An Efficient and Effective Framework for Session-based Social Recommendation

Tianwen Chen
tchenaj@cse.ust.hk
The Hong Kong University of Science and Technology
Hong Kong

Raymond Chi-Wing Wong
raywong@cse.ust.hk
The Hong Kong University of Science and Technology
Hong Kong

ABSTRACT

In many applications of session-based recommendation, social networks are usually available. Since users' interests are influenced by their friends, recommender systems can leverage social networks to better understand their users' preferences and thus provide more accurate recommendations. However, existing methods for session-based social recommendation are not efficient. To predict the next item of a user's ongoing session, the methods need to process many additional sessions of the user's friends to capture social influences, while non-social-aware methods (i.e., those without using social networks) only need to process one single session. To solve the efficiency issue, we propose an efficient framework for session-based social recommendation. In the framework, first, a heterogeneous graph neural network is used to learn user and item representations that integrate the knowledge from social networks. Then, to generate predictions, only the user and item representations relevant to the current session are passed to a non-social-aware model. During inference, since the user and item representations can be precomputed, the overall model runs as fast as the original non-social-aware model, while it can achieve better performance by leveraging the knowledge from social networks. Apart from being efficient, our framework has two additional advantages. First, the framework is flexible because it is compatible with any existing non-social-aware models and can easily incorporate more knowledge other than social networks. Second, our framework can capture cross-session item transitions while existing methods can only capture intra-session item transitions. Extensive experiments conducted on three public datasets demonstrate the effectiveness and the efficiency of the proposed framework. Our code is available at https://github.com/twchen/SEFrame.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Social recommendation**.

KEYWORDS

session-based recommendation, social recommendation, social network, graph neural network

WSDM '21, March 8–12, 2021, Virtual Event, Israel

ACM ISBN 978-1-4503-8297-7/21/03…$15.00
https://doi.org/10.1145/3437963.3441792

ACM Reference Format:
Tianwen Chen and Raymond Chi-Wing Wong. 2021. An Efficient and Effective Framework for Session-based Social Recommendation. In *Proceedings of the Fourteenth ACM International Conference on Web Search and Data Mining (WSDM '21), March 8–12, 2021, Virtual Event, Israel.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3437963.3441792

1 INTRODUCTION

The task of *Session-based Recommendation* (SR) is to predict the next action given the previous actions in the same session, where a session is a sequence of actions in close temporal proximity. When SR was initially proposed in [5], user IDs were not utilized because SR was intended for the use cases where user IDs cannot be tracked or most users generate only one or two sessions [6]. In these cases, it is impossible or not meaningful to provide recommendations by mining user-item interactions. Instead, SR learns user preferences from sequential transition patterns in anonymous sessions.

If user IDs can be tracked and most users generate a sufficient number of sessions for the recommender system to learn reliable user preferences, SR can still be applied because it is a common phenomenon that user actions in the same session share a common objective and user actions in different sessions have a weak correlation [2]. Therefore, it is better to provide recommendations based on sessions. Apart from capturing user interests from sequential properties as in standard SR, user IDs can be utilized so that customized recommendations can be made for users with different preferences when the same session prefix is given. This variant of SR can be called *personalized session-based recommendation* (PSR). Since the aforementioned phenomenon exists in many online services such as e-commerce and video sharing websites, SR has great practical value and therefore has attracted much attention recently.

In the scenarios where PSR is applicable, there is usually a social network between users. The service provider itself may have its own social network. For example, in the image sharing app Instagram, the following and being followed relationships between users define a social network. Even if there is no explicit social network in the service, it is still possible to construct a social network by either associating users to external social media platforms (e.g., Facebook) or using existing users' interactions (e.g., in Reddit, two users can be connected if they have replied to each other). The social relationships in the social network can be leveraged to provide more accurate recommendations because users' interests are influenced by their friends and connected users tend to share similar preferences [9, 21, 25]. Therefore, the recommender systems can better understand their users' preferences using the social network. Following the convention [14], we call this variant of SR *session-based social recommendation* (SSR), which has a broad application due to the prevalence of online social networks.

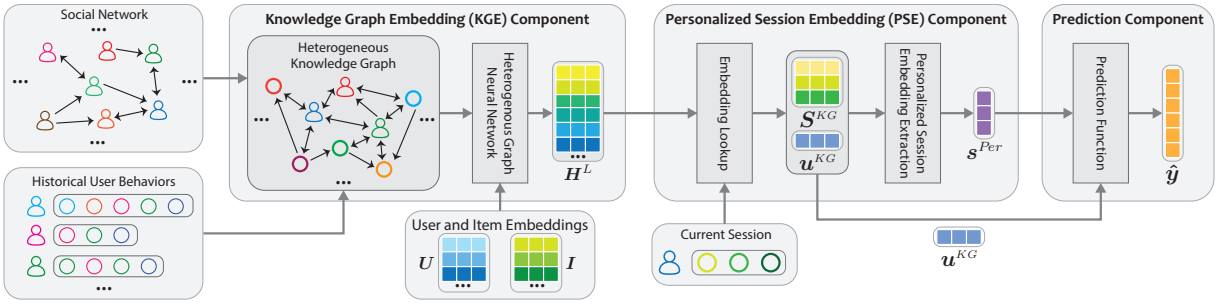

Figure 1: An overview of the proposed framework SEFrame

The general topic, *social recommendation*, has been greatly studied and many effective methods have been proposed. However, there is not yet much related work on the specific topic SSR because it is relatively new. Existing methods for social recommendation are not suitable for SSR because they do not consider the sequential order of user behaviors. DGRec [14] is currently the only method for SSR but it is not efficient. To capture social influences, DGRec uses a GNN to aggregate the preferences of neighbors for each user. Each user's preferences depend on those of their neighbors and the dependency is recursive. If an L-layered GNN is used, the current user's preferences recursively depend on those of at most $\overline{N}^L$ users, where $\overline{N}$ is the number of sampled neighbors for each user at each layer. Since users' preferences are characterized by their most recent sessions, the model needs to process at most $\overline{N}^L$ sessions to predict the next item of a single session. In contrast, non-social-aware methods are much more efficient because they only need to process the current session.

To solve the efficiency issue of DGRec, we propose an efficient framework for SSR, called ***S**ocial-aware **E**fficient **Frame**work (SEFrame)*, whose overview is shown in Figure 1. First, we build a heterogeneous knowledge graph from the social network and all historical user behaviors. Then, we use a heterogeneous graph network to learn user and item representations that fuse the knowledge from social relationships, user-item interactions and item transitions. Given a user and his/her current session, the relevant user and item representations are retrieved and passed to a PSR model. Since the user and item representations are social-aware, the PSR model can leverage the knowledge from the social network to provide recommendations. In this way, we have adapted a PSR model to a SSR model. The framework is efficient because the social-aware user and item representations can be precomputed for inference. Therefore, during inference, the SSR model just need to process the current session, which is as efficient as the original PSR model.

We summarize our contributions as follows.

- We propose an efficient framework for SSR called SEFrame. Using this framework, any existing models for SR can be adapted for SSR. The adapted models can leverage the knowledge from the social network to provide more accurate recommendations, while being as efficient as the original models during inference.
- SEFrame is highly flexible because any existing SR models can be plugged in and it is straightforward to integrate more knowledge in addition to the social network.
- Due to the way that the knowledge graph is constructed, SEFrame can capture cross-session item transitions, while existing methods can only capture intra-session item transitions.
- We also propose an effective model that implements SEFrame. The proposed model could give a higher prediction accuracy than both the baselines that are simple adaptations of existing SR models and the state-of-the-art SSR model, DGRec.
- We conducted extensive experiments to verify the effectiveness and the efficiency of SEFrame. SSR models adapted from existing SR models using SEFrame consistently give a higher prediction accuracy than the original models, while they are as efficient as the original models.

2 RELATED WORK

In this section, we review the related work of session-based recommendation and social recommendation.

Session-based Recommendation: Session-based Recommendation is a sequential modeling problem, for which recurrent neural networks (RNNs) are natural solutions. Hidasi et al. [5] first formally defined session-based recommendation and proposed a multi-layered GRU model. Li et al. [7] incorporated the attention mechanism into GRU to capture users' sequential behaviors and main purposes. Ren et al. [11] considered the repeat consumption phenomenon using a GRU-based model with a repeat-explore mechanism. Convolutional neural networks (CNNs) are also powerful sequential modeling tools. [23] applied dilated convolutional layers to effectively modeling long-range dependencies. Recently, graph neural networks (GNNs) have achieved superior performance in a variety of tasks including session-based recommendation. Wu et al. [19] represented sessions as graphs and applied a gated graph neural network to capture complex item transitions. Chen and Wong [2] solved two information loss problems of graph neural networks methods for session-based recommendation. Since these methods assume that the sessions are anonymous, they could not provide personalized recommendations.

Various attempts have been made to utilize user information in session-based recommendation. Quadrana et al. [10] proposed a hierarchical RNN model to capture users' evolving interests. Wu et al. [20] extended SR-GNN [19] for personalized session-based recommendation and used the attention mechanism to explicitly model the effect of user's historical interests on the current session.

Guo et al. [4] enhanced GRU with matrix factorization to model users' long-term interests. These methods could provide more tailored recommendations by modeling users' long-term and evolving interests, but they could not capture the influences between users in social networks.

Social Recommendation: Many previous studies attempted to leverage social networks to improve the recommendation results. Ma et al. [9] incorporated social networks into recommender systems by regularizing the latent user factors so that connected users have similar latent factors. Zhao et al. [25] extracted additional training instances from the social network for matrix factorization. Wang et al. [17] distinguished and learned the personalized preferences between strong and weak ties in social networks. Xiao et al. [21] adopted transfer learning to model user-item interactions and social relationships simultaneously. Wang et al. [16] enhanced user modeling by integrating the knowledge from multiple heterogeneous social networks. These methods only utilize collaborative information from user-item interactions without considering the sequential order of interactions, and thus they are not suitable for session-based recommendation. Currently, the only method for session-based social recommendation is DGRec [14], which models dynamic user behaviors with an RNN and context-dependent social influences with a graph attention network. However, this method is inefficient because it needs to process many additional sessions to predict the next item of the current session.

3 PROBLEM DEFINITION

In this section, we formally define three variants of session-based recommendation, including anonymous session-based recommendation, personalized session-based recommendation, and session-based social recommendation.

Anonymous Session-based Recommendation (ASR): Let I be the set of items. The dataset of users' historical behaviors $\mathcal{D}$ is a set of anonymous sessions. Each session $S \in \mathcal{D}$ is a sequence of items clicked by an anonymous user, where $S[t] \in I$ denotes the t^{th} item in session S.

Personalized Session-based Recommendation (PSR): Let U and I be the sets of users and items, respectively. The dataset of users' historical behaviors $\mathcal{D}$ contains all sessions of all users. Each user $u \in U$ is associated with a set of sessions denoted by $\mathcal{D}^u = \{S_1^u, S_2^u, \ldots, S_{|\mathcal{D}^u|}^u\}$, where S_T^u is the T^{th} session of u. Each session S_T^u is a sequence of items clicked by user u, where $S_T^u[t] \in I$ denotes the t^{th} item in session S_T^u. For brevity, we may drop the superscript u and/or the subscript T in S_T^u if there is no ambiguity.

Session-based Social Recommendation (SSR): In addition to the dataset of the historical behaviors (i.e., $\mathcal{D}$) in PSR, we have a social network which is a graph $\mathcal{S} = (U, E)$. The set of nodes in $\mathcal{S}$ is the user set U, and the set E of edges represents the social relationships between users. An edge (u, v) from u to v means that u is followed by v.

The objective of all variants is to predict the next item of a new session $S \notin \mathcal{D}$. The prediction model can access all relevant information in $\mathcal{D}$ (and $\mathcal{S}$ if the problem is SSR).

Following previous studies [7, 11, 14, 19], we embed the user IDs and item IDs into low-dimensional latent spaces. To make analysis easier, we set the same dimensionality d for both user and item embeddings. All the embeddings are randomly initialized and learned with other model parameters.

4 SOCIAL-AWARE EFFICIENT FRAMEWORK

In this section, we introduce our efficient framework for SSR called *SEFrame*. SEFrame has three components. The first component learns user and item representations from a heterogeneous knowledge graph. We call this component the *Knowledge Graph Embedding (KGE) component* (Section 4.1) because the learned representations can be viewed as embeddings which fuse knowledge from the heterogeneous graph. The second component called the *Personalized Session Embedding (PSE) component* (Section 4.2) takes as input the relevant user and item representations of a given session and produces a session-specific embedding that captures the user current interests and items' contextual information. The third component called the *prediction component* (Section 4.3) generates a probability distribution of the next item from the session embedding, the user embedding and the item embeddings. Next, we detail each component of SEFrame in Sections 4.1 to 4.3. We present the training process of SEFrame in Section 4.4.

4.1 Knowledge Graph Embedding (KGE) Component

This KGE component involves two tasks. The first task is to construct a heterogeneous knowledge graph $\mathcal{K}$ from all historical user behaviors $\mathcal{D}$ and the social network $\mathcal{S}$. The second task is to learn user and item representations that fuse the knowledge from $\mathcal{K}$ using a heterogeneous graph neural network (HGNN).

Formally, let $\mathcal{K} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R}, \phi, \psi)$ be the heterogeneous knowledge graph. The node set $\mathcal{V} = U \cup I$ consists of all users and items involved in $\mathcal{D}$ and $\mathcal{S}$. The edge set $\mathcal{E}$ contains four types of directed edges, namely *user-user edges*, *user-item edges*, *item-user edges* and *item-item edges*. We reuse the symbols for sets of users and items to denote the types of nodes and edges. Specifically, the set of node types is $\mathcal{A} = \{U, I\}$ and the set of edge types is $\mathcal{R} = \{UU, UI, IU, II\}$. Each edge in $\mathcal{E}$ is associated with a *weight* which is an integer. $\phi : \mathcal{V} \mapsto \mathcal{A}$ is a function that maps a node to its type and $\psi : \mathcal{E} \mapsto \mathcal{R}$ is a function that maps an edge to its type.

The user-user edges represent social relationships between users. A user-user edge $(u, v) \in \mathcal{E}$ if user u is followed by user v. We use "is followed by" instead of "follows" because GNNs update node representations using incoming edges and users are more influenced by the users they follow than those following them. The weight of a user-user edge is defined to be 1. The user-item and item-user edges represent user-item interactions. A user-item edge (u, i) and an item-user edge (i, u) are in $\mathcal{E}$ if user u has clicked item i in some sessions. The weight of each of these two edges is defined to be the number of times that the interaction happens. The item-item edges represent item transitions [19]. An item-item edge $(i, j) \in \mathcal{E}$ if there is a transition from i to j in any session. The weight of this edge to be the number of times that the transition happens.

We include the nodes and edges that must be available under the setting of SSR. It is possible and easy to add other types of nodes and edges that are useful to predict the next items. For example,

if we know the categories of items, we can add nodes for the categories and add edges between items and categories. Therefore, it is straightforward to integrate more knowledge into our framework.

After we obtain the heterogeneous knowledge graph, we do the second task. The second task is to apply a HGNN to learn representations of users and items, where the user representations capture user preferences and the social influences, and the item representations capture collaborative information from user-item interactions and cross-session item transition patterns. We call these representations *knowledge graph embeddings (KG embeddings)*.

4.2 Personalized Session Embedding (PSE) Component

The KG embeddings capture the *global* knowledge in all sessions and the entire social network, without capturing the *session-specific* context information. However, in SR, user behaviors in the current session S are important to capture the user's dynamic interests. Therefore, the PSE component generates a *personalized session-specific embedding* that captures the user's current preferences and the items' contextual properties. It involves two tasks.

The first task is to perform an operation called *embedding lookup* to extract the relevant KG embeddings of the current user and the items in S from the KGE component. These extracted KG embeddings includes (1) the KG embedding of the current user in S, denoted by $\boldsymbol{u}^{KG}$, and (2) the KG embeddings of all items in S, denoted by $\boldsymbol{S}^{KG}[t]$ where $t = 1, 2, \cdots$. The second task is to perform an operation called *personalized session embedding extraction* to compute a personalized session-specific embedding according to the extracted KG embeddings. Specifically, this operation is a function Θ that takes $\boldsymbol{u}^{KG}$ and $\boldsymbol{S}^{KG}$ as input and computes a personalized session-specific embedding $\boldsymbol{s}^{Per}$. That is, $\boldsymbol{s}^{Per} = \Theta(\boldsymbol{u}^{KG}, \boldsymbol{S}^{KG})$.

Any existing SR model, including existing models for PSR and existing modes for ASR, can be easily plugged into our framework. Firstly, existing models for PSR can be directly plugged into our function Θ because the KG embeddings $\boldsymbol{u}^{KG}$ and $\boldsymbol{S}^{KG}$ can be used as the user and item embeddings required by these models. The session representation generated by a PSR model can be used as the personalized session-specific embedding required by this PSE component. Secondly, although *original* models for ASR does not consider any user information, we could adapt each ASR model to a PSR model by appending the user KG embedding $\boldsymbol{u}^{KG}$ to each item KG embedding $\boldsymbol{S}^{KG}[t]$ to obtain the personalized representation of item $\boldsymbol{S}^{KG}[t]$, so that this model could be plugged into our framework too. The personalized item representations are passed as input to the original ASR model. Then, the ASR model can utilize the user information from this personalized item representations and become a PSR model. Thus, this model could be plugged into our framework too.

4.3 Prediction Component

After we obtain the personalized session embedding $\boldsymbol{s}^{Per}$, the prediction component can use it to generate a probability distribution of the next item. Since the user KG embedding $\boldsymbol{u}^{KG}$ can be viewed as the user's long-term interests, which has been shown to be useful to predict the next item in many previous studies [4, 15, 20], we obtain the *final session representation* $\boldsymbol{s}$ from both $\boldsymbol{s}^{Per}$ and $\boldsymbol{u}^{KG}$:

$$\boldsymbol{s} = \text{MLP}\left(\boldsymbol{s}^{Per} \| \boldsymbol{u}^{KG}\right) \tag{1}$$

where $\|$ denotes concatenation and $\text{MLP}(\cdot)$ is a neural network that transforms the concatenated vector to a vector that has the same dimensionality as item embeddings.

To generate the probability distribution of the next item, for each item i with embedding $\boldsymbol{i}$, we compute its score of being the next item of the current session as follows:

$$z_i = \boldsymbol{i}^T \boldsymbol{s} \tag{2}$$

Then, the scores in $\boldsymbol{z}$ are normalized using softmax to obtain a probability distribution $\hat{\boldsymbol{y}}$. That is, $\hat{\boldsymbol{y}} = \text{Softmax}(\boldsymbol{z})$. The items with the top-K probabilities are recommended as the candidates of the next item.

4.4 Training

Let $\boldsymbol{y}$ be the ground-truth probability distribution of the next item, which is a one-hot vector. The loss function is defined to be the cross-entropy of the prediction and the ground truth:

$$\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\boldsymbol{y}^T \log \hat{\boldsymbol{y}} \tag{3}$$

Then, all parameters including the embeddings are randomly initialized and jointly trained in an end-to-end manner using mini-batch stochastic gradient descent.

Therefore, existing SR models can be easily adapted for SSR using our framework. The adapted social-aware models can give a higher prediction accuracy than the original non-social-aware models and even the state-of-the-art SSR model. However, we do not stop here. We further propose a model that implements the framework SEFrame and is able to have a better prediction accuracy than these simple adaptions.

5 SOCIAL-AWARE EFFICIENT RECOMMENDER

In this section, we propose a model called ***S**ocial-aware **E**fficient **Rec**ommender (SERec)* that implements SEFrame by concretely defining the KGE and PSE components. The same prediction component and training step described in Sections 4.3 and 4.4 are used.

5.1 Implementing KGE Component

For the first task of the KGE component, we adopt the same method to construct the heterogeneous knowledge graph described in Section 4.1. For the second task of the KGE component, to learn the representations of user and item nodes in the graph called the *KG embeddings*, we design a HGNN based on the attention mechanism, which consists of L layers.

Let $\boldsymbol{H}^l[v]$ denote the representation of node v at layer l, where v could be either a user or an item. The initial node representations $\boldsymbol{H}^0$ are the user and item embeddings. In the following, we describe the recursive procedure that computes the new node representations $\boldsymbol{H}^l$ at layer l from the old node representations $\boldsymbol{H}^{l-1}$ at layer $l-1$.

At layer l, we compute the new user representations based on two concepts, namely *social influences* and *user preferences*. Firstly, we compute the messages passed between users to capture *social*

influences. The message from user v that u follows is computed as a linear transformation of the node representation of v at layer $l-1$:

$$\text{Message}^l_{UU}(v,u) = \boldsymbol{W}^l_{UU}\boldsymbol{H}^{l-1}[v] + \boldsymbol{b}^l_{UU} \tag{4}$$

where $\boldsymbol{W}^l_{UU} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{b}^l_{UU} \in \mathbb{R}^d$ are learnable parameters.

Secondly, we compute messages passed from items to users to capture *user preferences*. The message from item i that u has clicked before is computed as follows:

$$\text{Message}^l_{IU}(i,u) = \boldsymbol{W}^l_{IU}\boldsymbol{H}^{l-1}[i] + \boldsymbol{b}^l_{IU} \tag{5}$$

where $\boldsymbol{W}^l_{IU} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{b}^l_{IU} \in \mathbb{R}^d$ are learnable parameters. Note that since we consider user preferences which represent how a user prefers different items (or equivalently, how items influence a user), we consider only message passing from items to users (denoting how items influence a user) and thus, we do not need to consider message passing from users to items (denoting how users affects the click of an item).

To consider both social influences and user preferences, it is common to apply a *hierarchical aggregation scheme* [18, 24]. Specifically, two different aggregation functions are applied to gather the social influences from all neighboring users and the user preferences on all neighboring items:

$$\boldsymbol{G}^l_U[u] = \underset{v\in N^{\mathcal{K}}_{in}(u)\cap U}{\text{Aggregate}^l_{UU}}\left(\text{Message}^l_{UU}(v,u)\right) \tag{6}$$

$$\boldsymbol{G}^l_I[u] = \underset{i\in N^{\mathcal{K}}_{in}(u)\cap I}{\text{Aggregate}^l_{IU}}\left(\text{Message}^l_{IU}(i,u)\right) \tag{7}$$

where $N^{\mathcal{K}}_{in}(u)$ denotes the in-neighbors of u in $\mathcal{K}$.

Then, the aggregated information from neighboring users and items is merged using a second-level aggregation:

$$\widetilde{\boldsymbol{H}}^l[u] = \text{Aggregate}^l_U\left(\boldsymbol{G}^l_U[u], \boldsymbol{G}^l_I[u]\right) \tag{8}$$

However, we found that this hierarchical aggregation scheme has a problem because the numbers of neighboring users and items of a user may be imbalanced. For example, user u may have clicked many items but just follow one or two users. In this case, the information from neighboring users is noisy while the information from neighboring items is more reliable. To handle this problem, we propose the *attention aggregation scheme* which can automatically decide to trust the more reliable information source. Specifically, we directly aggregate the messages from both neighboring users and items:

$$\widetilde{\boldsymbol{H}}^l[u] = \underset{v\in N^{\mathcal{K}}_{in}(u),e=(v,u)}{\text{Aggregate}^l_U}\left(\text{Message}^l_{\psi(e)}(v,u)\right) \tag{9}$$

where $\psi(e)$ is the type of edge e.

We define the aggregation function using the attention mechanism. Specifically, we first compute the *importance score* of the message passed along the edge $e=(v,u)$ as follows:

$$\text{Importance}^l_{\psi(e)}(v,u) = \left(\boldsymbol{q}^l_{\psi(e)}\right)^T \sigma\left(\boldsymbol{W}_{\psi(e)}\left(\boldsymbol{H}^{l-1}[v] \| \boldsymbol{H}^{l-1}[u]\right) + \boldsymbol{e}^l\right) \tag{10}$$

where $\boldsymbol{q}^l_{\psi(e)} \in \mathbb{R}^d$ and $\boldsymbol{W}^l_{\psi(e)} \in \mathbb{R}^{d\times 2d}$ are learnable parameters. σ denotes the sigmoid activation function and $\boldsymbol{e}^l \in \mathbb{R}^d$ is the feature vector of edge e at layer l. Here, for each layer l, we embed the weight of each edge into a dense vector as the feature vector $\boldsymbol{e}^l$ of this edge instead of using its original weight value because the influence of the edge on the *attention scores* may not be monotonic. Note that the same edge could have different feature vectors at different layers to have a higher modeling capacity.

The importance scores are normalized using softmax to obtain the attention weights:

$$\text{Attention}^l_{\psi(e)}(v,u) = \underset{v\in N^{\mathcal{K}}_{in}(u)}{\text{Softmax}}\left(\text{Importance}^l_{\psi(e)}(v,u)\right) \tag{11}$$

Then, the influences from all neighboring nodes are computed as the weighted sum of all messages:

$$\widetilde{\boldsymbol{H}}^l[u] = \sum_{v\in N^{\mathcal{K}}_{in}(u),e=(v,u)} \text{Attention}^l_{\psi(e)}(v,u)\cdot\text{Message}^l_{\psi(e)}(v,u) \tag{12}$$

The aggregation information from all neighboring nodes of *items* can be computed similarly to that of *users*. To handle more types of nodes and edges, we generalize the aggregation scheme as follows.

To aggregate the information from all neighboring nodes of a target node u, we first compute the message passed from each neighboring node $v \in N^{\mathcal{K}}_{in}(u)$ along the edge $e=(v,u)$:

$$\text{Message}^l_{\psi(e)}(v,u) = \boldsymbol{W}^l_{\psi(e)}\boldsymbol{H}^{l-1}[v] + \boldsymbol{b}^l_{\psi(e)} \tag{13}$$

which applies an edge-type-specific linear transformation on the feature vector of the source node v, so that feature vectors from different node types are transformed into the same feature space. $\boldsymbol{W}^l_{\psi(e)} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{b}^l_{\psi(e)} \in \mathbb{R}^d$ are learnable parameters for the edge type $\psi(e)$.

Then, the aggregated information $\widetilde{\boldsymbol{H}}^l[u]$ that gathers the messages from all neighbors in $N^{\mathcal{K}}_{in}(u)$ can be computed using Equations (10) to (13). Therefore, it is straightforward to add more types of nodes and items.

The final step is to compute the new node representation from the aggregated information and the old node representation. To do so, we apply a simple node-specific linear transformation followed by the ReLU activation function:

$$\boldsymbol{H}^l[u] = \text{ReLU}\left(\boldsymbol{W}^l_{\phi(u)}\left(\widetilde{\boldsymbol{H}}^l[u] \| \boldsymbol{H}^{l-1}[u]\right) + \boldsymbol{b}^l_{\phi(u)}\right) \tag{14}$$

where $\boldsymbol{W}^l_{\phi(u)} \in \mathbb{R}^{d\times 2d}$ and $\boldsymbol{b}^l_{\phi(u)} \in \mathbb{R}^d$ are learnable parameters for the node type $\phi(u)$.

In this way, we obtain the node representation $\boldsymbol{H}^l[u]$ at the l^{th} HGNN layer for each node u, which captures both social influences from neighboring users and the user's own preferences on neighboring items. By stacking L such HGNN layers, the final node representations $\boldsymbol{H}^L$, called the KG embeddings, capture highly contextualized information within the L-hop community of each node, which are fed into the PSE component to learn personalized session-specific preferences.

5.2 Implementing PSE Component

The objective of the PSE component is to extract the current user's dynamic and personalized preferences in the current session. To do so, we propose a GNN model to learn a personalized session-specific embedding $\boldsymbol{s}^{Per}$. It involves three tasks.

The first task is to construct a weighted directed graph called the *session-specific graph* $G=(V,E)$ based on an ongoing session S of a user u. Specifically, the node set V contains the unique items

in S and edge set E contains an edge (i, j) if there is a transition from item i to item j in S. The *weight* of edge (i, j), denoted by w_{ij}, is the number of occurrences of $i \rightarrow j$ in S.

The second task is to learn a *contextualized feature vector* of each item in S for user u, representing u's *current* interests on items, by performing message passing on the session-specific graph G. The message content of a node in G for the message passing is initialized based on KG embeddings, including both the item KG embeddings (i.e., $\boldsymbol{S}^{KG}[1], \boldsymbol{S}^{KG}[2], ...$) and the user KG embedding (i.e., $\boldsymbol{u}^{KG}$), which could capture u's personalized preferences on items in S. Note that for each item in S (represented by $\boldsymbol{S}^{KG}[t]$), there is a corresponding node i in G. The initial message content is modeled by an *initial feature vector* of node i, denoted by $\boldsymbol{x}_i$, corresponding to item $\boldsymbol{S}^{KG}[t]$ as follows.

$$\boldsymbol{x}_i = \boldsymbol{S}^{KG}[t] \| \boldsymbol{u}^{KG}$$

The message passing mechanism, which could help to capture u's current interests by using the session-specific graph G, is modeled as follows. Let $N^G_{in}(i)$ and $N^G_{out}(i)$ denote the incoming and outgoing neighbors of node i in G, respectively. To learn the contextualized feature vector of i, inspired by [19], we gather the information from both $N^G_{in}(i)$ and $N^G_{out}(i)$:

$$\boldsymbol{a}^{in}_i = \frac{1}{\sum_{k \in N^G_{in}(i)} w_{ki}} \sum_{k \in N^G_{in}(i)} w_{ki} \cdot \boldsymbol{W}_{in} \boldsymbol{x}_k \tag{15}$$

$$\boldsymbol{a}^{out}_i = \frac{1}{\sum_{k \in N^G_{out}(i)} w_{ik}} \sum_{k \in N^G_{out}(i)} w_{ik} \cdot \boldsymbol{W}_{out} \boldsymbol{x}_k \tag{16}$$

$$\boldsymbol{a}_i = \boldsymbol{a}^{in}_i \| \boldsymbol{a}^{out}_i \tag{17}$$

where $\boldsymbol{W}_{in}, \boldsymbol{W}_{out} \in \mathbb{R}^{d \times 2d}$ are learnable parameters and $\boldsymbol{a}_i$ denotes the aggregated information from i's neighboring nodes.

Then, we obtain the contextualized feature vector by applying a gating mechanism to incorporate the information from neighboring nodes (i.e., $\boldsymbol{a}_i$) and the initial feature vector (i.e., $\boldsymbol{x}_i$):

$$\widetilde{\boldsymbol{h}}_i = \tanh\left(\boldsymbol{W}_h \left(\boldsymbol{a}_i \| \boldsymbol{x}_i\right) + \boldsymbol{b}_h\right) \tag{18}$$

$$\boldsymbol{r}_i = \sigma\left(\boldsymbol{W}_r \left(\boldsymbol{a}_i \| \boldsymbol{x}_i\right) + \boldsymbol{b}_r\right) \tag{19}$$

$$\boldsymbol{h}_i = \boldsymbol{r}_i \odot \widetilde{\boldsymbol{h}}_i + (1 - \boldsymbol{r}_i) \odot \boldsymbol{W}_x \boldsymbol{x}_i \tag{20}$$

where $\boldsymbol{W}_h, \boldsymbol{W}_r \in \mathbb{R}^{d \times 4d}$, $\boldsymbol{W}_x \in \mathbb{R}^{d \times 2d}$, and $\boldsymbol{b}_h, \boldsymbol{b}_r \in \mathbb{R}^d$ are learnable parameters. $\odot$ denotes element-wise multiplication, and $\boldsymbol{h}_i$ is the contextualized feature vector of node i.

The third task is to obtain the personalized session-specific embedding $\boldsymbol{s}^{Per}$ by aggregating the contextualized feature vectors of all nodes using the attention mechanism. Specifically, inspired by [19], we use the last item to select the important items in S. Let $\boldsymbol{h}_{last}$ be the contextualized feature vector of the last item in S. Besides, since u's long-term interests are also important to understand u's current focus, we also consider the user KG embedding $\boldsymbol{u}^{KG}$ when the attention mechanism is considered. The *importance score* of node i is defined as:

$$\epsilon_i = \boldsymbol{p}^T \sigma\left(\boldsymbol{W}\left(\boldsymbol{h}_i \| \boldsymbol{h}_{last} \| \boldsymbol{u}^{KG}\right) + \boldsymbol{r}\right) \tag{21}$$

where $\boldsymbol{p}, \boldsymbol{r} \in \mathbb{R}^d$ and $\boldsymbol{W} \in \mathbb{R}^{d \times 3d}$ are learnable parameters.

Then, the personalized session-specific embedding $\boldsymbol{s}^{Per}$ is computed as the weighted sum of all contextualized feature vectors:

$$\boldsymbol{s}^{Per} = \sum_{1 \le t \le |S|} \beta_{i_t} \boldsymbol{h}_{i_t} \tag{22}$$

$$\beta_{i_t} = \underset{1 \le t \le |S|}{\text{Softmax}}(\epsilon_{i_t}) \tag{23}$$

where i_t is the node corresponding to the item at time step t in S.

6 COMPUTATIONAL COMPLEXITY

In this section, we show that the methods that implement SEFrame have lower computational complexities than the state-of-the-art SSR model, DGRec, during both training and inference in a mini-batch setting. We do not include into analysis the step that computes the probability distribution of the next item (i.e., Equation (2)) because this step is the same for all models.

Training Phase: Consider the training phase of DGRec. In Section 1, we have established that DGRec needs to process at most $\overline{N}^L$ sessions to generate recommendations for each session, where $\overline{N}$ is the number of sampled neighboring users for each user at each layer and L is the number of GAT layers. Given a batch of B sessions, DGRec needs to process $O(B\overline{N}^L)$ sessions. Since DGRec employs LSTM to process sessions, the total running time is $O(B\overline{N}^L\overline{L})$, where $\overline{L}$ is the average session length.

Consider the training phase of the methods that implement SEFrame. Given a batch of B sessions, the first step is to obtain the KG embeddings of the users and items involved in the batch (Section 5.1). Since the KG embeddings are not related to a specific session, we just need to find the KG embeddings of the unique users and items, denoted by Q. Suppose that we apply an L-layered HGNN to learn the KG embeddings so that each user can be influenced by his/her L-order neighboring users as in DGRec. For each node $q \in Q$, we sample $\overline{N}$ neighboring nodes of q for each type of neighbors to obtain its final embedding $\boldsymbol{H}^L[q]$:

$$\boldsymbol{H}^L[q] = f(\{\boldsymbol{H}^{L-1}[v] : v \in \{q\} \cup N^L_s(q)\}) \tag{24}$$

where $N^L_s(q)$ is the sampled neighbors of q at layer L, and f composes the functions defined in Equations (10) to (14). It is easy to verify that the complexity of f is $O(|N^L_s(q)|) = O(\overline{N})$, which is proportional to the number of sampled neighbors. Since we sample $O(\overline{N})$ neighbors for each node in Q, there are in total $O(|Q|\overline{N})$ neighbors sampled. Thus, the complexity at layer L is $O(|Q|\overline{N})$.

Since $\boldsymbol{H}^l$ recursively depends on $\boldsymbol{H}^{l-1}$, we need to recursively sample neighbors. As a result, layer 1 has the largest total number of neighbors sampled, and thus the complexity of the first *whole* step is dominated by that of this step at layer 1. Since the total number of sampled neighbors at layer 1 is $O(|Q|\overline{N}^L)$, the running time of the first whole step is $O(|Q|\overline{N}^L)$.

The second step is to generate session-specific embeddings (Section 5.2). Existing methods that are based on GNNs, CNNs and RNNs can compute the session-specific embedding for session S in $O(|S|)$ time, where $|S|$ denotes the number of items in S. Therefore, since $\overline{L}$ is the average session length, the total running time in this step is $O(B\overline{L})$.

The remaining steps which are prediction and training can be finished in $O(B)$ time.

Therefore, the total computational complexity of SEFrame is $O(|Q|\overline{N}^L + B\overline{L} + B) = O(|Q|\overline{N}^L + B\overline{L})$.

Comparison: Since the number of unique users and items is usually less than the number of item occurrences in all sessions (i.e., $|Q| < B\overline{L}$) and $\overline{N} > 1$, we derive that $|Q|\overline{N}^L + B\overline{L}$ is smaller than $B\overline{N}^L\overline{L}$. Therefore, methods that implement SEFrame can be faster than DGRec during training.

Inference Phase: Consider the inference phase of the methods that implement SEFrame. We can *precompute* the KG embeddings, so the computational complexity at inference time is simply dominated by that of the second step in the training phase (i.e., $O(B\overline{L})$).

Consider the inference phase of DGRec. Since most recent sessions of neighboring users could be different even for the different sessions of the same user, precomputation could *not* be done in DGRec. Thus, the running time of DGRec in the inference phase is the same as that in the training phase (i.e., $O(B\overline{N}^L\overline{L})$).

Comparison: Therefore, methods that implement SEFrame runs much faster than DGRec during inference.

7 EXPERIMENTS

In this section, we first describe the experimental settings, and then make detailed analysis on the experimental results.

7.1 Experimental Settings

In this subsection, we describe datasets, compared methods and evaluation metrics for experimental settings.

7.1.1 Datasets. We conducted our experiments on the following three public real-world datasets which are commonly used in the literature of SR and social recommendation [4, 14, 19]: (1) **Gowalla** [13] contains the check-in data and social network on a location-based social networking website. Following [2, 4], we consider two check-in records in different sessions if the time interval between them is longer than 1 day. (2) **Delicious** [1] was collected from an online bookmarking system where users can assign a variety of semantic tags to bookmarks. Following [14], we consider a sequence of tags with timestamps assigned to a bookmark as a session and the task is to provide personalized tag recommendations. (3) **Foursquare** [22] is another large-scale check-in dataset. The social network is collected from an external social media platform. Similar to *Gowalla*, we set the splitting interval to 1 day.

For each dataset, we kept the first 60% sessions as the training set. The remaining sessions were evenly divided into the validation set and the test set. Following [2, 7, 8, 11, 19], we first filtered short sessions and infrequent items and then applied a data augmentation technique described in [2, 7, 8, 19]. Some statistics of the datasets after preprocessing are shown in Table 1.

7.1.2 Compared Methods and Evaluation Metrics. To evaluate the performance of the proposed framework and model, we used the following representative SR methods: (1) **ItemKNN** [3] is a commonly used baseline that recommends items that are most similar to the last item. Each item i is represented as a binary vector $\boldsymbol{x}_i \in \mathbb{R}^M$, where M is the number of sessions and $\boldsymbol{x}_{i,j} = 1$ if item i appears in the j^{th} session. The similarity between two items is defined to be the cosine similarity between their binary vectors. For LBSN datasets, we also used a variant of ItemKNN denoted by "ItemKNN (geo)" which measures the similarity between items by their geographical distances. (2) **FPMC** [12] is a Markov-chain based method for next-basket recommendation. To adapt it for SR, we consider the next item as the next basket. (3) **NextItNet** [23] is a CNN-based method for ASR that models long-range dependencies by dilated convolution. (4) **NARM** [7] is an RNN-based method for ASR that integrates attention into GRU to capture users' main purposes and sequential behaviors. (5) **STAMP** [8] is an ASR model that applies the attention mechanism to better capture users' short-term interests. (6) **SR-GNN** [19] is a method for ASR that utilizes gated graph neural network to capture complex item transitions inside sessions. (7) **SSRM** [4] is the state-of-the-art method for streaming SR. Its MF-based attentive session recommender could be used for PSR. (8) **DGRec** [14] is the state-of-the-art method for SSR that captures users' dynamic interests and context-dependent social influences using RNNs and a graph attention network. We did not include the methods for social recommendation because they are uncompetitive as shown in [14].

Table 1: Statistics of datasets used in the experiments

Statistic	Gowalla	Delicious	Foursquare
# clicks	1,218,599	266,190	3,627,093
# sessions	258,732	60,397	888,798
# users	33,661	1313	39,302
# items	41,229	5793	45,595
# social links	283,778	9130	304,030

Following [2, 8], we applied grid search to find the optimal hyper-parameters for all models using the validation sets. The values we searched were: $\{32, 64, 96, 128\}$ for the embedding size d, $\{10^{-4}, 10^{-3}, \cdots, 10^{-1}\}$ for the learning rate η, and $\{1, 2, 3\}$ for the number of GNN layers L. We used the Adam optimizer to train the models and the batch size was set to 128. We reported models' performance under their optimal hyper-parameter settings.

Following [2, 4, 7, 8, 19, 23], we adopted the commonly used HR@K (Hit Rate at K) and MRR@K (Mean Reciprocal Rank at K) as our evaluation metrics. The values of K included $\{10, 20\}$.

7.2 Effectiveness of SEFrame and SERec

To prove the effectiveness of SEFrame, we plugged the existing non-social-aware SR methods into our framework to obtain social-aware methods for SSR. Then, we compared the performance of the adapted social-aware methods and the original methods. To prove the effectiveness of SERec, we compared SERec with the adapted social-aware methods and the state-of-the-art SSR method, DGRec. The results are shown in Table 2. We denote the social-aware model adapted from an existing non-social-aware model by adding a prefix 'S' to the model name. From the results, we have the following observations.

The simple baselines, ItemKNN and FPMC, which only use the last item for prediction, perform much worse than other methods that can consider all previous items, showing the importance of utilizing the complete sequential information. On LBSN datasets, ItemKNN (geo) has the worse performance, meaning that it is not enough to only consider distances in these datasets.

Table 2: Performance of SR methods in %

Model	Gowalla				Delicious				Foursquare			
	HR@10	MRR@10	HR@20	MRR@20	HR@10	MRR@10	HR@20	MRR@20	HR@10	MRR@10	HR@20	MRR@20
ItemKNN	33.27	18.47	39.11	18.88	20.84	9.98	27.82	10.46	43.88	23.58	52.11	24.15
ItemKNN (geo)	25.07	11.26	32.95	11.80	—	—	—	—	28.51	12.28	37.02	12.87
FPMC	35.31	17.66	42.57	18.17	29.59	14.46	38.26	15.02	44.51	20.93	55.05	21.66
NextItNet	39.87	21.51	47.80	22.04	35.14	18.04	44.62	18.69	52.02	27.67	60.83	28.28
NARM	41.56	22.50	49.55	23.04	37.18	19.76	46.39	20.40	53.63	29.40	62.32	30.00
STAMP	41.93	22.55	49.68	23.10	36.29	19.05	44.96	19.63	53.12	28.32	62.14	29.05
SR-GNN	41.31	22.39	49.31	22.94	37.01	19.57	45.74	20.20	53.19	28.78	62.07	29.40
SSRM	41.63	22.45	49.64	22.98	37.51	19.83	46.57	20.46	53.83	29.33	62.50	29.93
SNextItNet	45.42	24.82	52.93	25.33	38.87	20.84	48.48	21.50	61.11	33.57	69.48	34.16
SNARM	44.51	24.02	52.08	24.54	39.84	21.15	49.41	21.95	60.09	32.90	68.30	33.50
SSTAMP	45.35	24.77	52.84	25.30	38.90	20.57	48.20	21.12	61.30	33.81	69.69	34.40
SSR-GNN	45.46	24.97	52.95	25.49	39.92	21.10	49.35	21.81	61.04	33.47	69.50	34.06
SSSRM	45.01	24.34	52.60	24.87	39.94	21.08	49.26	21.64	60.94	33.48	69.51	34.08
DGRec	42.18	23.04	49.95	23.58	37.78	20.07	47.36	20.73	57.05	31.53	65.85	32.15
SERec	**46.01**	**25.14**	**53.72**	**25.67**	**40.02**	**21.29**	**49.53**	**21.98**	**61.66**	**34.03**	**70.05**	**34.62**

Generally, the more information a model considers, the better it can perform. In most cases, the PSR model, SSRM, outperforms the ASR models, and the SSR models outperform those non-social-aware models. Therefore, it is of great advantage for SR models to consider personalized preferences and social influences when reliable user information can be obtained.

All SSR models adapted from non-social-aware methods significantly outperform their original models and even the state-of-the-art SSR model, DGRec, which strongly proves the effectiveness of SEFrame. DGRec performs better than the non-social-aware models because it leverages the social network to learn more accurate user preferences. However, it does not perform better than the models that implement SEFrame. One possible reason is that in DGRec, the way of using the information from social networks introduces noise to the model. Specifically, some of the most recent sessions of neighbors and the current session may not have any common items so they are totally uncorrelated. Another reason is that models using SEFrame can leverage more knowledge than DGRec. In addition to the social network, we also integrate all historical user-item interactions and cross-session item transitions into the knowledge graph, so the models can learn more accurate user preferences.

The proposed model, SERec, outperforms the simple SSR models that are adapted from non-social-aware methods. Compared with SNextItNet, SNARM, SSTAMP and SSR-GNN, SERec considers the user' long-term interests when learning the personalized session embedding. Compared with SSSRM, SERec computes contextualized item embeddings to understand what features the user is focusing on for each item. Therefore, SERec can learn a better session representation that more accurately captures the user's dynamic and personalized interests in the current session.

7.3 Efficiency of SEFrame

To prove the efficiency of SEFrame, we compared the models' running time during both training and inference on the largest dataset *Foursquare*. The embedding size d is set to 128 and the number of GNN layers L is set to 1. The results are shown in Table 3.

Table 3: Running time in seconds per 1000 batches

Model	Training	Inference	Model	Training	Inference
NextItNet	18.77	5.56	SR-GNN	27.73	26.61
SNextItNet	58.17	5.60	SSR-GNN	50.81	25.96
NARM	11.95	5.08	SSRM	14.63	5.24
SNARM	48.37	5.16	SSSRM	45.71	5.20
STAMP	11.55	4.98	DGRec	62.77	62.85
SSTAMP	49.11	5.06	SERec	54.62	27.52

SSR models adapted from non-social-aware methods run slightly faster than DGRec during training and run as fast as their original models during inference, which is consistent with our theoretical analysis in Section 6. SSR models are much slower than non-social-aware models because they need to process much more information. A larger running time during training is acceptable in practice as long as the model can have a better performance because the model needs to be trained only once in a period. However, a larger running time during inference is less tolerable because inference requires low latency and high throughput in most applications. The inference time of DGRec is more than twice than that of the slowest non-social-aware model, SR-GNN, which greatly limits the practical use of DGRec. In contrast, SSR models adapted from non-social-aware methods using our framework can have a higher prediction accuracy, while the inference time is as fast as that of the original models, making them better choices than DGRec in terms of both accuracy and efficiency. Although the adapted model requires more time and memory for training, during inference, there is no additional time and memory required (the original embeddings can be replaced by the KG embeddings during inference). Therefore, our framework is of great practical value.

7.4 Ablation Study

To evaluate the effectiveness of different parts of SEFrame, we conducted an ablation study. We used SERec in the experiment but the results were similar for other models that implement SEFrame.

We compared the following six variants of SERec with the original SERec: (1) **SERec-noPSE** has no PSE component. The average of all item KG embeddings is used as $\boldsymbol{s}^{Per}$. (2) **SERec-noKGE** has no KGE component. The original user and item embeddings are used as the KG embeddings. (3) **SERec-noUU** has no user-user edges in $\mathcal{K}$. (4) **SERec-noUIIU** has no user-item and item-user edges in $\mathcal{K}$. (5) **SERec-noII** has no item-item edges in $\mathcal{K}$. (6) **SERec-HA** uses the hierarchical aggregation scheme instead of the attention aggregation scheme. The second-level aggregation is just defined as a linear transformation. The results are shown in Table 4.

Table 4: Performance of the variants of SERec

Model	Gowalla		Delicious		Foursquare	
	HR@20	MRR@20	HR@20	MRR@20	HR@20	MRR@20
SERec-noPSE	50.92	23.29	47.09	20.30	68.29	31.16
SERec-noKGE	52.47	24.50	47.32	21.26	68.55	33.29
SERec-noUU	53.49	25.58	48.95	21.87	69.75	34.15
SERec-noUIIU	53.50	25.56	49.13	21.92	68.82	33.58
SERec-noII	52.58	25.32	49.19	21.91	69.35	34.06
SERec-HA	53.56	25.53	49.18	21.96	69.95	34.32
SERec	**53.72**	**25.67**	**49.53**	**21.98**	**70.05**	**34.62**

Variants with either component completely removed, i.e., SERec-noPSE and SERec-noKGE, have the worst performance, indicating both components have a great contribution to the performance. SERec-noPSE performs worse than SERec-noKGE, so users' dynamic interests are more important than social influences. Variants with some edges removed perform better than SERec-noKGE but are still inferior to the complete model SERec, suggesting that it is beneficial to incorporate the knowledge from social influences, user-item interactions and cross-session item transitions. SERec-HA performs worse than SERec, which proves that the proposed attention aggregation scheme is better than the hierarchical aggregation scheme because the attention aggregation scheme can automatically choose the more reliable information source.

8 CONCLUSION

In this paper, we propose an efficient framework called SEFrame for SSR . In the framework, a heterogeneous knowledge graph is constructed from the social network and historical user behaviors. Then, a heterogeneous GNN is applied to learn KG embeddings of users and items that capture the knowledge from social connections, user-item interactions and cross-session item transitions. The KG embeddings can be fed into a SR model to provide more accurate recommendations by leveraging the information from the knowledge graph. Since the KG embeddings can be precomputed, the overall SSR model can be as efficient as the original SR model during inference while being more accurate. Apart from being efficient and effective, SEFrame is also flexible because it is compatible with any existing SR models and it can incorporate more information other than social networks. To further prove the advantages of SEFrame, we propose an implementation of the framework called SERec which has a better prediction accuracy than the simple baselines. Finally, We prove theoretically and empirically the efficiency of SEFrame, and we conduct extensive experiments to show the effectiveness of SEFrame and SERec. In the future, we want to explore the usage of SEFrame in other applications of SR where more information other than social networks are available. Besides, our current framework has a static heterogeneous knowledge graph, which means that it may be less effective in situations where the intra- and inter-dependencies among users and items are constantly evolving. We would study how to adapt our method for a more dynamic setting.

Acknowledgements: We are grateful to the anonymous reviewers for their constructive comments on this paper. The research is supported by 16214017.

REFERENCES

[1] HetRec 2011. 2011. *Delicious.* https://grouplens.org/datasets/hetrec-2011/
[2] Tianwen Chen and Raymond Chi-Wing Wong. 2020. Handling Information Loss of Graph Neural Networks for Session-based Recommendation. In *KDD*. 1172–1180.
[3] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. 2010. The YouTube Video Recommendation System. In *RecSys*. 293–296.
[4] Lei Guo, Hongzhi Yin, Qinyong Wang, Tong Chen, Alexander Zhou, and Nguyen Quoc Viet Hung. 2019. Streaming Session-based Recommendation. In *KDD*. 1569–1577.
[5] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR*.
[6] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations. In *RecSys*. 241–248.
[7] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *CIKM*. 1419–1428.
[8] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In *KDD*. 1831–1839.
[9] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. 2011. Recommender Systems with Social Regularization. In *WSDM*. 287–296.
[10] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In *RecSys*. 130–137.
[11] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten de Rijke. 2019. RepeatNet: A Repeat Aware Neural Recommendation Machine for Session-based Recommendation. In *AAAI*. 4806–4813.
[12] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing Personalized Markov Chains for Next-basket Recommendation. In *WWW*. 811–820.
[13] SNAP. 2010. *Gowalla.* https://snap.stanford.edu/data/loc-gowalla.html
[14] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-based Social Recommendation via Dynamic Graph Attention Networks. In *WSDM*. 555–563.
[15] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *WSDM*. 565–573.
[16] Weiqing Wang, Hongzhi Yin, Xingzhong Du, Wen Hua, Yongjun Li, and Quoc Viet Hung Nguyen. 2019. Online User Representation Learning Across Heterogeneous Social Networks. In *SIGIR*. 545–554.
[17] Xin Wang, Steven C.H. Hoi, Martin Ester, Jiajun Bu, and Chun Chen. 2017. Learning Personalized Preference of Strong and Weak Ties for Social Recommendation. In *WWW*. 1601–1610.
[18] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous Graph Attention Network. In *WWW*. 2022–2032.
[19] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based Recommendation with Graph Neural Network. In *AAAI*. 346–353.
[20] Shu Wu, Mengqi Zhang, Xin Jiang, Ke Xu, and Liang Wang. 2019. Personalizing Graph Neural Networks with Attention Mechanism for Session-based Recommendation. *TKDE* 31 (2019).
[21] Lin Xiao, Zhang Min, Zhang Yongfeng, Liu Yiqun, and Ma Shaoping. 2017. Learning and Transferring Social and Item Visibilities for Personalized Recommendation. In *CIKM*. 337–346.
[22] Dingqi Yang. 2012. *Foursquare.* https://sites.google.com/site/yangdingqi/home/foursquare-dataset
[23] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. 2019. A Simple Convolutional Generative Network for Next Item Recommendation. In *WSDM*. 582–590.
[24] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. 2019. Heterogeneous Graph Neural Network. In *KDD*. 793–803.
[25] Tong Zhao, Julian McAuley, and Irwin King. 2014. Leveraging Social Connections to Improve Personalized Ranking for Collaborative Filtering. In *CIKM*. 261–270.