Graph-based Embedding Smoothing for Sequential Recommendation

Tianyu Zhu, Leilei Sun, and Guoqing Chen

Abstract—In real-world scenarios, a user's interactions with items could be formalized as a behavior sequence, indicating his/her dynamic and evolutionary preferences. To this end, a series of recent efforts in recommender systems aim at improving recommendation performance by considering the sequential information. However, impacts of sequential behavior on future interactions may vary greatly in different scenarios. Additionally, semantic item relations underlying item attributes have not been well exploited in sequential recommendation models, which could be crucial for measuring item similarities in recommendation. To deal with the above problems, this paper provides a general embedding smoothing framework for sequential recommendation models. Specifically, we first construct a hybrid item graph by fusing sequential item relations derived from user-item interactions with semantic item relations built upon item attributes. Second, we perform graph convolutions on the hybrid item graph to generate smoothed item embedding. Finally, we equip sequential recommendation models with the smoothed item representations to enhance their performances. Experimental results demonstrate that with our embedding smoothing framework, the state-of-the-art sequential recommendation model, SASRec, achieves superior performance to most baseline methods on three real-world datasets. Moreover, the results show that most mainstream sequential recommendation models could benefit from our framework.

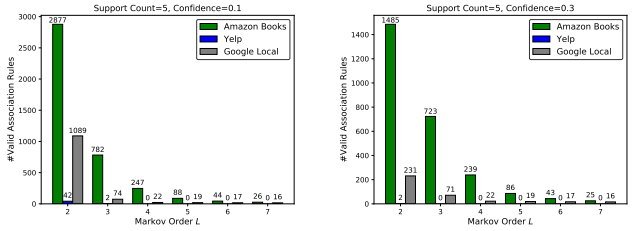
Index Terms—Recommender Systems, Embedding Smoothing, Graph Convolution, Sequential Recommendation.

1 Introduction

Recommender System (RS) has become a crucial technology for e-commerce platforms and other online services in recent years. How to accurately capture user preferences is a core issue in recommendation. In practical applications, a user's interest is usually shifting over time and can be affected by his/her recent interactions. To model the dynamics in user preferences, recent efforts have proposed various sequential architectures (e.g., RNN, CNN, and self-attention [1]) for generating sequential recommendations [2], [3], [4].

Despite the success of sequential models in Natural Language Processing (NLP) and other areas, some issues of concern may arise when they are applied to recommendation. Specifically, although user interactions appear in the form of sequences, the sequential dependency is not necessarily held in some real-world scenarios. For example, when shopping online, a user who has bought a laptop would buy accessories (e.g., a mouse or a keyboard) later. However, when choosing restaurants for dinner, people's choices may not follow a certain order. Therefore, the sequentiality in user interactions may vary greatly in different recommendation scenarios, which may seriously affect the applicability of sequential recommendation models.

To empirically demonstrate the sequentiality in user interactions in different recommendation scenarios, we extract sequential association rules above certain thresholds of support count and confidence [3], [5] in the form of $(S^u_{t-L}, \cdots, S^u_{t-1}) \rightarrow S^u_t$ in three real-world datasets, i.e., Amazon Books, Yelp, and Google Local (see the details of these datasets in Section 5). Figure 1 summarizes the number of rules under the Markov order L with the minimum support count of 5 and the minimum confidence of 10%/30%. It can be seen that user interactions in Amazon Books/Google Local/Yelp show a high/medium/low sequentiality, respectively.



(a) Support=5, Confidence=0.1 (b) Support=5, Confidence=0.3

Fig. 1: The number of sequential association rules w.r.t. the Markov order L in three real-world datasets.

In addition, sequential models only consider sequential item relations in terms of user behavior, while neglecting semantic item relations that are crucial for measuring item similarities in recommendation. Semantic item relations refer to connections between items in terms of attributes or features (e.g., genres for books or locations for POIs). However, these relations are not necessarily captured by sequential models. For example, two movies of the same genre but with different release time may rarely appear adjacent to each other in user interaction sequences. Therefore, it is

- *Tianyu Zhu and Guoqing Chen are with the Department of Management Science and Engineering, Tsinghua University, Beijing, China. E-mail: zhuty.16@sem.tsinghua.edu.cn, chengq@sem.tsinghua.edu.cn*
- *Leilei Sun is with the School of Computer Science, Beihang University, Beijing, China. E-mail: leileisun@buaa.edu.cn*

Manuscript received April 19, 2005; revised August 26, 2015.
(Corresponding author: Leilei Sun.)

important to consider semantic item relations in sequential recommendation models, since they can complement and regularize the item similarities obtained from user interactions, thereby improving recommendation accuracy.

In this paper, we propose an effective framework to address these issues. The main idea is to smooth the item embedding with item relation graphs to maintain the global item similarity structure in sequential models. Specifically, we first construct a hybrid item graph for embedding smoothing. We build a sequential item graph from user-item interactions and construct a semantic item graph through item side information (e.g., item attributes). Then we combine both graphs to generate a hybrid item graph that involves multiple item relations. Second, we perform graph convolutions on the hybrid item graph to generate smoothed item embedding. We point out the drawbacks of using Graph Convolutional Network (GCN) [6] directly and implement a simplified graph convolution by removing the redundant nonlinearity in GCN, one that specifically tailors for recommendation problems. Finally, the smoothed item embedding works as the input of sequential models to generate recommendations. In this way, sequential models involve both user interacted items and their related items when predicting the next item, relaxing the sequential assumptions in these models and enhancing their generalization ability.

Furthermore, we evaluate the proposed embedding smoothing method with a state-of-the-art sequential recommendation model, SASRec [4], on three public datasets. Results of extensive experiments show that the improved SASRec outperforms all baselines on high/medium sequentiality dataset (i.e., Amazon Books/Google Local) and achieves better results on low sequentiality dataset (i.e., Yelp). We also conduct experiments to show how SASRec performs with shuffled user interaction orders and the effect of the proposed method on alleviating the performance degradation. Lastly, experiments are conducted to verify the effectiveness of the proposed method on different categories of sequential recommendation models.

The rest of this paper is organized as follows. Section 2 introduces the preliminaries, Section 3 presents the proposed method in detail, Section 4 discusses the relationship of the proposed method with existing efforts and analyzes the time complexity of the proposed method, Section 5 evaluates the performances of the compared models, and Section 6 reviews the related work. Finally, Section 7 concludes the work with possible future directions.

2 Preliminaries

This section formulates the sequential recommendation problem and presents a brief description of the backbone model SASRec and graph convolutions.

2.1 Problem Formulation

In sequential recommendation, let $U = \{u_1, u_2, \cdots, u_{|U|}\}$ be a set of users, $I = \{i_1, i_2, \cdots, i_{|I|}\}$ be a set of items, and $S^{(u)} = \left(i_1^{(u)}, i_2^{(u)}, \cdots, i_{n_u}^{(u)}\right)$ be the interaction sequence of user u, where n_u denotes the length of the sequence. The goal of sequential recommendation is to predict the next item that user u would interact with, given his historical interactions $S^{(u)}$. It could be formalized as calculating the probability

$$p\left(i_{n_u+1}^{(u)} | S^{(u)}\right). \tag{1}$$

Then the top-N items could be recommended to user u according to the probabilities in a descending order.

2.2 SASRec

Here we concisely introduce the backbone model SASRec [4]. SASRec (short for Self-Attention based Sequential Recommendation) models the sequential dependency through the self-attention architecture in Transformer [1]. Specifically, it first converts user interaction sequences into the vector sequences by embedding look-up. Then a trainable positional embedding is injected into the input embedding, and a self-attention layer with two feed-forward layers is used to encode the sequences. After stacking the self-attention blocks with residual connection, the final representation vector at time step t is treated as the sequence representation. Finally, the similarity score between the sequence representation and the embedding of a candidate item is calculated by dot-product and the cross-entropy loss is adopted for model training.

2.3 Graph Convolution

Since graph convolution is one of the essential components in our proposed method, we introduce it briefly here. Early efforts defined graph convolution in the Fourier domain by computing the eigendecomposition of the normalized graph Laplacian [7]. Then graph convolution can be defined as the multiplication of a signal with a parametric filter. To avoid computing the eigendecomposition of the normalized graph Laplacian, ChebNet [8] approximates the filter by the Chebyshev polynomials. The widely used Graph Convolutional Network (GCN) [6] simplifies ChebNet by introducing a first-order approximation, and its matrix form can be written as

$$\mathbf{H}^{(k)} = \sigma(\hat{\mathbf{A}}\mathbf{H}^{(k-1)}\mathbf{\Theta}^{(k)}), \tag{2}$$

where $\mathbf{H}^{(k)} \in \mathbb{R}^{N\times d}$ and $\mathbf{\Theta}^{(k)} \in \mathbb{R}^{d\times d}$ are the hidden embedding matrix and the filter parameter matrix in the k-th layer, $\mathbf{H}^{(0)} = \mathbf{X}$ is the node feature matrix, and $\sigma(\cdot)$ is the activation function. $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ is the renormalized adjacency matrix of the graph, where $\tilde{\mathbf{A}} = \mathbf{A}+\mathbf{I}$. By stacking multiple such layers, GCN could aggregate the information from multi-hop neighbors into the center node.

3 Methodology

This section introduces the proposed method. We present the framework, the construction of item graphs, and the process of graph convolution for embedding smoothing.

3.1 Framework

The framework of our proposed method is illustrated in Figure 2. First, we construct a hybrid item graph by fusing the sequential item relations from user interaction sequences with the semantic item relations from item attributes. The

hybrid item graph involves both sequential and semantic relations among items. Second, we perform graph convolutions on the hybrid item graph to generate smoothed item embedding as the input of sequential recommendation models for improving their performance.

Our proposed method can be considered as a plug-in module for sequential recommendation models, namely *Graph-based Embedding Smoothing* (GES). In this paper, we adopt SASRec as the backbone model and apply the proposed GES to it as GES-SASRec to verify its effectiveness. We will describe the main parts of our proposed method in the following subsections.

3.2 Generating Item Graphs

3.2.1 Sequential Item Graph

Compared with graph-based recommendation models, neural models are not necessarily capturing the multi-hop transitivity of items [9]. That is, they are poor at maintaining the global structure of the user-item interaction graph when generating recommendations. For instance, a user's three-hop items in the user-item interaction graph should be ranked higher than his/her five-hop items, while such an order may not be kept for neural models. Hence, we consider encoding the graph structure explicitly in the item embedding to cope with this problem.

Specifically, we aggregate all user interaction sequences to build a global sequential item graph, which reflects relations among items in terms of user behavior. Let $\mathbf{A}_{seq} \in \mathbb{R}^{|I|\times|I|}$ be the adjacency matrix of the sequential item graph. It contains the frequency that two items are adjacent in all user interaction sequences. An example of generating $\mathbf{A}_{seq}$ is shown in Figure 2.

Note that the sequential item graph here is built as an undirected weighted graph. Since we aim at smoothing item embedding according to the item graph, we mainly focus on the adjacency of the items when constructing the item graph, and the sequentiality in user behavior has been modeled by the sequential architecture in SASRec.

3.2.2 Semantic Item Graph

The sequential item graph built from user-item interactions reflects item transition patterns in terms of user behavior. However, it is worth mentioning that semantic item relations are also crucial for measuring item similarities in recommendation, which are not necessarily captured by sequential models. For example, two movies of the same genre but with different release time may rarely appear adjacent to each other in user interaction sequences. Therefore, it is important to consider semantic item relations in sequential recommendation models to complement and regularize the item similarities obtained from user interactions.

Semantic item relations refer to connections between items in terms of attributes or features, which can be obtained from item side information. For instance, books of the same genre or POIs within a small area can be considered semantically related. Then, a semantic item graph can be constructed according to the semantic item relations. Let $\mathbf{A}_{sem} \in \mathbb{R}^{|I|\times|I|}$ be the adjacency matrix of the semantic item graph. An example of generating $\mathbf{A}_{sem}$ is shown in Figure 2. More precisely, we define a_{ij} in $\mathbf{A}_{sem}$ as

$$a_{ij} = \begin{cases} 2 & \text{if } i \text{ has a bidirectional relation with } j, \\ 1 & \text{if } i \text{ has a unidirectional relation with } j, \\ 0 & \text{otherwise.} \end{cases}$$

Although semantic item relations may be unidirectional in real-world scenarios, we construct the semantic item graph as an undirected weighted graph, and the weights reflect the relation strength.

3.2.3 Graph Fusion

To combine both sequential and semantic item graphs for embedding smoothing, a simple idea is to propagate item embedding in each graph and then fuse the output to obtain the final item embedding. However, such an idea couples the two kinds of graphs loosely and neglects item transitions between the two relations. For example, if a user bought item B after item A, and item B shares similar features with item C, then item A and C may be highly related, while such relations are not modeled if we perform embedding smoothing on the item graphs respectively. To this end, we fuse the item graphs before item embedding.

We apply a weighted sum to fusing the sequential and semantic item graphs with self-loops into a hybrid item graph, as

$$\mathbf{A}_{hyb} = \mathbf{I} + \alpha\mathbf{A}_{seq} + \beta\mathbf{A}_{sem}, \tag{3}$$

where α and β are weight coefficients for the sequential item graph and the semantic item graph, respectively. Intuitively, the weights can be regarded as unnormalized probabilities of transition from an item to its sequential/semantic neighbors.

3.3 Graph Convolution for Embedding Smoothing

We perform graph convolutions on the hybrid item graph to obtain smoothed item embedding. Specifically, using GCN [6] for embedding smoothing could be formulated as

$$\mathbf{V}^{(k)} = \sigma(\hat{\mathbf{A}}\mathbf{V}^{(k-1)}\mathbf{W}^{(k)}), \tag{4}$$

where $\mathbf{V}^{(k)} \in \mathbb{R}^{|I|\times d}$ and $\mathbf{W}^{(k)} \in \mathbb{R}^{d\times d}$ are the hidden representations of items and the trainable parameter matrix in the k-th layer, and $\hat{\mathbf{A}}$ is the adjacency matrix of the item graph after symmetric normalization.

Although GCN obtains superior performances on graph-based semi-supervised classification problems, it may be unsuitable for embedding smoothing in recommendation problems [10]. In semi-supervised node classification, nodes have rich features as the input of GCN, e.g., words in articles, such that the multi-layer nonlinear transformation is helpful to improve the model expressiveness. However, items are represented by trainable ID embedding in recommendation models. Hence, performing multi-layer nonlinear transformations may affect the learning of ID embedding, leading to poor recommendation performances.

Inspired by [11], we consider removing the nonlinear transformation in GCN layers and perform a simple graph convolution to smooth the embedding in sequential recommendation models as

$$\mathbf{V}^{(k)} = \hat{\mathbf{A}}\mathbf{V}^{(k-1)}, \tag{5}$$

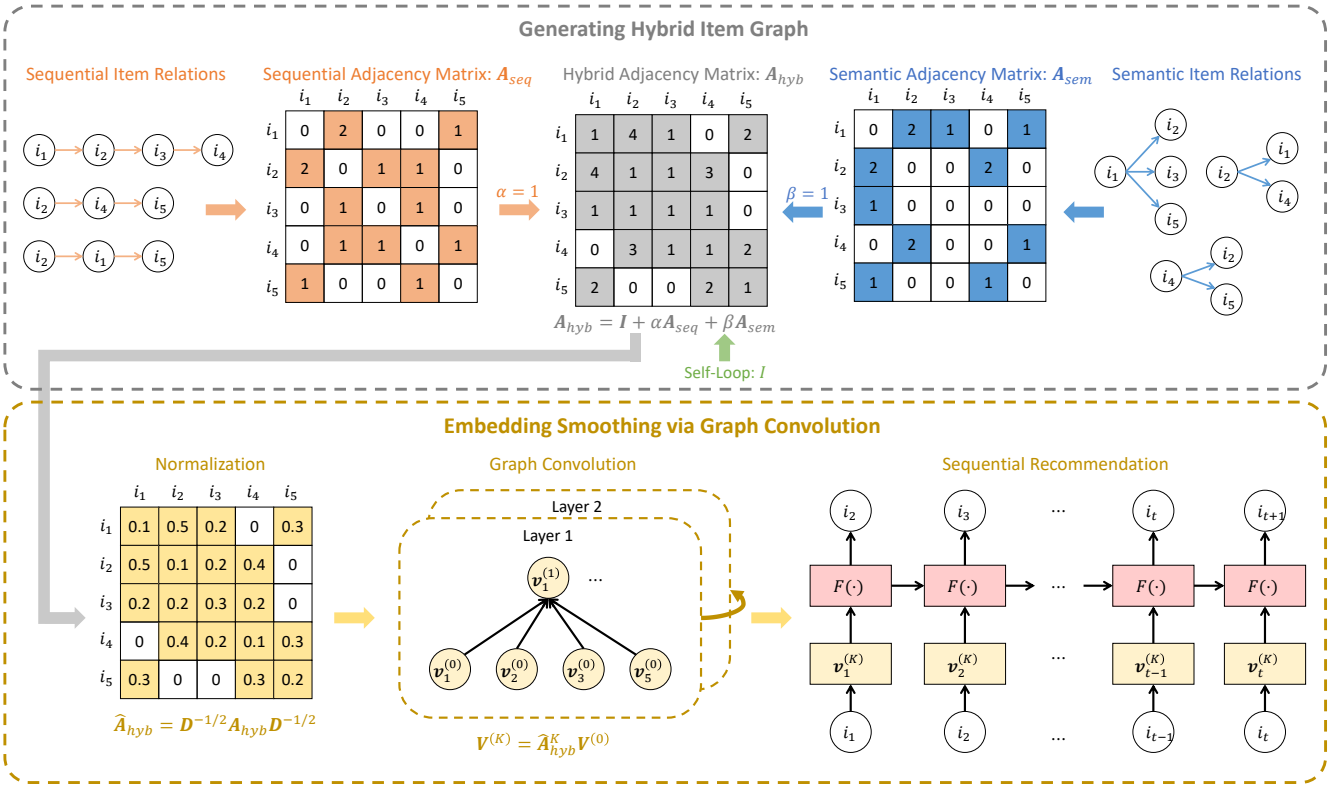


Fig. 2: The framework of Graph-based Embedding Smoothing (GES).

where $\mathbf{V}^{(0)}$ is the trainable ID embedding for items. Note that different from the simple graph convolution in [11] which has a single weight matrix, we do not apply any transformation to item embedding. Thus, the graph convolution acts as a weighted mean filter to denoise the item representations according to the graph structure. The hidden representation matrix in the last layer $\mathbf{V}^{(K)}$ is used as the input of sequential models.

It is worth mentioning that we could also adopt layer aggregation strategies to alleviate the over-smoothing problem of deep graph convolutional networks [12], e.g., sum-pooling or concatenation. We will compare their performances in the experiments.

3.4 Theoretical Analysis

We can draw some insights into how graph convolutions lead to embedding smoothing. Here we take the one-layer simple graph convolution as an instance. The weights in the adjacency matrix are set to 1 for an easy analysis. For item i, its representation after graph convolution is

$$\mathbf{v}_i^{(1)} = \sum_{m \in \mathcal{N}_i^+} \frac{1}{\sqrt{|\mathcal{N}_i^+|}\sqrt{|\mathcal{N}_m^+|}} \mathbf{v}_m^{(0)}, \tag{6}$$

where $\mathcal{N}_i^+ = \mathcal{N}_i \cup \{I_i\}$. Then, the similarity score between item i and item j is calculated by dot-product (consistent with the predictive model in recommendation) as

$$\begin{aligned}
\mathbf{v}_i^{(1)^T} \mathbf{v}_j^{(1)} &= w_{ij} \sum_{m \in \mathcal{N}_i^+} \sum_{n \in \mathcal{N}_j^+} \frac{1}{\sqrt{|\mathcal{N}_m^+|}\sqrt{|\mathcal{N}_n^+|}} \mathbf{v}_m^{(0)^T} \mathbf{v}_n^{(0)} \\
&= w_{ij} \Big(\sum_{m \in \mathcal{N}_i} \sum_{n \in \mathcal{N}_j} \frac{1}{\sqrt{|\mathcal{N}_m^+|}\sqrt{|\mathcal{N}_n^+|}} \mathbf{v}_m^{(0)^T} \mathbf{v}_n^{(0)} \\
&+ \frac{1}{\sqrt{|\mathcal{N}_i^+|}\sqrt{|\mathcal{N}_j^+|}} \mathbf{v}_i^{(0)^T} \mathbf{v}_j^{(0)} \\
&+ \sum_{m \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_m^+|}\sqrt{|\mathcal{N}_j^+|}} \mathbf{v}_m^{(0)^T} \mathbf{v}_j^{(0)} \\
&+ \sum_{n \in \mathcal{N}_j} \frac{1}{\sqrt{|\mathcal{N}_n^+|}\sqrt{|\mathcal{N}_i^+|}} \mathbf{v}_n^{(0)^T} \mathbf{v}_i^{(0)} \Big).
\end{aligned} \tag{7}$$

where $w_{ij} = \frac{1}{\sqrt{|\mathcal{N}_i^+|}\sqrt{|\mathcal{N}_j^+|}}$. The similarity between two items can be divided into four parts: the first part is the neighbor-to-neighbor similarity, which has $|\mathcal{N}_i||\mathcal{N}_j|$ terms; the second part is the node-to-node similarity; and the third and fourth parts are the node-to-neighbor similarity with $|\mathcal{N}_i| + |\mathcal{N}_j|$ terms. Therefore, the similarity between two items after the graph convolution mainly depends on the neighbor-to-neighbor similarity. That is to say, the more similar their neighborhood structure is, the higher their similarity of representation vectors would be. To conclude, embedding smoothing through graph convolutions is mainly based on the second-order proximity [13].

In addition, from the perspective of graph signal processing, [11] proves that the simple graph convolution with self-loops shrinks the graph spectral domain, resulting in

a low-pass filter that captures low-frequency signals, corresponding to smooth features across the graph. As a result, nearby nodes tend to share similar representations after the simple graph convolution.

Applying graph convolution-based embedding smoothing to sequential recommendation models can be considered as predicting the next item i_{t+1} with both the past items $\{i_t, i_{t-1}, \cdots\}$ and the sequentially or semantically related items $\{i_c : c \in \mathcal{N}_{i_t} \cup \mathcal{N}_{i_{t-1}} \cup \cdots\}$. It relaxes the strict sequential dependency in sequential models and enhances their generalization ability.

4 Discussions

In this section, we discuss the connections between our proposed method and related efforts. We then analyze the time complexity of our method.

4.1 Relationship with Graph Laplacian Regularization

Graph Laplacian Regularization (GLR) [14] is widely used for graph-based semi-supervised node classification problems

$$L = \sum_i l(y_i, f(\mathbf{x}_i)) + \lambda \sum_{(i,j)\in\mathcal{E}} a_{ij}||f(\mathbf{x}_i) - f(\mathbf{x}_j)||^2, \quad (8)$$

where the first term is the supervised loss, and the second term is the graph Laplacian regularization that performs label propagation across the graph, assuming that connected nodes share similar labels. λ denotes the weight coefficient.

Semi-supervised embedding [15] extends the regularization term from the prediction layer to the embedding layer. Inspired by Laplacian eigenmaps [16], [17] proposes Graph Regularized Matrix Factorization (GRMF). The formulation of GRMF with item graph regularization can be written as

$$\begin{aligned} L &= \sum_{(i,j)} l(y_{ij}, \mathbf{u}_i^T \mathbf{v}_j) + \lambda \sum_{(m,n)\in\mathcal{E}} a_{mn}||\mathbf{v}_m - \mathbf{v}_n||^2 \\ &= \sum_{(i,j)} l(y_{ij}, \mathbf{u}_i^T \mathbf{v}_j) + \lambda \mathrm{tr}(\mathbf{V}^T \mathbf{L} \mathbf{V}), \end{aligned} \quad (9)$$

where $\mathbf{u}_i$ and $\mathbf{v}_j$ are representation vectors for user i and item j respectively, $\mathrm{tr}(\cdot)$ is the trace of a matrix, and $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the unnormalized Laplacian matrix. GRMF can be regarded as embedding propagation on the graph, assuming that nearby items tend to share similar representations.

Performing graph Laplacian regularization can also smooth the item embedding according to the item relations, but it has the following disadvantages compared with graph convolutions:

First, graph Laplacian regularization is built on the first-order proximity, i.e., whether there is a relation between two items. However, observed item relations in real-world datasets are often sparse and many similar items may not be explicitly linked [13]. Therefore, considering the similarity of neighborhood structure (i.e., the second-order proximity) can highly enrich the relations of items and is more efficient for embedding smoothing.

Second, graph Laplacian regularization is based on the semi-supervised learning framework, and its performance could be highly sensitive to the weight λ. Too small or too large regularization would result in under-smoothing or over-smoothing, hurting recommendation performance. While the graph convolution uses the normalized propagation matrix and its performance is less sensitive to the weight that balances the information from the node itself and from its neighbors.

4.2 Relationship with SR-GNN

SR-GNN [18] uses the Gated Graph Neural Network (GGNN) [19] to model the local dependency in session-based recommendation, which is highly related to our work. The session sequences are gathered to construct a global graph, and each session sequence is modeled as a subgraph that extracted from the global graph. Then GGNN is applied to these subgraphs, which can be formulated as [20]

$$\mathbf{a}_t^{in} = \mathbf{A}_t^{in}([\mathbf{v}_1, \cdots, \mathbf{v}_n]\mathbf{W}^{in} + \mathbf{b}^{in}), \quad (10)$$

$$\mathbf{a}_t^{out} = \mathbf{A}_t^{out}([\mathbf{v}_1, \cdots, \mathbf{v}_n]\mathbf{W}^{out} + \mathbf{b}^{out}), \quad (11)$$

$$\mathbf{a}_t = [\mathbf{a}_t^{in}; \mathbf{a}_t^{out}], \quad (12)$$

$$\mathbf{h}_t = \mathrm{GRU}(\mathbf{a}_t, \mathbf{v}_{t-1}), \quad (13)$$

where $\mathbf{A}^{in}$ and $\mathbf{A}^{out}$ are the adjacency matrices that represent incoming and outgoing edges in the subgraph of a session, and $\mathbf{W}$ and $\mathbf{b}$ are the trainable transformation matrix and the bias vector.

It is worth mentioning that our proposed method differs from SR-GNN in two aspects:

First, SR-GNN only considers how items within a session communicate with each other according to the global graph. That is, related items not interacted in the session are excluded. On the contrary, our proposed method performs embedding smoothing on the global graph. For a user's interaction sequence, our method also considers the related items that the user has not interacted with but other users have.

Second, our proposed method is a general framework. It can take advantage of different item graphs to improve recommendation performance, such as the sequential item graph from user-item interactions, the semantic item graph from item attributes, or the hybrid item graph. More importantly, this framework can be applied to a variety of sequential models for improving their performances, such as models based on Markov Chain, RNN, CNN, or self-attention.

4.3 Time Complexity Analysis

Since the proposed embedding smoothing method is a plug-in module for sequential recommendation models, here we only analyze the time complexity of the graph convolution in it. By using sparse matrices, the original form of GCN takes $\mathcal{O}(|\mathcal{E}|dK + |\mathcal{I}|d^2K)$ [21], where $|\mathcal{E}|$ is the number of edges in the item graph, $|\mathcal{I}|$ is the number of items, d is the embedding size, and K denotes the depth of GCN. By removing the unnecessary nonlinear transformations, embedding smoothing with the simple graph convolution takes $\mathcal{O}(|\mathcal{E}|dK)$. In addition, if the layer aggregation strategies are not used , the propagation matrix $\hat{\mathbf{A}}^K$ can be precomputed [11], and the time complexity further reduces to $\mathcal{O}(|\mathcal{E}|d)$. We consider that the additional time cost is acceptable for significant improvements on model performance. We can also adopt neighborhood sampling strategies to further reduce the time cost of the graph convolution.

TABLE 1: Statistics of the evaluation datasets.

Statistics	Amazon Books	Yelp	Google Local
#Users	27,738	32,206	16,381
#Items	43,790	27,671	34,928
#Interactions	624,573	726,154	427,910
Sparsity	0.051%	0.081%	0.075%
Avg. Length	20.52	20.55	24.12
#Relations	847,258	268,402	687,397

5 EXPERIMENTS

In this section, data experiments were conducted to verify the effectiveness of the proposed method. The codes have been published on GitHub[1]. The experiments were designed to answer the following research questions:

- **RQ1.** Can the proposed embedding smoothing method improve the performance of SASRec?
- **RQ2.** Can the proposed GES-SASRec outperform the state-of-the-art recommendation methods?
- **RQ3.** How do the hyperparameters and setups affect the performance of GES-SASRec?
- **RQ4.** How do SASRec and GES-SASRec perform under the shuffled user interaction orders?
- **RQ5.** Can the proposed embedding smoothing method apply to other categories of sequential recommendation models?

5.1 Datasets

We evaluated the proposed method on three real-world datasets with different domains and sparsity. The statistics of the datasets are summarized in Table 1.

Amazon Books.[2] This is a product-review dataset crawled from *Amazon.com*, a well-known online shopping platform [22]. The entire data is split into several subsets according to the top-level categories. Here the largest category, i.e., the 'Books' category, was adopted. We used the data within 2013 and retained the items/users with at least 20/10 records. The 'also-view', 'also-buy', 'buy after viewing' and 'buy together' relations were treated as semantic item relations.

Yelp.[3] This dataset is a subset of the businesses, reviews, and user data in *Yelp.com*, a large location-based social platform. Here we used the review and business data within 2014-2016 in Yelp Challenge 2019 and retained the items/users with at least 20/10 records. The nearest at most 20 businesses in the same city were regarded as semantically related items for each business.

Google Local.[4] This is a POI-based dataset crawled from *Google Local*, which contains user reviews and POI data [23]. The datasets were preprocessed by retaining the items/users with at least 20/10 records. The nearest (truncated by a distance threshold) at most 20 POIs were treated as semantically related items for each POI.

1. http://github.com/zhuty16/GES
2. http://jmcauley.ucsd.edu/data/amazon/index_2014.html
3. https://www.yelp.com/dataset/
4. https://cseweb.ucsd.edu/jmcauley/datasets.html

5.2 Experimental Settings

Evaluation Protocols. We adopted the leave-one-out scheme to evaluate the performances of the models on sequential recommendation, which has been widely used in literature [4], [24]. Given a user's interaction sequence $S^{(u)} = (i_1^{(u)}, i_2^{(u)}, \cdots, i_{n_u}^{(u)})$, the last item $i_{n_u}^{(u)}$ and the next-to-last item $i_{n_u-1}^{(u)}$ were used for test and validation respectively, and the remaining items $(i_1^{(u)}, i_2^{(u)}, \cdots, i_{n_u-2}^{(u)})$ were for training. For time-saving evaluation, we followed a strategy that 999 items which have not been interacted by a user were randomly sampled to be ranked along with the ground-truth item by the models. The performance of a ranking list was judged by Hit Ratio (HR), Normalized Discounted Cumulative Gain (NDCG), and Mean Reciprocal Rank (MRR)

$$\mathrm{HR@}N = \frac{1}{|U|}\sum_{u\in U}\mathbf{1}(r_{u,g_u} \le N), \tag{14}$$

$$\mathrm{NDCG@}N = \frac{1}{|U|}\sum_{u\in U}\frac{\mathbf{1}(r_{u,g_u} \le N)}{\log_2(r_{u,g_u}+1)}, \tag{15}$$

$$\mathrm{MRR@}N = \frac{1}{|U|}\sum_{u\in U}\frac{\mathbf{1}(r_{u,g_u} \le N)}{r_{u,g_u}}, \tag{16}$$

where g_u is the ground-truth item for user u, r_{u,g_u} is the rank generated by a recommendation model for user u and item g_u, and $\mathbf{1}(\cdot)$ is an indicator function. Without special mention, the ranking list was truncated at 10 for all metrics. As such, HR@10 intuitively measures whether the ground-truth item is presented on the top-10 list, while NDCG@10 and MRR@10 account for the position of the hit by assigning higher scores to hits at top ranks. The average scores of all users for three metrics were reported. Due to space limitation, we only show the performances of NDCG@10 in some figures.

Baselines. To verify the effectiveness of GES-SASRec, we compare its performance with several groups of baselines. The first group contains general recommendation methods, which only use the user-item interaction data for item recommendation:

- **Most Popular.** It is a non-personalized method that simply ranks items according to their popularity, i.e., the number of interactions by all users.
- **BPR** [25]. It is an MF-based model with a pairwise ranking loss to learn from implicit feedback.
- **Mult-DAE** [26]. It is a state-of-the-art item-based recommendation model built upon denoising autoencoders with multinomial conditional likelihood.
- **LightGCN** [10]. It is a state-of-the-art graph convolution-based recommendation model that learns user/item embedding by linear propagation on the user-item bipartite graph.

The second group includes sequential recommendation models that capture user dynamic preferences:

- **FPMC** [27]. It combines matrix factorization and factorized first-order Markov chains for sequential recommendation.

- **TransRec** [23]. It models each user as a translation vector between items to capture the transition patterns for sequential recommendation.
- **GRU4Rec** [2]. It adopts GRU to model the session sequences for session-based recommendation.
- **NARM** [28]. It uses GRU with attention mechanism to capture both sequential behavior and main purpose of users for session-based recommendation.
- **Caser** [3]. It adopts horizontal and vertical convolutions to learn sequential patterns for recommendation.
- **SASRec** [4]. It is a state-of-the-art sequential recommendation model based on the self-attention module in Transformer [1].

The third group contains relation-aware recommendation models that use semantic item relations to improve recommendation performance:

- **CKE** [29]. It is an embedding-based model that combines structural, textual, and visual information in an MF-based framework. Here only the structural information was used.
- **MCF** [30]. It is a relation-aware recommendation model that simultaneously factorizes the user-item rating matrix and the item-item relation matrix for rating prediction.
- **LightGCN+**. It performs LightGCN [10] on the heterogeneous graph that consists of user-item interactions and item-item semantic relations to generate user and item embedding for recommendation.

The fourth group includes a recommendation model that simultaneously encodes the sequential and semantic item relations for recommendation:

- **MoHR** [24]. It adopts the translational operators to model the mixture of heterogeneous item-item relations under a multi-task learning framework.

Hyperparameter Settings. We implemented the compared methods with TensorFlow. For common hyperparameters in all learning-based methods, the embedding size d was set to 64 with the random normal initializer (with a mean of 0 and standard deviation of 0.01), the l_2 regularization was tuned in $\{0, 1e\text{-}8, 1e\text{-}7, 1e\text{-}6, 1e\text{-}5, 1e\text{-}4, 1e\text{-}3\}$, and the learning rate was tuned in $\{1e\text{-}2, 5e\text{-}3, 2e\text{-}3, 1e\text{-}3, 5e\text{-}4, 2e\text{-}4, 1e\text{-}4\}$. All the learning-based methods were trained with mini-batch Adam [31], in the batch size of 256. The results of baseline methods under their optimal hyperparameter settings were reported. For GRU4Rec, NARM, SASRec, and GES-SASRec, we set the maximum sequence length to 50. For GES-SASRec, the number of graph convolutional layers is tuned in $\{1, 2, 3\}$.

5.3 Performance Comparison with SASRec (RQ1)

We report the performances of SASRec and GES-SASRec with 1-3 graph convolutional layers on the sequential/semantic/hybrid item graph in Table 2. We further plot their learning curves (test performance of NDCG@10) in Figure 3 to show the benefits of the proposed embedding smoothing method. Note that the optimal learning rate may be different for SASRec and GES-SASRec with different item graphs. For comparison, we use 0.001 in Figure 3. We conclude the findings as follows:

First, the effectiveness of embedding smoothing with sequential item graphs may rely on the strength of sequentiality in user interactions. For the high sequentiality dataset (Amazon Books), embedding smoothing with sequential item graphs may cause over-smoothing and harm the performance of SASRec. While for the medium and low sequentiality datasets (Yelp and Google Local), it could benefit recommendation performance.

Second, the effectiveness of embedding smoothing with semantic item graphs may depend on the reliability of semantic item relations and the sparsity of the dataset. As can be seen, embedding smoothing with semantic item graphs is more effective for Amazon Books and Yelp.

Third, using hybrid item graphs for embedding smoothing could further improve recommendation performance and achieves the best results on all datasets. This may be because the hybrid item graph is a balanced combination of the information from user behavior and from item attributes.

Fourth, the performances of GES-SASRec do not necessarily improve with more graph convolutional layers. This may be because too many graph convolutional layers would lead to over-smoothing for item embedding, hurting the recommendation performance. Generally speaking, a 1-2 graph convolutional layer may be more suitable.

5.4 Performance Comparison with State-of-the-Arts (RQ2)

Table 3 shows the performance comparison with baseline methods. We have the following main observations:

First, the strength of sequentiality in user interactions may affect the performances of the methods from different categories. For the high sequentiality dataset (Amazon Books), the best sequential recommendation model SASRec outperforms the best general recommendation model Mult-DAE significantly. While for the low sequentiality dataset (Yelp), SASRec achieves much worse performance than Mult-DAE. For the medium sequentiality dataset (Google Local), the performances of SASRec and Mult-DAE are comparable.

Second, semantic item relations are significantly beneficial to the performances of recommendation methods. For MF-based methods, MCF outperforms BPR significantly. For graph convolution-based methods, LightGCN+ obtains better results than LightGCN.

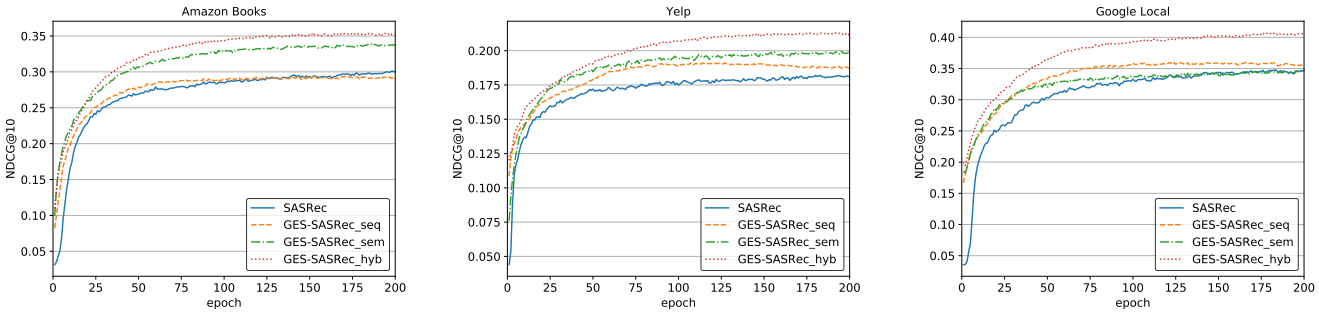
Third, GES-SASRec outperforms other baselines on Amazon Books and Google Local but does not perform best on Yelp. That is, GES-SASRec shows stronger recommendation performance on high sequentiality datasets.

5.5 Hyperparameter and Ablation Analyses (RQ3)

We perform detailed analyses on GES-SASRec to show how key hyperparameters and layer aggregation strategies affect its performance. We also investigate the performance of SASRec with different embedding smoothing approaches.

TABLE 2: Performance of GES-SASRec with different item graphs and different numbers of graph convolutional layers. Best results are in boldface.

#Layers	Item Graph	Amazon Books			Yelp			Google Local		
		HR@10	NDCG@10	MRR@10	HR@10	NDCG@10	MRR@10	HR@10	NDCG@10	MRR@10
0 Layer	-	0.4838	0.3075	0.2535	0.3583	0.1946	0.1451	0.5680	0.3464	0.2788
1 Layer	sequential	0.4596	0.2948	0.2445	0.3422	0.1897	0.1436	0.5679	0.3626	0.2994
	semantic	0.5295	0.3364	0.2766	0.3620	0.1978	0.1478	0.5714	0.3433	0.2731
	hybrid	0.5314	0.3446	0.2877	0.3773	0.2105	0.1600	**0.6257**	**0.4031**	**0.3358**
2 Layers	sequential	0.4597	0.2923	0.2428	0.3531	0.1948	0.1468	0.5653	0.3578	0.2951
	semantic	0.5321	0.3345	0.2744	0.3208	0.1715	0.1266	0.5460	0.3220	0.2532
	hybrid	**0.5405**	**0.3553**	**0.2991**	**0.3825**	**0.2128**	**0.1620**	0.6194	0.3910	0.3206
3 Layers	sequential	0.4529	0.2832	0.2314	0.3441	0.1883	0.1411	0.5572	0.3438	0.2780
	semantic	0.5105	0.3189	0.2604	0.3041	0.1611	0.1178	0.5409	0.3176	0.2495
	hybrid	0.5359	0.3469	0.2886	0.3765	0.2061	0.1548	0.6016	0.3713	0.3001



(a) Amazon Books (b) Yelp (c) Google Local

Fig. 3: Test performances of SASRec and GES-SASRec w.r.t. the training epochs.

TABLE 3: Performance comparison with embedding size 64. Best results are in boldface.

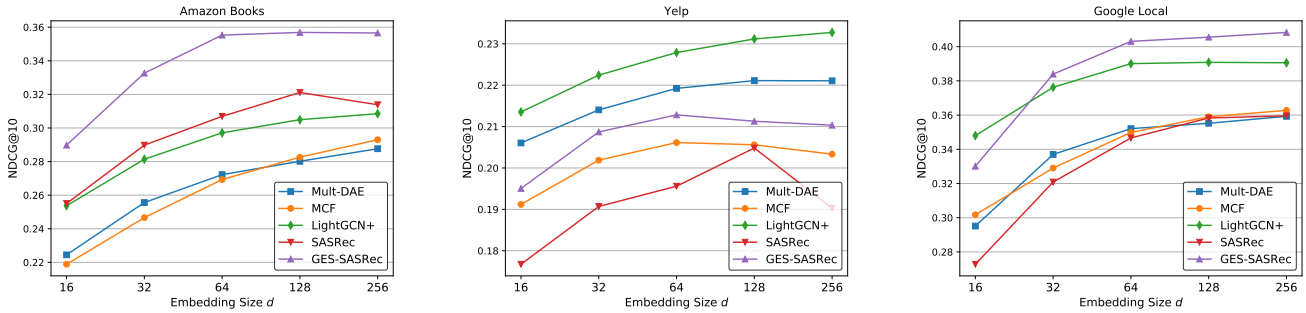
Method	Information	Amazon Books			Yelp			Google Local		
		HR@10	NDCG@10	MRR@10	HR@10	NDCG@10	MRR@10	HR@10	NDCG@10	MRR@10
MP	-	0.0578	0.0270	0.0179	0.0868	0.0433	0.0303	0.0673	0.0326	0.0223
BPR	-	0.3877	0.2200	0.1690	0.3484	0.1905	0.1429	0.5224	0.3093	0.2438
Mult-DAE	-	0.4450	0.2722	0.2193	0.3907	0.2192	0.1670	0.5676	0.3521	0.2859
LightGCN	-	0.4461	0.2612	0.2049	0.3889	0.2167	0.1644	0.5614	0.3499	0.2860
FPMC	sequential	0.4412	0.2732	0.2218	0.3338	0.1819	0.1362	0.5654	0.3567	0.2932
TransRec	sequential	0.4332	0.2550	0.2006	0.3666	0.2027	0.1528	0.5534	0.3313	0.2647
GRU4Rec	sequential	0.4262	0.2508	0.1973	0.3225	0.1704	0.1250	0.5288	0.3131	0.2475
NARM	sequential	0.3945	0.2354	0.1873	0.3114	0.1626	0.1176	0.5175	0.3164	0.2554
Caser	sequential	0.4054	0.2499	0.2022	0.3065	0.1639	0.1225	0.5313	0.3347	0.2740
SASRec	sequential	0.4824	0.3069	0.2528	0.3588	0.1956	0.1461	0.5664	0.3466	0.2792
MCF	semantic	0.4510	0.2693	0.2145	0.3760	0.2061	0.1552	0.5722	0.3498	0.2811
CKE	semantic	0.4415	0.2530	0.1954	0.3726	0.2055	0.1548	0.5735	0.3464	0.2765
LightGCN+	semantic	0.4885	0.2971	0.2397	**0.4074**	**0.2279**	**0.1732**	0.6172	0.3901	0.3196
MoHR	hybrid	0.4946	0.2847	0.2212	0.3974	0.2215	0.1681	0.6032	0.3591	0.2845
GES-SASRec	hybrid	**0.5405**	**0.3553**	**0.2991**	0.3825	0.2128	0.1620	**0.6257**	**0.4031**	**0.3358**

5.5.1 Effect of Embedding Size

Figure 4 shows the performances of the compared methods w.r.t. the embedding size d. From a macro perspective, with the increase of embedding size, the performances of all models improve due to stronger expressiveness. However, the marginal improvement is decreasing, and overfitting may occur with large embedding sizes for some models. Specifically, GES-SASRec achieves better results than SASRec under different embedding sizes on all datasets. In addition, GES-SASRec significantly outperforms the baseline methods on higher sequentiality datasets (Amazon Books and Google Local), especially when the embedding size is large.

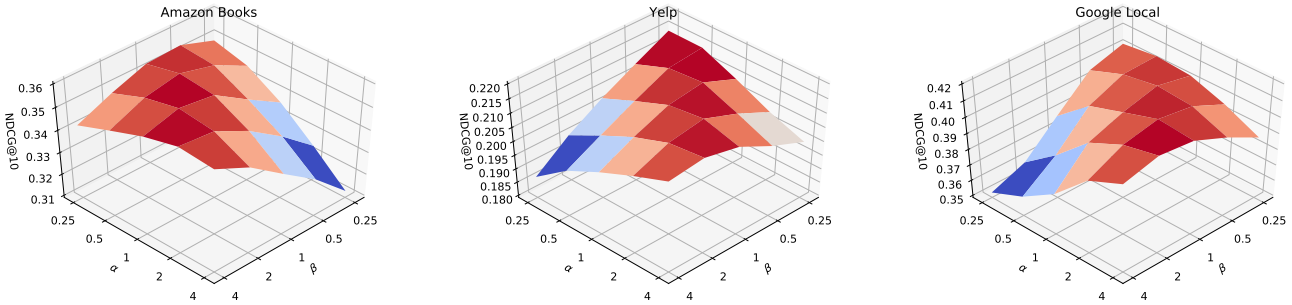
5.5.2 Effect of Relation Coefficients

Figure 5 shows the results of GES-SASRec w.r.t. the sequential/semantic relation coefficient α/β. Generally, both item relations are significant for the performance of GES-SASRec. When α and β are close to each other, the model performs well. Note that the optimal ratio of α to β depends on datasets. For Amazon Books, the optimal ratio of α to β is about 0.5, that is, the optimal weight of semantic relations is twice that of sequential relations. While for Yelp and Google Local, the optimal ratio is about 2, that is, the optimal weight of sequential relations is twice that of semantic relations.



(a) Amazon Books (b) Yelp (c) Google Local

Fig. 4: Performances of the models w.r.t. the embedding size d.



(a) Amazon Books (b) Yelp (c) Google Local

Fig. 5: Performance of GES-SASRec w.r.t. the item relation coefficients α and β.

5.5.3 *Effect of Layer Aggregation*

Figure 6 shows the performances of GES-SASRec with different layer-aggregation functions. As can be seen, the performance of GES-SASRec without layer aggregations drops when the number of layers increases from 1-2 to 3, indicating that the over-smoothing problem occurs. While using layer aggregations could effectively avoid this problem, as the model performance slightly improves with the increase of layers in most cases. Moreover, GES-SASRec using 1 or 2 layers without layer aggregations achieves better performances than using layer aggregations with 3 layers in most cases. That is, using 1-2 graph convolutional layers without layer aggregations may achieve more suitable smoothness for item embedding in recommendation.

5.5.4 *Effect of Embedding Smoothing Approach*

Table 4 shows the performances of SASRec with different embedding smoothing methods on the hybrid item graphs, including Graph Laplacian Regularization [14] (GLR-SASRec), Graph Convolutional Network [6] (GCN-SASRec), and Simple Graph Convolution [11] (GES-SASRec). We can see that all embedding smoothing methods could improve the performances of SASRec in most cases. Compared with graph Laplacian regularization, graph convolutions seem more effective. Additionally, GES-SASRec outperforms GCN-SASRec in most cases, showing that the redundant complexity of transformations in graph convolutional layers may harm recommendation performance.

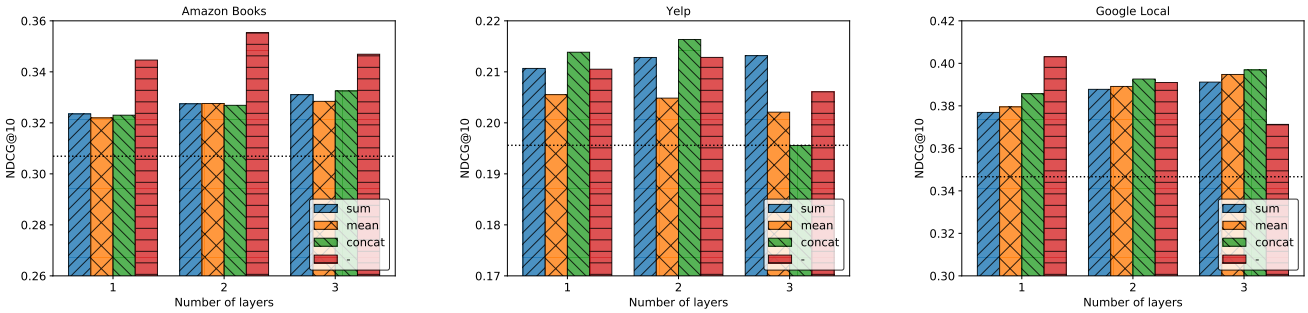
5.6 Performance under shuffled User Interaction Orders (RQ4)

It is known that the performance of sequential recommendation models will strongly rely on the order of user interactions. Here experiments are conducted to show how SASRec performs when the order of user interactions is disrupted and whether the proposed embedding smoothing method could alleviate the performance degradation. In Figure 7, we show the performances of SASRec, GES-SASRec, LightGCN, and LightGCN+ under different shuffle ratios in the test process. The shuffle ratio is defined as the proportion of items whose positions are randomly shuffled in user interaction sequences. We have the following findings:

First, the performances of SASRec and GES-SASRec are both sensitive to the order of user interactions, as their recommendation accuracy drops with the increase of the shuffle ratio.

Second, for Amazon Books, SASRec outperforms LightGCN when the shuffle ratio is lower than 70%. GES-SASRec outperforms LightGCN+ even when the shuffle ratio increases to 90%. Thus, embedding smoothing could help SASRec go further ahead of LightGCN on recommendation performance on high sequentiality datasets.

Third, for Yelp, LightGCN performs significantly better than SASRec even when the positions of items are not shuffled, while the performance of GES-SASRec gets closer to LightGCN and LightGCN+ owing to the proposed embedding smoothing method. That is, embedding smoothing could narrow the performance gap between SASRec and



(a) Amazon Books (b) Yelp (c) Google Local

Fig. 6: Performance of GES-SASRec w.r.t. the layer aggregation functions. Dotted line indicates the performance of SASRec.

TABLE 4: Performance of SASRec with different embedding smoothing methods. Best results are in boldface.

Method	Amazon Books			Yelp			Google Local		
	HR@10	NDCG@10	MRR@10	HR@10	NDCG@10	MRR@10	HR@10	NDCG@10	MRR@10
SASRec	0.4824	0.3069	0.2528	0.3588	0.1956	0.1461	0.5664	0.3466	0.2792
GLR-SASRec	0.4901	0.2982	0.2391	0.3708	0.2026	0.1519	0.6080	0.3760	0.3041
GCN-SASRec	0.5138	0.3296	0.2725	**0.3890**	**0.2169**	**0.1645**	0.6085	0.3901	0.3224
GES-SASRec	**0.5405**	**0.3553**	**0.2991**	0.3825	0.2128	0.1620	**0.6257**	**0.4031**	**0.3358**

LightGCN on low sequentiality datasets.

Fourth, for Google Local, SASRec achieves comparable performances with LightGCN when the shuffle ratio equals 0, while GES-SASRec outperforms LightGCN+ when the shuffle ratio is lower than 30%. Hence, embedding smoothing could help SASRec outperform LightGCN on medium sequentiality datasets.

5.7 Embedding Smoothing for Other Models (RQ5)

We also conduct experiments to show whether the proposed embedding smoothing method could apply to other categories of sequential recommendation models. Specifically, we try to apply the proposed method to other four sequential models, including Markov Chain-based model (TransRec), CNN-based model (Caser), RNN-based model (GRU4Rec), and RNN with attention-based model (NARM). We smooth the item embedding in these models with the proposed hybrid item graph and show the results in Table 5. As can be seen, the proposed embedding smoothing method could improve the performances of all these sequential recommendation models significantly, especially for item-based methods (GRU4Rec and NARM).

6 Related Work

In this section, we review recent efforts relating to our work, including sequential recommendation models, GNN-based recommendation models, and regularization in recommendation.

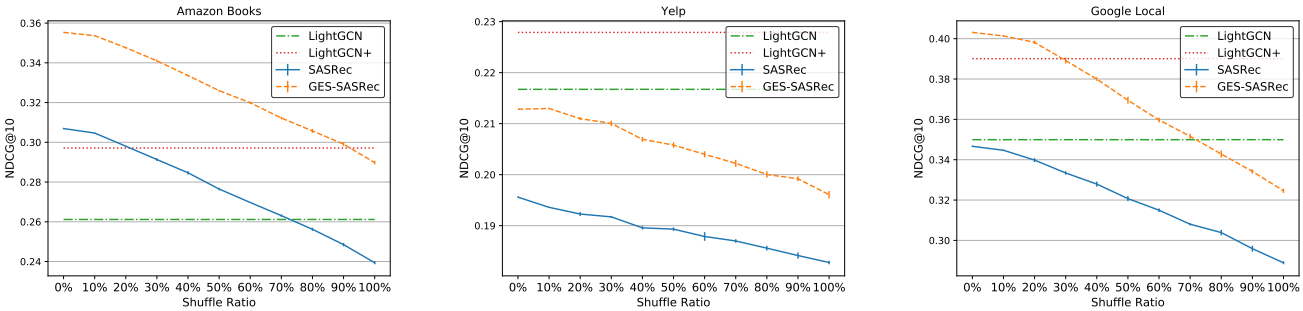
6.1 Sequential Recommendation Models

Efforts on sequential recommendation problems focus on how to capture user dynamic preferences. Early work adopts Markov Chains (MCs) to learn the transition patterns over items. For example, FPMC [27] combines the Matrix Factorization (MF) with the first-order Markov chain to model both long-term and short-term preference for users. Fossil [32] fuses the similarity-based model with high-order Markov chains to predict personalized sequential behavior. TransRec [23] adopts the translational operator to model the transitions of items.

Recently, deep learning-based sequential models (e.g., RNN, CNN, Transformer, etc.) achieve great success in many areas. Thus, some efforts seek to apply these models to sequential recommendation. For example, GRU4Rec [2] exploits Gated Recurrent Unit (GRU) for session-based recommendation. NARM [28] adopts GRU with an attention mechanism to capture the global and local preference of users. Caser [3] considers user recent interactions as an image and uses CNNs with both horizontal and vertical convolutional filters to learn sequential patterns. RUM [33] designs the item-level and feature-level memory-augmented neural networks to store and update user preferences. SHAN [34] uses a two-layer hierarchical attention network to couple user's long-term and short-term preference. SASRec [4] and BERT4Rec [35] adopt the self-attention blocks in Transformer to encode user's sequential behavior with the left-to-right setting and the bidirectional setting, respectively.

6.2 GNN-based Recommendation Models

Graph Neural Networks (GNNs) have been widely used in recent efforts to deal with graph-structure data in recommendation models [36]. Specifically, these graphs fall within four categories: 1) User-item bipartite graph. These models perform embedding propagation on the user-item bipartite graph with different strategies to obtain smoothed user/item embedding for recommendation, such as GCMC [37], SpectralCF [38], NGCF [39], LR-GCCF [40], and LightGCN [10]. 2) User-user social network. These models apply GNNs to the user social network to model the social influence propagating among users for social recommendation,



(a) Amazon Books (b) Yelp (c) Google Local

Fig. 7: Performances of SASRec, GES-SASRec, LightGCN and LightGCN+ w.r.t. the shuffle ratio.

TABLE 5: Performances of sequential recommendation models with and without the proposed embedding smoothing method.

Category	Method	Amazon Books			Yelp			Google Local		
		HR@10	NDCG@10	MRR@10	HR@10	NDCG@10	MRR@10	HR@10	NDCG@10	MRR@10
Markov Chain	**TransRec**	0.4332	0.2550	0.2006	0.3666	0.2027	0.1528	0.5534	0.3313	0.2647
	GES-TransRec	0.4670	0.2697	0.2093	0.3879	0.2139	0.1614	0.6012	0.3624	0.2892
	Improvement	7.81%	5.77%	4.31%	5.81%	5.53%	5.66%	8.64%	9.37%	9.24%
CNN	**Caser**	0.4054	0.2499	0.2022	0.3065	0.1639	0.1225	0.5313	0.3347	0.2740
	GES-Caser	0.5016	0.3138	0.2558	0.3706	0.2065	0.1565	0.6216	0.3967	0.3268
	Improvement	23.74%	25.56%	26.53%	20.92%	25.98%	27.72%	16.99%	18.54%	19.27%
RNN	**GRU4Rec**	0.4262	0.2508	0.1973	0.3225	0.1704	0.1250	0.5288	0.3131	0.2475
	GES-GRU4Rec	0.4893	0.2920	0.2315	0.3576	0.1939	0.1445	0.5933	0.3589	0.2873
	Improvement	14.78%	16.46%	17.31%	10.87%	13.80%	15.56%	12.20%	14.62%	16.09%
RNN+Attention	**NARM**	0.3945	0.2354	0.1873	0.3114	0.1626	0.1176	0.5175	0.3164	0.2554
	GES-NARM	0.5155	0.3182	0.2578	0.3835	0.2136	0.1622	0.6081	0.3800	0.3096
	Improvement	30.67%	35.21%	37.68%	23.18%	31.41%	37.86%	17.52%	20.08%	21.22%

such as DGRec [41], DANSER [42], GraphRec [43], and DiffNet [44]. 3) Item-item graph. These models attempt to leverage item relations from the training data or the side information to regularize item embedding for improving recommendation accuracy, such as SR-GNN [18], GC-SAN [20], and PinSage [45]. 4) Heterogeneous information network/knowledge graph. These models propagate user/item embedding in the heterogeneous information network or the knowledge graph to generate knowledge-aware recommendations, such as RippleNet [46], KGCN-LS [47], and KGAT [48].

6.3 Regularization in Recommendation Models

Recommendation models often suffer from data sparsity problems, which make them hard to capture user preferences and item features and lead to unsatisfactory performances. To cope with the problems, some recent efforts seek to leverage the graph data to regularize these models by making the users/items that are close in the graph have similar representations. The strategies have two schools of thought: 1) Using graphs generated from the user-item interactions. These efforts construct the user co-purchase or the item co-occurrence relations from user-item interactions, and treat these relations as global features for users/items to regularize recommendation models, such as CoFactor [49], RME [50], BiNE [51], and GRMF [17]. 2) Using graphs built from the side information. These methods leverage the side information of users/items to construct user/item graphs for regularizing recommendation models [52], such as the user-user graph in SoReg [53] built from user social networks, the item-item graph in CMF [54] constructed upon item features, and the item-entity graph in MKR [55] derived from the knowledge graph. Most of these methods follow a multi-task learning framework to regularize the main task (recommendation) with the auxiliary task (relation learning).

7 Conclusion

In this study, we have pointed out that the sequentiality in user interactions may vary greatly in different recommendation scenarios, which may seriously affect the performances of sequential recommendation models. Additionally, sequential models only consider sequential item relations in terms of user behavior, while neglecting semantic item relations that are crucial for measuring item similarities in recommendation. To deal with the problems, we have proposed an effective framework to enhance the performances of sequential recommendation models. Our main idea is to smooth the item embedding in sequential models with item graphs, which consists of two essential steps: generating a hybrid item relation graph and performing graph convolutions. To construct a hybrid item relation graph, we fuse the sequential item relations derived from user-item interactions with the semantic item relations built upon item attributes. Then we perform graph convolutions on the hybrid item graph to generate smoothed item embedding as the input of sequential recommendation models. We have evaluated the proposed embedding smoothing

method with a state-of-the-art sequential recommendation model, SASRec, on three public datasets. The experimental results show that the improved SASRec model outperforms all baselines on high/medium sequentiality dataset and achieves better results on low sequentiality dataset. We have also conducted experiments to verify the effectiveness of the proposed embedding smoothing method on different categories of sequential recommendation models.

For future work, one possible effort is to explore heterogeneous semantic relations between items (e.g., movies with the same genre/director/actor), so as to design embedding smoothing strategies with these relations.

ACKNOWLEDGMENTS

The work was partly supported by the National Natural Science Foundation of China (71901011) and the MOE Project of Key Research Institute of Humanities and Social Sciences at Universities (17JJD630006).

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[2] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," *arXiv preprint arXiv:1511.06939*, 2015.

[3] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018, pp. 565–573.

[4] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 197–206.

[5] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Proceedings of the eleventh international conference on data engineering*. IEEE, 1995, pp. 3–14.

[6] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

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

[8] 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.

[9] Y. Zhang, D. Wang, and Y. Zhang, "Neural ir meets graph embedding: A ranking model for product search," in *The World Wide Web Conference*, 2019, pp. 2390–2400.

[10] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," *arXiv preprint arXiv:2002.02126*, 2020.

[11] F. Wu, T. Zhang, A. H. d. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," *arXiv preprint arXiv:1902.07153*, 2019.

[12] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," *arXiv preprint arXiv:1806.03536*, 2018.

[13] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 1067–1077.

[14] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," *arXiv preprint arXiv:1603.08861*, 2016.

[15] G. B. Orr and K.-R. Müller, *Neural networks: tricks of the trade*. Springer, 2003.

[16] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[17] N. Rao, H.-F. Yu, P. K. Ravikumar, and I. S. Dhillon, "Collaborative filtering with graph information: Consistency and scalable methods," in *Advances in neural information processing systems*, 2015, pp. 2107–2115.

[18] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 346–353.

[19] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[20] C. Xu, P. Zhao, Y. Liu, V. S. Sheng, J. Xu, F. Zhuang, J. Fang, and X. Zhou, "Graph contextualized self-attention network for session-based recommendation," in *Proc. 28th Int. Joint Conf. Artif. Intell.(IJCAI)*, 2019, pp. 3940–3946.

[21] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[22] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *proceedings of the 25th international conference on world wide web*, 2016, pp. 507–517.

[23] R. He, W.-C. Kang, and J. McAuley, "Translation-based recommendation," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, 2017, pp. 161–169.

[24] W.-C. Kang, M. Wan, and J. McAuley, "Recommendation through mixtures of heterogeneous item relationships," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 1143–1152.

[25] 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*, 2009, pp. 452–461.

[26] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 689–698.

[27] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 811–820.

[28] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 1419–1428.

[29] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 353–362.

[30] C. Park, D. Kim, J. Oh, and H. Yu, "Do" also-viewed" products help user rating prediction?" in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 1113–1122.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[32] R. He and J. McAuley, "Fusing similarity models with markov chains for sparse sequential recommendation," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 2016, pp. 191–200.

[33] X. Chen, H. Xu, Y. Zhang, J. Tang, Y. Cao, Z. Qin, and H. Zha, "Sequential recommendation with user memory networks," in *Proceedings of the eleventh ACM international conference on web search and data mining*, 2018, pp. 108–116.

[34] H. Ying, F. Zhuang, F. Zhang, Y. Liu, G. Xu, X. Xie, H. Xiong, and J. Wu, "Sequential recommender system based on hierarchical attention network," in *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

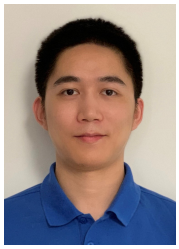
[35] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1441–1450.

[36] 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. 3697–3707.

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

[38] L. Zheng, C.-T. Lu, F. Jiang, J. Zhang, and P. S. Yu, "Spectral collaborative filtering," in *Proceedings of the 12th ACM Conference on Recommender Systems*, 2018, pp. 311–319.

[39] 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.
[40] L. Chen, L. Wu, R. Hong, K. Zhang, and M. Wang, "Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 27–34.
[41] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang, "Session-based social recommendation via dynamic graph attention networks," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 2019, pp. 555–563.
[42] Q. Wu, H. Zhang, X. Gao, P. He, P. Weng, H. Gao, and G. Chen, "Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems," in *The World Wide Web Conference*, 2019, pp. 2091–2102.
[43] 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*, 2019, pp. 417–426.
[44] L. Wu, P. Sun, Y. Fu, R. Hong, X. Wang, and M. Wang, "A neural influence diffusion model for social recommendation," in *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, 2019, pp. 235–244.
[45] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 974–983.
[46] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo, "Ripplenet: Propagating user preferences on the knowledge graph for recommender systems," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 417–426.
[47] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, and Z. Wang, "Knowledge-aware graph neural networks with label smoothness regularization for recommender systems," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 968–977.
[48] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "Kgat: Knowledge graph attention network for recommendation," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 950–958.
[49] D. Liang, J. Altosaar, L. Charlin, and D. M. Blei, "Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence," in *Proceedings of the 10th ACM conference on recommender systems*, 2016, pp. 59–66.
[50] T. Tran, K. Lee, Y. Liao, and D. Lee, "Regularizing matrix factorization with user and item embeddings for recommendation," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 687–696.
[51] M. Gao, L. Chen, X. He, and A. Zhou, "Bine: Bipartite network embedding," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 715–724.
[52] C. Yang, L. Bai, C. Zhang, Q. Yuan, and J. Han, "Bridging collaborative filtering and semi-supervised learning: a neural approach for poi recommendation," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1245–1254.
[53] 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*, 2011, pp. 287–296.
[54] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 650–658.
[55] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, and M. Guo, "Multi-task feature learning for knowledge graph enhanced recommendation," in *The World Wide Web Conference*, 2019, pp. 2000–2010.



Tianyu Zhu is a Ph.D. candidate at Department of Management Science and Engineering, Tsinghua University (Beijing, China). His research interests include machine learning, data mining and recommender systems.

Leilei Sun received his Ph.D. from Institute of Systems Engineering, Dalian University of Technology (Dalian, China) in 2017. Currently, he is an assistant professor at the School of Computer Science, Beihang University (Beijing, China). His research interests include machine learning and data mining.

Guoqing Chen received his PhD from Catholic University of Leuven (KUL, Belgium) in 1992, and currently is chair professor at Department of Management Science and Engineering, Tsinghua University (Beijing China). His research interests include data mining and business analytics, recommender systems, information extraction, fuzzy logic and data modeling.