [6] first introduced graph convolutional networks(GCNs) for graph-level classification by learning node degree-specific weight matrices. Kipf et al. [14] then introduced an efficient variant of GCN for semi-supervised node representation learning, which can scale to large graphs. GCNs pass messages between each node and its neighbors to extract local substructure features around nodes. Unfortunately, GCNs require that the full graph Laplacian is known. To overcome that, Hamilton et al. [9] presented GraphSAGE methods based on GCNs, which generated embeddings by sampling and aggregating features from node's local neighborhoods. Moreover, GraphSAGE can efficiently generate embeddings for previously unseen nodes and scale to large graphs.

GCNs have been applied in recommender systems by taking items and users as nodes. Graph-based recommender systems can leverage both content information as well as graph structure. Ying et al. [39] developed a data-efficient graph convolution network method called PinSage, which combined efficient random walks and graph convolutions to generate embeddings of nodes for web-scale recommendations. Wu et al. [34] proposed a session-based recommendation with graph neural networks(SR-GNN), that utilized GCN in sequence prediction task. Based on the session graph, GCN can explore rich transitions among items to get accurate latent vectors of items. Fan et al. [7] presented a novel graph neural network framework (GraphRec), which combined the user-item graph, user-user social graph, and the user-item graph for social recommendations. Song et al. [31] proposed a graph-attention social network(DGRec), which utilized a social graph to pass the information between users and their friends. The influence of each friend is decided on the similarity with the target user. In terms of algorithm design, our work follows this pipeline but contributes in that: 1) We fundamentally improve upon DGRec by introducing an item graph to extract item transitions. 2) We adopt an attention mechanism in learning embeddings of friends. 3) We also show a new method of embedding users' long-term interests to improve performance.

3. Proposed Model

In this section, we will first introduce the definitions and notations used in this paper, next give an overview of the proposed framework, namely ***E****nhancing session-based social recommendation through item* ***G****raph embedding and contextual* ***F****riendship modeling*(EGFRec). Then we describe each model component in detail and finally go through the model optimization.

3.1. Definitions and Notations

In online platforms, users' behaviors are recorded constantly, and preferences of users evolve rapidly. A common approach to catch time-varying interests of users is to segment users' behaviors into sequential sessions and recommend at session level[10]. Here we give a formulation of the session-based social recommendation problem as below.

Let $U = \{u_1, u_2, \cdots, u_n\}$ and $V = \{v_1, v_2, \cdots, v_m\}$ denote the set of users and items involved in all sessions respectively, where n is the number of users, and m is the number of items. $G = (U, E)$ is the social network, where E is the set of social links between users. All sessions by the user u_i are denoted as $S(i) = \left\{s_i^1, s_i^2, \cdots, s_i^t\right\}$, where s_i^t is the t-th session. Within each session, the sequence is denoted as $\left\{v_i^{t,1}, v_i^{t,2}, \cdots, v_i^{t,n}\right\}$, where $v_i^{t,p} \in V$ represents p-th clicked item of the user u_i in t-th session. Formally, our model aims to predict the next item $v_i^{t,n+1}$ that matches the user u_i's preferences the most for a given session $s_i^t = \left\{v_i^{t,1}, v_i^{t,2}, \cdots, v_i^{t,n}\right\}$, social network G and historical session behaviors $S(i)$. The mathematical notations used in this paper are summarized in Table 1.

3.2. Proposed Method

As shown in Figure 2, the proposed model EGFRec contains four key components. The first component is item embedding based on graph convolution network, which is to learn latent factors of items. Secondly, we use a recurrent neural network (RNN) to model the sequence of items consumed in target users' current session, which can capture the time-evolving dynamic interests of users. Meanwhile, we argue that long-term preferences of each user keep evolving over time, but is relatively stable compared with short-term interests. We use a max-pooling layer to aggregate recent session vectors to model long-term interests of users. Thirdly, friends' interests are modeled using an attention network, which depends on the matching target users' current session and the friends' historical behaviors. Fourthly, the representation of the users' current session and friends are fed to a user-user social network to learn the social influence. At the final step, the model generates recommendations by concatenating users' short-term, long-term preferences and social influences.

3.2.1. Item Embedding on Item Graph

To take complex transitions among distant items into account, we construct an item graph from all sessions. Given a session $s = \{v_1, v_2, \cdots, v_m\}$, we treat each item v_j as a graph node and (v_i, v_j) as an incoming edge which means that a user clicks item v_j after item v_i. In-node v_i of v_j may appear repeatedly,

Table 1: Notations

Notations	Definitions and Descriptions
U	User set, $\|U\| = n$
V	Item set, $\|V\| = m$
G	The user-user social graph
T	The item-item graph
$N(i)$	The set of social friends of user u_i
$B(j)$	The set of item neighbours of item v_j in item-item graph, $\|B(j)\| = k$
$S(i)$	The set of sessions of user u_i
s_i^t	The t-th session of user u_i in time t
$\mathbf{p}_i^s$	The short-term interests embedding of user u_i
$\mathbf{p}_i^l$	The long-term interests embedding of user u_i
$\mathbf{q}_j$	The embedding of item v_j produced by GCN
$\mathbf{z}_{s_i}$	The embedding of session s_i
$\mathbf{x}_j$	The input vector of item v_j
$\mathbf{Q}$	The latent matrix of item via GCN
$\mathbf{h}_j^k$	The latent factor of item v_j in k^{th} layer of GCN
$\mathbf{h}_i^k$	The latent factor of user u_i in k^{th} layer of GCN
$\mathbf{h}_i$	The latent factor of user u_i's social influences
$\alpha_{(i,j)}$	The attention weight between the target user u_i and j^{th} session of his/her friend
$\mathbf{f}_{(i,l)}$	The embedding of l-th friend of target user u_i
$\oplus$	The concatenation operator of two vectors

which means that node v_j are more likely to follow v_i. Hence we assign each in-node v_i of the node v_j with the frequency of occurrence. To capture a directed transition from one item to another, we utilize solely in-node neighborhoods of each node when generating node features. In such settings, the training process is equivalent to the undirected graph. So we use GraphSAGE [9] to obtain accurate item embedding, which leverages a batch-training algorithm to save

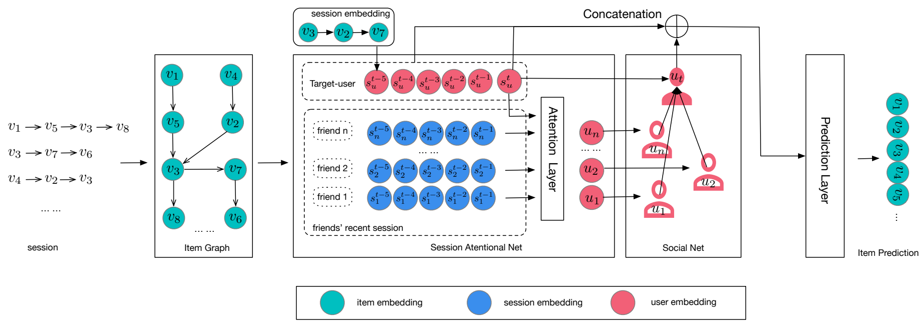


Figure 2: The overall architecture of the proposed model EGFRec. It contains four main components: item embedding, user long-term and short-term interests modeling, contextual friends modeling and social aggregation.

memory.

We then present how to train item latent vectors in our model via GraphSAGE. We first embed every item $v \in V$ into an unified low-dimension latent space, and node vector $\mathbf{x}_j \in \mathbb{R}^d$ denotes a d-dimensional latent vector of item v_j. Then we feed item latent vectors into the GraphSAGE [9], which updates item embeddings by sampling and aggregating features from a node's in-node neighborhood. Sampling strategy defines the neighbourhood $B(j)$ of the node v_j. And we set the $B(j)$ as the top-K neighbours sorted by the frequency of occurrence descendingly. For the node v_j of item-item graph T, the update functions of aggregator are given as follows:

$$\mathbf{h}^k_{B(j)} \leftarrow \mathrm{AGGREGATE}_k\left(\left\{\mathbf{h}^{k-1}_v, \forall v \in B(j)\right\}\right) \tag{1}$$

$$\mathbf{h}^k_j \leftarrow \sigma(\mathbf{W}^k \cdot \mathrm{CONCAT}(\mathbf{h}^{k-1}_j, \mathbf{h}^k_{B(j)})) \tag{2}$$

where $k \in \{1, 2, \cdots, K\}$ denotes the depth of the search, and $\mathbf{W}^k$ determines how nodes in the graph communicate with each other. $\mathbf{h}^k_j$ represents the latent vector of node v_j in k-depth. For the input $\mathbf{h}^0_j$ in the first depth, the feature vector $\mathbf{x}_j$ is passed in. $\sigma(\cdot)$ is the sigmoid function. The embedding of item v_j lastly is defined as $\mathbf{q}_j \leftarrow \mathbf{h}^K_j$. The aggregator architecture $\mathrm{AGGREGATE}_k$ we adopted is *max-pooling* method, defined as follow:

$$\mathrm{AGGREGATE}_k = \max(\left\{\sigma(\mathbf{W}_{\mathrm{pool}}\mathbf{h}^k_v + \mathbf{b}), \forall v \in B(j)\right\}) \tag{3}$$

In this pooling approach, max denotes an element-wise max operation, which effectively captures different aspects of the neighborhood. Due to the sampling and aggregating scheme, GraphSAGE does not conduct the full-batch training and is more suitable for large graphs. In addition, GraphSAGE can effectively and naturally generate representations for unseen nodes.

3.2.2. RNN-based Users' Interests Representation

After feeding session sequences into the graph convolution network, we obtain the latent vectors of items. Then, we adopt a recurrent neural network(RNN) to model sessions of target users. RNN is a standard method for modeling sequences.

To capture users' short-term interests, a LSTM is used to model current session $s_i = \left\{v_i^1, v_i^2, \cdots, v_i^n\right\}$ for target user u_i. The short-term representation $\mathbf{p}_i^s$ of user u_i is the embedding vector $\mathbf{z}_{s_i}$ of session s_i, which can be defined as follows:

$$\mathbf{p}_i^s = \mathbf{z}_{s_i} = \mathrm{LSTM}(\mathbf{Q}(:, s_i)) \tag{4}$$

where $\mathbf{Q}$ is the latent matrix of items got from GraphSAGE, and $\mathbf{Q}(:, s_i)$ denotes all item latent vectors that appear in s_i. Relying only on the user's current sequential behaviors is unreliable when a user accidentally clicks on wrong items. Therefore, we consider both the user's short-term preferences and long-term preferences from historical sessions. The interval between user's historical sessions may be large, so the sequential patterns in them can be ignored. We ignore the time attributes in historical sessions and adopt a pooling operation that transform user u_i recent historical sessions $S(i) = \left\{s_i^1, s_i^2, \cdots, s_i^{t-1}\right\}$ into a fixed-size latent representation $\mathbf{p}_i^l$ as:

$$\mathbf{p}_i^l = \mathrm{POOLING}(\{\mathbf{z}_{s_i}, \forall s_i \in S(i)\}) \tag{5}$$

Common pooling operations include the max-pooling and mean-pooling. In practise, we find there are no significant difference between max-pooling and mean-pooling, hence we focus on max-pooling in the following experiments.

3.2.3. Attention-based Friends' Interests Representation

In online social communities, user's interests are time-varying and diverse, and target users and their friends only have the same preferences in certain sessions. Existing methods assume that the latent vector of a friend's interests keeps the same for different target users [38, 29, 31], which is inconsistent with reality. To be specific, DGRec [31] uses the latest session to represent friends' preferences. Here we present a different approach to effectively generate friends' vector according to the target user. For different users at session s^t, we focus on corresponding parts in friends' historical sessions.

The input of this module is a user-friend pair $(u_i, u_{(i,l)})$, where u_i denotes i-th user, and $u_{(i,l)}$ denotes the l-th friend of i-th user. The output of the module is the vector $\mathbf{f}_{(i,l)}$, which represents the relationship between u_i and $u_{(i,l)}$. The friend vector $\mathbf{f}_{(i,l)}$ of user u_i is generated using a neural attention mechanism. The attention mechanism has been successfully applied to many fields, such as computer vision, machine translation and social recommendations [31]. Here the target of session-level attention is to assign non-uniform weights to friend's historical sessions, and the weights are varied when the friend interacts with different target users. For a target user's current session s_i^t, one of his/her l-th friend's recent sessions are $S_l^t = \left\{s_l^1, s_l^2, \cdots, s_l^{t-1}\right\}$. where s_l^j is modeled using the LSTM mentioned above, which share the same weights with the target user's session embedding. Each element of the attention vector is defined as:

$$\alpha_{(i,j)} = \mathbf{q}^\top \cdot \sigma(\mathbf{W}_1 \cdot \mathbf{p}_i^s + \mathbf{W}_2 \cdot \mathbf{z}_{s_l^j} + \mathbf{c}) \tag{6}$$

$$\mathbf{f}_{(i,l)} = \sum_{j=1}^{t-1} \alpha_{(i,j)} \cdot \mathbf{z}_{s_l^j} \tag{7}$$

where vector $\mathbf{p}_i^s$ is the short-term interests of target user u_i, and vector $\mathbf{z}_{s_l^j}$ is the representation of l-th friend's session s_l^j. Parameters $\mathbf{q} \in \mathbb{R}^d$ and $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d\times d}$ control the weights of session embedding vectors, and d is the dimension of session embedding. Finally, we generate the friend vector $\mathbf{f}_{(i,l)}$, which can be seen as the embedding of l-th friend's influence vector to target user u_i's preferences.

3.2.4. Social Aggregation

In a social platform, a user's preferences are generally similar to his/her directly connected friends. Social aggregation aims to incorporate friends' interests to further model latent factors of users. Firstly, we encode the friendship network in a graph where nodes represent users and edges denote friendship. For the target user, the node is initialized by the representation of short-term interests. Meanwhile, the node of neighbours are represented by friends' interests representation based on attention layer. Secondly, we aggregate the latent factors of neighboring users from the social graph using a message-passing algorithm, namely GraphSAGE. The social-space latent factor $\mathbf{h}_i$ of user u_i is to aggregate the neighbours $N(i)$ of user u_i.

$$\mathbf{h}_{N(i)}^k \leftarrow \text{AGGREGATE}_k \left(\left\{\mathbf{h}_l^{k-1}, \forall l \in N(i)\right\}\right) \tag{8}$$

$$\mathbf{h}_i^k \leftarrow \sigma(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_i^{k-1}, \mathbf{h}_{N(i)}^k)) \tag{9}$$

where $\mathbf{h}_i^0 \leftarrow \mathbf{p}_i^s$ for target user u_i, and $\mathbf{h}_l^0 \leftarrow \mathbf{f}_{(i,l)}$ for l-th friend of user u_i. We assume that users' preferences are only influenced by their direct-connected friends, then k is set with 1. AGGREGATE_k adopt *max-pooling* operation:

$$\text{AGGREGATE}_k = \max\left(\left\{\sigma(\mathbf{W}_{\text{pool}}\mathbf{h}_u^k + \mathbf{b}), \forall u \in N(i)\right\}\right) \tag{10}$$

We also try context-aware graph neural network by attention mechanism like [31], but it does not show increase in performance. Because in subsection 3.2.3, we model friend representation by attention mechanism as $\mathbf{f}_{(i,l)} = \sum_{j=1}^{t-1} \alpha_{(i,j)} \cdot \mathbf{z}_{s_k^j}$. Through parameter α, we can focus on the more relevant parts of friend's interests compared with target user. Furthermore, parameter α can model tie strengths between a user and his/her friends. Normally, users prefer to share more similar tastes with strong ties than weak ties. Therefore, attention-graph neural network is ineffective here.

3.3. Model Learning

With the embedding of target user short-term, long-term embedding and social influences, the final user representation is obtained by combining them using a fully-connected layer:

$$\mathbf{g}_i = \mathbf{W}_3 \cdot \left[\mathbf{p}_i^s \oplus \mathbf{p}_i^l \oplus \mathbf{h}_i\right] \tag{11}$$

where $\mathbf{W}_3$ is a linear transformation matrix, and $\mathbf{p}_i^s$, $\mathbf{p}_i^l$ are the short-term and long-term interests of user respectively. Vector $\mathbf{h}_i$ denotes the social-influenced representation. $\oplus$ is the vectors concatenation operation. Then, we compute the score for each candidate item v_j by multiplying its embedding $\mathbf{q}_j$, and then apply a softmax function to obtain the output:

$$\hat{\mathbf{y}}_j = \mathrm{softmax}\left(\mathbf{g}^\top \cdot \mathbf{q}_j\right) \tag{12}$$

where $\hat{\mathbf{y}}_j$ denotes the predicted probability of item v_j to be the next click. Finally, loss function is defined as the cross-entropy of the prediction result $\hat{\mathbf{y}}$, which can be written as follow:

$$\mathcal{L}(\hat{\mathbf{y}}) = -\sum_{j=1}^{m} \mathbf{y}_j \log\left(\hat{\mathbf{y}}_j\right) + \left(1 - \mathbf{y}_j\right) \log\left(1 - \hat{\mathbf{y}}_j\right), \tag{13}$$

where $\mathbf{y}_j$ denotes the one-hot encoding vector of the ground truth item v_j. The complete algorithm is detailed in Algorithm 1.

4. Experiments

4.1. Experimental Settings

4.1.1. Datasets and Data Preparation

We evaluate the proposed method on two real-world representative datasets collected from well-known online communities, i.e. *Douban Movie*[1] and *Delicious*[2].

- **Douban Movie.** It is an online social network that allows Internet users to share their comments and viewpoints about movies. The dataset contains both the friendship network and rich user review behaviors. Our task is to predict users next review behaviour on movies.

[1] http://www.douban.com

[2] Data set available from `https://grouplens.org/datasets/hetrec-2011/`

Algorithm 1: EGFRec Algorithm

Input: current session s, historical sessions S, social network G, learning rate η, item embedding size K_e, sample size K_s

Output: model parameters Θ

Draw Θ from Normal Distribution $N(0, 0.01)$;
Construct the item graph T from S ;
repeat
    shuffle the set of observations $\{(s_i^t, S(i), G)\}$;
    for *each observation* $(s_i^t, S(i), G)$ **do**
        get item embedding $\mathbf{h}$ according to Eq.(1)-(3) ;
        capture user's short-term interest $\mathbf{p}_i^s$ according to Eq.(4) ;
        capture user's long-term interest $\mathbf{p}_i^l$ according to Eq.(5) ;
        capture friends' interests $\mathbf{f_i}$ according to Eq.(6)-(7) ;
        compute the final representation according to Eq.(8)-(11) ;
        compute $\hat{y}_j$ according to Eq.(12) ;
        update Θ with gradient descent ;
    end
until *Convergence*;
return Θ

Table 2: Statistics of experimental datasets after preprocessing.

	Douban	Delicious
# Users	26,438	1,629
# Items	12,591	3,450
# Events	2,747,077	282,482
# Sessions	658,672	66,503
# Social links	77,197	12,571
Start Date	01/12/2008	08/12/2009
End Date	07/22/2016	07/01/2016
Avg. friends/user	3.18	7.75
Avg. events/user	103.91	173.41
Avg. session length	4.17	4.25

- **Delicious.** It is a social bookmarking site where users can save, manage and share web bookmarks. Users can also assign websites a variety of semantic tags and discover new links based on what people in their social network were sharing. The dataset contains tags assigned to the bookmarked URLs and contact relations between users. The task is to recommend tags for bookmarks.

For session-based recommendations, user's behaviors are segmented into week-long sessions in *Douban* dataset following the method in [31]; while a session in *Delicious* dataset is a sequence of tags that a user has assigned to bookmarks. We set the sessions data of last d days as the test data, and the remaining for training. To be specific, we assign $d = 180$ and $d = 25$ for *Douban* and *Delicious* dataset respectively. Considering that we can not recommend an item that has not appeared before, we filter out interactions from the test set where items do not appear in the training set. The statistics of the two datasets after preprocessing are shown in Table 2.

4.1.2. Baselines

To evaluate the performance of our proposed model, we compare it with both state-of-the-art models and traditional methods. These recommendation systems are closely related to our work.

- **ItemKNN** [18]: The method recommends the items similar to previously accessed items for user, based on the cosine item-item similarity.
- **BPR-MF** [28]: The method is a state-of-the-art matrix factorization for non-sequential recommendations, which optimizes a pairwise ranking objective function.
- **SoReg** [21]: The method employs social network information in traditional matrix factorization framework. More specifically, SoReg designs two social regularization terms to constrain the matrix factorization objective function.
- **SBPR** [41]: The method considers social connections to better estimate users' rankings of products, assuming that users tend to assign higher ranks to items that their friends prefer.
- **TranSIV** [35]: The model is a probabilistic generative model for item recommendations, which uses visibility to model user-item and user-user interactions simultaneously. Moreover, the method adopts a transfer learning strategy to coordinate social and rating information in a unified model.
- **GRU4Rec** [10]: The model applies recurrent neural networks (RNNs) on short session-based recommender systems.
- **NARM** [17]: The model applies an attention mechanism to capture the user's main purpose from the current session, and combines it with the user's sequential behavior as a unified session representation.
- **DGRec** [31]: The model takes into account both the users' session-based interests and dynamic social influences. Specifically, the method models context-dependent social influence with a graph-attention neural network.

4.1.3. Evaluation Metrics

To evaluate the performance of all models, we adopt three common metrics, i.e., Recall@K, Mean Reciprocal Rank(MRR@K) and Normalized Discounted Cumulative Gain (NDCG).

- **Recall@20**[17]. It is the primary evaluation metric that is the proportion of correct results amongst the top-20 items in all test cases , defined as:

$$\text{Recall@20} = \frac{n_{hit}}{N} \tag{14}$$

where N denotes the number of test data in the SRS system O, n_{hit} denotes the number of cases which have the desired items in the top-20 item's list. Note that Recall@20 does not consider the actual rank of the item as long as it is amongst the top-20 item's list.

- **MRR@20**. The second metric used is MRR@20 (Mean Reciprocal Rank) [33], which is the average of reciprocal ranks of the desired items t. The reciprocal rank is manually set to zero if the rank is greater than 20.

$$\text{MRR@20} = \frac{1}{N}\sum_{t\in O}\frac{1}{Rank(t)} \tag{15}$$

- **NDCG**[31]. It is a widely used ranking metric, which considers the hit position of the item and gives a higher score if the hit item in the top position. It is formulated as:

$$\text{NDCG} = \frac{1}{N}\sum_{t\in O}\frac{1}{\log_2(1+r)} \tag{16}$$

where r is the ranks of positive items. We average values of NDCG over all the testing examples.

MRR@20 focuses on the ranks of positive items in the top-20 items, while NDCG pay attention to the ranks amongst the whole test cases.

4.1.4. Parameter Settings

We implement our model based on TensorFlow [1], a well-known python library for deep neural networks. For a fair comparison, the hyper-parameters of all the methods including baselines are optimized via extensive grid search on both of the two datasets, and the best models are selected by early stopping based on the Recall@20 score on the validation set. All parameters of the latent factors and weight matrix are initialized by sampling from the Gaussian distribution $N\left(0, 0.1^2\right)$, and all biases are set to zeros. The dimensions of items embedding and user embedding(when needed) are set to 100. The sampling and aggregating method is applied in our model following GraphSAGE [9], which can reduce the computational cost of training process. The sample sizes of item graph and social network are set to 5, 10 respectively. The search depths of item graph and social network are set with 1. The number of target users' historical sessions as long-term interests is set to 10. Meanwhile, the number of friends' attention session is set as 5. The initial learning rate is set to 0.002 and will decay by 0.98 after every 400 epochs. Optimizing is done by using Adam with a batch size of 200 at each iteration and we adopt dropout with a rate of 0.2 to alleviate overfitting problems. To exclude the impact of random initialization of parameters, we run the evaluation 10 times with the same parameters and report the average results.

4.2. Quantitative Results

We compare the recommendation performance of all methods. The overall predictions in terms of Recall@20, MRR@20 and NDCG are shown in Table 3, with the best results highlighted in boldface. The 95% confidence intervals of our model on Recall@20, MRR@20 and NDCG are (around) $\pm 2.75 \times 10^{-4}, \pm 2.09 \times 10^{-3}, \pm 1.82 \times 10^{-4}$ for Douban and $\pm 2.83 \times 10^{-4}, \pm 9.70 \times 10^{-4}, \pm 9.42 \times 10^{-4}$ for Delicious. As we can see, our model achieves the best performance among all the methods on both datasets in terms of Recall@20, MRR@20 and NDCG. We have the following observations:

- ItemKNN always outperforms matrix factorization based models, like BPR-MF, SoReg, SBPR, and TranSIV on *Douban*. The reason is that users typically only consume each item once on *Douban*, and MF-based methods tend to recommend previously accessed items. While users in *Delicious* prefer to assign websites with same tags. Furthermore, SoReg, SBPR and TranSIV obtain better performance than BPR-MF on *Douban* on Recall@20 metric, which supports that social network information is complementary to interaction information.

- GRU4Rec and NARM perform better than ItemKNN and MF-based methods on both of the datasets. Note that, ItemKNN only utilizes the similarity between items without considering sequential information. The results help verify the importance of session-based recommendations. Moreover, it demonstrates that RNN-based models are good at dealing with sequential information in sessions.

- DGRec achieves the best performance among the baselines, because it not only models the sequential behaviors using RNN but also uses graph-attention neural network to capture social influence. Compared to GRU4Rec and NARM methods, the obvious improvement of DGRec further indicates that social information is necessary for recommendations.

- Our proposed method consistently and significantly outperforms the best baseline. Since two models both integrate interaction and social network information, we can attribute the performance improvement to the proposed attention-based friend vector embedding and item-graph neural network. We will provide further investigations to better understand the contributions of model components in the following subsection.

4.3. Model Analysis

In this subsection, we study the impact of each model component.

4.3.1. Effect of Model Components

In our model EGFRec, users' final representation is a combination of users' short-term session vector, long-term historical behavior vector, and social influence. To understand the working of EGFRec, we compare EGFRec with

Table 3: Performance comparison of different algorithms. The best results are highlighted in boldface.

Methods	Douban			Delicious		
	Recall@20	MRR@20	NDCG	Recall@20	MRR@20	NDCG
ItemKNN	0.1431	0.0687	0.1635	0.2729	0.0966	0.2241
BPR-MF	0.0163	0.0053	0.1110	0.2775	0.1000	0.2293
SoReg	0.0177	0.0064	0.1113	0.2703	0.0931	0.2271
SBPR	0.0171	0.0056	0.1059	0.2948	0.1098	0.2391
TranSIV	0.0173	0.0058	0.1102	0.2588	0.0919	0.2158
GRU4Rec	0.1643	0.0697	0.1854	0.3445	0.1335	0.2581
NARM	0.1755	0.0729	0.1872	0.3776	0.1452	0.2768
DGRec	0.1861	0.0736	0.1950	0.4066	0.1459	0.2944
EGFRec	**0.1923**	**0.0770**	**0.2008**	**0.4181**	**0.1573**	**0.3000**
Improv.	+3.3%	+4.6%	+3.0%	+2.8%	+7.8%	+1.9%

its three variants: $\text{EGFRec}_{\text{short}}$, $\text{EGFRec}_{\text{long}}$ and $\text{EGFRec}_{\text{social}}$. Note that the item-graph structure is not used here. These variants are defined as follows:

- $\textbf{EGFRec}_{\textbf{short}}$: This variant only uses the user's current session, while ignoring the social influence and the user's long-term interests. $\text{EGFRec}_{\text{short}}$ is identical to model GRU4Rec.

- $\textbf{EGFRec}_{\textbf{long}}$: Similar with $\text{EGFRec}_{\text{short}}$, social-graph neural network is removed. $\text{EGFRec}_{\text{long}}$ only utilizes the long-term preference of users.

- $\textbf{EGFRec}_{\textbf{social}}$: This is a model using social graph neural networks only.

The experimental results of different variants on *Douban* and *Delicious* are shown in Table 4. From the results, we have the following observations. Firstly, except to model EGFRec, $\text{EGFRec}_{\text{short}}$ performs best and $\text{EGFRec}_{\text{long}}$ gets the worst performance, which indicates the superior predictive capability of short-term interests. Specially on *Douban* datasets, $\text{EGFRec}_{\text{long}}$ achieves poor performance because users on *Douban* website have dynamic preferences. Secondly, $\text{EGFRec}_{\text{short}}$ slightly outperforms $\text{EGFRec}_{\text{social}}$, which means that users' short-term interests provide a more advanced effect on recommender systems.

Table 4: Performance comparison of EGFRec and three variations.

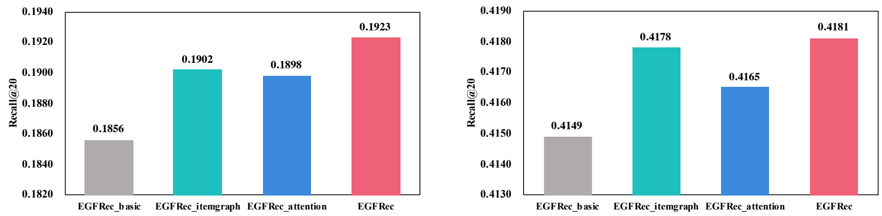
Data Sets	Models	Recall@20	MRR@20	NDCG
Douban	$\text{EGFRec}_{\text{long}}$	0.0591	0.0136	0.1300
	$\text{EGFRec}_{\text{social}}$	0.1806	0.0713	0.1926
	$\text{EGFRec}_{\text{short}}$	0.1845	0.0748	0.1949
	EGFRec	0.1898	0.0763	0.1991
Delicious	$\text{EGFRec}_{\text{long}}$	0.2236	0.1556	0.2058
	$\text{EGFRec}_{\text{social}}$	0.3966	0.1564	0.2916
	$\text{EGFRec}_{\text{short}}$	0.4037	0.1558	0.2970
	EGFRec	0.4165	0.1567	0.3013

Both $\text{EGFRec}_{\text{short}}$ and $\text{EGFRec}_{\text{social}}$ perform worse than model EGFRec, which verifies that social network is crucial to boost the recommendation performance.

4.3.2. Effect of Item-graph and Attention-based Friend Embedding

The key characteristics in our proposed model EGFRec are the two designed enhanced operations: the item-graph that captures more complex and implicit connections between user interactions, and the attention-based friend latent vector embedding that models specific user-friend relationship. We study the impact of each operation on the model's performance. The following variants are designed to compare with EGFRec:

- **$\textbf{EGFRec}_{\textbf{basic}}$**: This is our basic model without the item-graph and attention-layer for friend vector embedding parts. Friends' interests are represented using the session right before session T. And we replace the item-graph neural network with a randomly initialized item embedding.
- **$\textbf{EGFRec}_{\textbf{itemgraph}}$**: This is a variant model utilizing the item-graph neural network only.
- **$\textbf{EGFRec}_{\textbf{attention}}$**: This is a variant model that using an attention-based layer to represent friends' latent vector.
- **EGFRec**: This is our proposed model that integrates both of two operations.



(a) Douban (b) Delicious

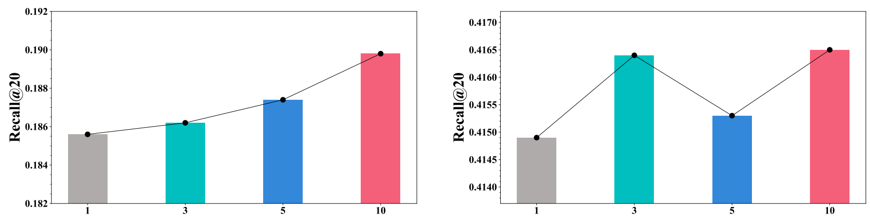
Figure 3: Performance w.r.t. item-graph and attention-layer on different data sets.

Figure 3 shows the performance of different variants on Recall@20 metric. We can observe that model EGFRec$_{\text{attention}}$ outperforms EGFRec$_{\text{basic}}$ on both of the datasets. The result of attention-layer is used for social network initialization, hence the accuracy of friend embedding promotes the impact of social influences. Furthermore, EGFRec$_{\text{itemgraph}}$ generally performs better than EGFRec$_{\text{basic}}$, which proves the power of complex and sequential connections between items captured by graph neural network. Lastly, EGFRec achieves the best performance. This further demonstrates that the graph neural network and attention mechanism are complementary, and play important roles in improving recommendation performance.

4.3.3. Analysis of Model Parameters

To have an in-depth understanding of the proposed model, we investigate the influence of model parameters under the primary evaluation metric Recall@20. To be specific, we make detailed analysis of attention size of friend embedding, user history length for long-term interests modeling, neighborhood sample size in item graph and social network, and search depth in graph.

Effect of Attention Size. We analyze the effect of attention size of friend embedding. Note that the item-graph structure is not used here, to highlight the impact of attention size. Figure 4 presents the performance w.r.t. the length of friend historical sessions of our proposed model on *Douban* and *Delicious* datasets. For *Douban* dataset, when increasing the length of friend historical sessions from 1 to 10, the performance improves significantly. It demonstrates that attention mechanism performs better with long session-sequence lengths.



(a) Douban

(b) Delicious

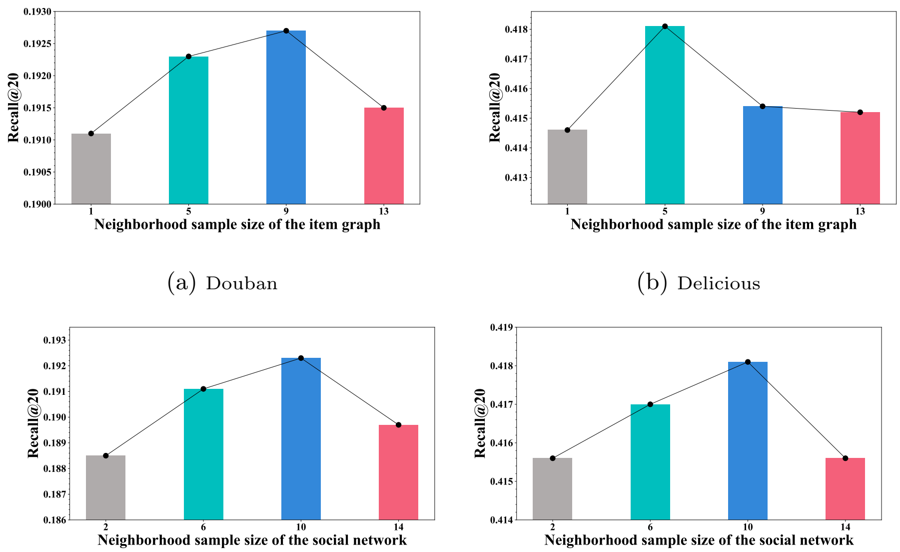
Figure 4: Effect of attention size on different data sets.

However, on the *Delicious* dataset, there is no obvious difference across varied session length. This is because users on *delicious* website trend to have static interests.

Effect of Neighborhood Sample Size. The neighborhood sample size in item graph K_{item} specifies the number of in-nodes of each node to be aggregated. We report the performance patterns of our model EGFRec by tuning K_{item} amongst $\{1, 5, 9, 13\}$ to investigate its effect. As shown in Figure 5a and 5b, the performance first increases and then decreases with K_{item} increasing on both datasets. The optimal K_{item} value is 9 and 5 on *Douban* and *Delicious*, respectively. Considering the small performance variation and computational efficiency, we choose to use $K_{item} = 5$ in our experiments.

In a similar way, we report the performance patterns of our model EGFRec by tuning K_{social} amongst $\{2, 6, 10, 14\}$ to investigate its effect in social network. As shown in Figure 5c and 5d, the optimal value $K_{social} = 10$ is desired on both datasets, which is used in our experiments.

Analysis on the user history length. Considering that a user's long-term preference may change with the evolution of interests, we apply max-pooling on his/her historical session interests to learn the dynamics of long-term interests. Figure 6 shows the performance of EGFRec with different historical session lengths evaluated in terms of Recall@20. We can see that, the performance increases with the number of historical sessions increasing on both datasets.



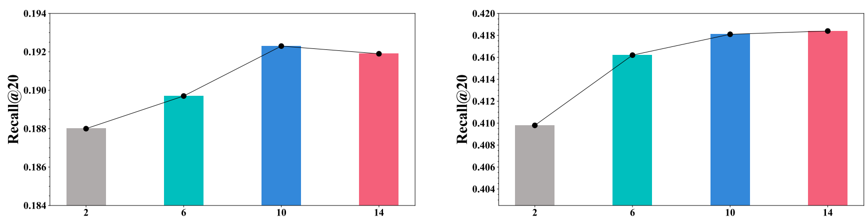
(a) Douban (b) Delicious

(c) Douban (d) Delicious

Figure 5: Effect of neighborhood sample size of the item graph and social network.

When $K_{long} = 14$, there is a slight drop on *Douban* but a small improvement on *Delicious*. This is not surprising since, on *Delicious* website, users tend to have static interests. We use $K_{long} = 10$ in our experiments, which can produce the competitive results.

Effect of Search Depth on Graph. Benefiting from the learning framework of graph neural networks, our method can attain more accurate node embedding vectors and users' interests representations. To investigate the effect of search depth on graph, we report the performances of one- and two-layer networks, while fix the neighborhood sample size to 10 and 5 in social network and item graph respectively. As shown in Table 5, the performance declines slightly in terms of Recall@20 when increasing the search depth both over *Douban* and *Delicious* datasets, which may be attributed to noises or overfitting by higher-order neighborhoods. So we choose to use the one-layer network structure in our experiments. Though only a one-layer network is used, our model can be seen as a heterogeneous two-layer graph by combining the item graph and



(a) Douban (b) Delicious

Figure 6: Effect of the number of target users' historical sessions.

Table 5: Analysis of search depth in terms of Recall@20.

Datasets	Social Network		Item Graph	
	one-layer	two-layer	one-layer	two-layer
Douban	0.1923	0.1917	0.1923	0.1912
Delicious	0.4181	0.4131	0.4181	0.4150

social network. In other words, we aggregate feature information from items' local neighborhoods to update their latent representations. Based on the item representations, we model user's short-term interests in current session, and then automatically select informative friends in the social network to adjust the interests.

4.3.4. Runtime Analysis

GraphSAGE is proposed to alleviate the memory overflow problem in earlier ConvGNNs such as GCN at the cost of sacrificing time efficiency. The time and memory complexity of which is $O(K^r md^2)$ and $O(sK^r d + rd^2)$ respectively, where K is the number of neighbors being sampled for each node, r is the number of layers, m is the total number of nodes, d denotes the dimension of the node hidden features, and s is the batch size. It can be observed that the time and memory complexity of GraphSage grows exponentially with an increase of K and r. However, GraphSAGE can achieve satisfactory performance with very small $r = 1$ and $K = 5$ for the item graph in the proposed model EGFRec,

Table 6: Runtime in Testing Process.

Method	Dataset	Time(seconds)
DGRec	Douban	152
	Delicious	150
$\text{EGFRec}_{\text{itemgraph}}$	Douban	26
	Delicious	22
$\text{EGFRec}_{\text{attention}}$	Douban	27
	Delicious	25
EGFRec	Douban	102
	Delicious	93

which revealed in section 4.3.3.

To further verify the computational efficiency of our proposed model in real-world datasets, we record the runtime of EGFRec and the strong baseline DGRec which also utilizes graph neural networks and is the basis of our model in the test process on the same GPU server. We compare the runtime of variants, i.e. $\text{EGFRec}_{\text{itemgraph}}$ and $\text{EGFRec}_{\text{attention}}$ with the model DGRec as well. Table 6 displays the comparisons between EGFRec and DGRec on two datasets. Firstly, $\text{EGFRec}_{\text{itemgraph}}$ and $\text{EGFRec}_{\text{attention}}$ significantly dominate the model DGRec in terms of the runtime, while outperforming DGRec on primary evaluation metric Recall@20. Next, to facilitate recommendation, we integrate the two variants into our model EGFRec. The runtime rises but is still lower greatly than the model DGRec. In a word, our model is more efficient than DGRec. We argue that this is because of the graph-attention network used in DGRec, which contains more sophisticated operations.

4.3.5. Attention Analysis

Furthermore, we suppose that friends have varying interests, so a user-friend pair only has the same interests in certain aspects. To illustrate the role of the attention mechanism intuitively, we present an example in Figure 7. We randomly choose a user from *Douban* test sets who have at least 9 friends, and

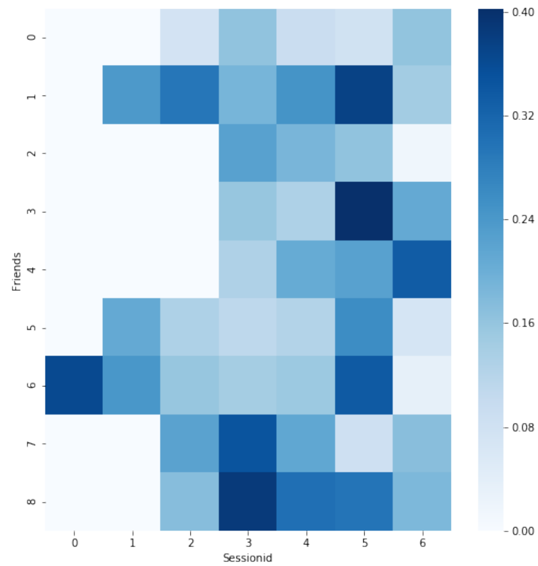


Figure 7: Attention visualization. Attention weights across different friends is used for color-coding. The y-axis represents friends of the target user and the x-axis represents recent sessions of the friends.

plot the attention weight of each friend's recent 7 sessions. The depth of the color corresponds to the importance of friends' sessions and the similarity with the target user, the darker the color the more important a session is. We can observe that for a friend, not all sessions are important for the target user and our model can capture relevant sessions of friends. Further, all sessions of friend #0 obtain little attention, which means that friend #0 recent behaviors are not relevant to the target user's current interest.

5. Conclusion and Future Work

We presented a recommender system based on graph neural network and attention mechanism. There are two main improvement operations: 1) we conducted item graph from user historical interactions. It can involve transitions between items of session sequence, which helps to generate item representations. 2) After generating user representations, we applied session-level attention mechanism to get friends' representations according to the target user's current interest. We also conducted extensive experiments on two real-world datasets. The results demonstrated that our model outperforms state-of-the-art baselines.

In the future, we plan to integrate different types of user behavior into the item graph to generate items' and users' representations. We also plan to study the impact of different types of relationships in multi-relational social networks, like classmates, colleague and normal friends.

6. ACKNOWLEDGMENTS

This work was supported in part by the Subject of the Major Commissioned Project "Research on China's Image in the Big Data" of Zhejiang Province's Social Science Planning Advantage Discipline "Evaluation and Research on the Present Situation of China's Image" No. 16YSXK01ZD-2YB, the Zhejiang University Education Foundation under grants No. K18-511120-004, No. K17-511120-017, and No. K17-518051-021, the Major Scientific Project of Zhejiang Lab under grant No. 2018DG0ZX01, the National Natural Science Foundation of China under grant No. 61672453, and the Key Laboratory of Medical Neurobiology of Zhejiang Province.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] Allison JB Chaney, David M Blei, and Tina Eliassi-Rad. A probabilistic model for using social networks in personalized item recommendation. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 43–50. ACM, 2015.

[3] Shuo Chen, Josh L Moore, Douglas Turnbull, and Thorsten Joachims. Playlist prediction via metric embedding. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 714–722. ACM, 2012.

[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[5] Tyler Derr, Yao Ma, and Jiliang Tang. Signed graph convolutional networks. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 929–934. IEEE, 2018.

[6] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[7] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The World Wide Web Conference*, pages 417–426. ACM, 2019.

[8] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. Trustsvd: collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[9] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[10] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *International Conference on Learning Representations (ICLR)*, 2016.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee, 2008.

[13] Mohsen Jamali and Martin Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 135–142. ACM, 2010.

[14] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[15] Yehuda 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*, pages 426–434. ACM, 2008.

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

[17] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1419–1428. ACM, 2017.

[18] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, (1): 76–80, 2003.

[19] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings*

of the 17th ACM conference on Information and knowledge management, pages 931–940. ACM, 2008.

[20] Hao Ma, Irwin King, and Michael R Lyu. Learning to recommend with social trust ensemble. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 203–210. ACM, 2009.

[21] Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 287–296. ACM, 2011.

[22] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. Using sequential and non-sequential patterns in predictive web usage mining tasks. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 669–672. IEEE, 2002.

[23] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[24] Sanjay Purushotham, Yan Liu, and C.-C. Jay Kuo. Collaborative topic regression with social matrix factorization for recommendation systems. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 691–698. ACM, 2012.

[25] Martin L Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

[26] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten de Rijke. Repeatnet: A repeat aware neural recommendation machine for session-based recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4806–4813, 2019.

[27] Zhaochun Ren, Shangsong Liang, Piji Li, Shuaiqiang Wang, and Maarten de Rijke. Social collaborative viewpoint regression with explainable recommendations. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 485–494. ACM, 2017.

[28] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

[29] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, Lexing Xie, and Darius Braziunas. Low-rank linear cold-start recommendation from social data. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[30] Guy Shani, David Heckerman, and Ronen I Brafman. An mdp-based recommender system. *Journal of Machine Learning Research*, 6(Sep):1265–1295, 2005.

[31] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. Session-based social recommendation via dynamic graph attention networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 555–563. ACM, 2019.

[32] Yong Kiam Tan, Xinxing Xu, and Yong Liu. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 17–22. ACM, 2016.

[33] Ellen M Voorhees et al. The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82, 1999.

[34] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 346–353, 2019.

[35] Lin Xiao, Zhang Min, Zhang Yongfeng, Liu Yiqun, and Ma Shaoping. Learning and transferring social and item visibilities for personalized recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 337–346. ACM, 2017.

[36] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. Graph contextualized self-attention network for session-based recommendation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3940–3946. AAAI Press, 2019.

[37] Bo Yang, Yu Lei, Jiming Liu, and Wenjie Li. Social collaborative filtering by trust. *IEEE transactions on pattern analysis and machine intelligence*, 39(8):1633–1647, 2016.

[38] Mao Ye, Xingjie Liu, and Wang-Chien Lee. Exploring social influence for recommendation: a generative model approach. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 671–680. ACM, 2012.

[39] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983. ACM, 2018.

[40] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. Sequential click prediction for sponsored search with recurrent neural networks. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[41] Tong Zhao, Julian McAuley, and Irwin King. Leveraging social connections to improve personalized ranking for collaborative filtering. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, pages 261–270. ACM, 2014.

[42] Andrew Zimdars, David Maxwell Chickering, and Christopher Meek. Using temporal data for making recommendations. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 580–588. Morgan Kaufmann Publishers Inc., 2001.