Session-Based Recommendation with Graph Neural Networks

Shu Wu,[1,2] **Yuyuan Tang,**[3] **Yanqiao Zhu,**[4] **Liang Wang,**[1,2] **Xing Xie,**[5] **Tieniu Tan**[1,2]
[1]Center for Research on Intelligent Perception and Computing
National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
[3]School of Computer and Communication Engineering, University of Science and Technology Beijing
[4]School of Software Engineering, Tongji University
[5]Microsoft Research Asia
shu.wu@nlpr.ia.ac.cn, tangyyuanr@gmail.com, sxkdz@tongji.edu.cn,
wangliang@nlpr.ia.ac.cn, xing.xie@microsoft.com, tnt@nlpr.ia.ac.cn

Abstract

The problem of session-based recommendation aims to predict user actions based on anonymous sessions. Previous methods model a session as a sequence and estimate user representations besides item representations to make recommendations. Though achieved promising results, they are insufficient to obtain accurate user vectors in sessions and neglect complex transitions of items. To obtain accurate item embedding and take complex transitions of items into account, we propose a novel method, i.e. *Session-based Recommendation with Graph Neural Networks*, SR-GNN for brevity. In the proposed method, session sequences are modeled as graph-structured data. Based on the session graph, GNN can capture complex transitions of items, which are difficult to be revealed by previous conventional sequential methods. Each session is then represented as the composition of the global preference and the current interest of that session using an attention network. Extensive experiments conducted on two real datasets show that SR-GNN evidently outperforms the state-of-the-art session-based recommendation methods consistently.

1 Introduction

With the rapid growth of the amount of information on the Internet, recommendation systems become fundamental for helping users alleviate the problem of information overload and select interesting information in many Web applications, e.g., search, e-commerce, and media streaming sites. Most of the existing recommendation systems assume that the user profile and past activities are constantly recorded. However, in many services, user identification may be unknown and only the user behavior history during an ongoing session is available. Thereby, it is of great importance to model limited behavior in one session and generate the recommendation accordingly. Conversely, conventional recommendation methods relying on adequate user-item interactions have problems in yielding accurate results under this circumstance.

Due to the highly practical value, increasing research interests in this problem can be observed, and many kinds of proposals for session-based recommendation have been developed. Based on Markov chains, some work (Shani, Brafman, and Heckerman 2002; Rendle, Freudenthaler, and Schmidt-Thieme 2010) predicts the user's next behavior based on the previous one. With a strong independence assumption, independent combinations of the past components confine the prediction accuracy.

In recent years, the majority of research (Hidasi et al. 2016a; Tan, Xu, and Liu 2016; Tuan and Phuong 2017; Li et al. 2017a) apply Recurrent Neural Networks (RNNs) for session-based recommendation systems and obtain promising results. The work (Hidasi et al. 2016a) proposes a recurrent neural network approach at first, then the model is enhanced by data augmentation and considering temporal shift of user behavior (Tan, Xu, and Liu 2016). Recently, NARM (Li et al. 2017a) designs a global and local RNN recommender to capture user's sequential behavior and main purposes simultaneously. Similar to NARM, STAMP (Liu et al. 2018) also captures users' general interests and current interests, by employing simple MLP networks and an attentive net.

Although the methods above achieve satisfactory results and become the state-of-the-arts, they still have some limitations. *Firstly*, without adequate user behavior in one session, these methods have difficulty in estimating user representations. Usually, the hidden vectors of these RNN methods are treated as the user representations, such that recommendations can be then generated based on these representations, for instance, the global recommender of NARM. In session-based recommendation systems, however, sessions are mostly anonymous and numerous, and user behavior implicated in session clicks is often limited. It is thus difficult to accurately estimate the representation of each user from each session. *Secondly*, previous work reveals that patterns of item transitions are important and can be used as a local factor (Li et al. 2017a; Liu et al. 2018) in session-based recommendation, but these methods always model single-way transitions between consecutive items and neglect the transitions among the contexts, i.e. other items in the session. Thus, complex transitions among distant items are often overlooked by these methods.

To overcome the limitations mentioned above, we propose a novel method for Session-based Recommendation

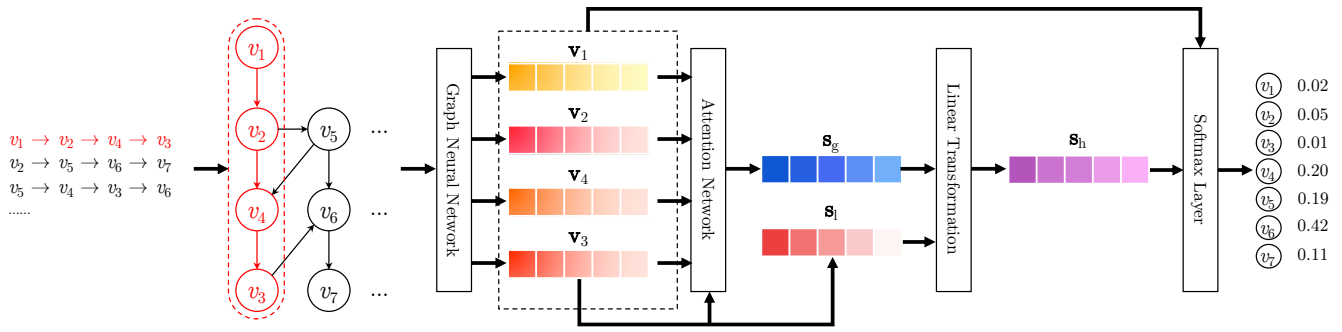


Figure 1: The workflow of the proposed SR-GNN method. We model all session sequences as session graphs. Then, each session graph is proceeded one by one and the resulting node vectors can be obtained through a gated graph neural network. After that, each session is represented as the combination of the global preference and current interests of this session using an attention net. Finally, we predict the probability of each item that will appear to be the next-click one for each session.

with Graph Neural Networks, SR-GNN for brevity, to explore rich transitions among items and generate accurate latent vectors of items. Graph Neural Networks (GNNs) (Scarselli et al. 2009; Li et al. 2015) are designed for generating representations for graphs. Recently, it has been employed to model graph-structured dependencies for natural language processing and computer vision applications flourishingly, e.g., script event prediction (Li, Ding, and Liu 2018), situation recognition (Li et al. 2017b), and image classification (Marino, Salakhutdinov, and Gupta 2017). For the session-based recommendation, we first construct directed graphs from historical session sequences. Based on the session graph, GNN is capable of capturing transitions of items and generating accurate item embedding vectors correspondingly, which are difficult to be revealed by the conventional sequential methods, like MC-based and RNN-based methods. Based on accurate item embedding vectors, the proposed SR-GNN constructs more reliable session representations and the next-click item can be inferred.

Figure 1 illustrates the workflow of the proposed SR-GNN method. At first, all session sequences are modeled as directed session graphs, where each session sequence can be treated as a subgraph. Then, each session graph is proceeded successively and the latent vectors for all nodes involved in each graph can be obtained through gated graph neural networks. After that, we represent each session as a composition of the global preference and the current interest of the user in that session, where these global and local session embedding vectors are both composed by the latent vectors of nodes. Finally, for each session, we predict the probability of each item to be the next click. Extensive experiments conducted on real-world representative datasets demonstrate the effectiveness of the proposed method over the state-of-arts. The main contributions of this work are summarized as follows:

- We model separated session sequences into graph-structured data and use graph neural networks to capture complex item transitions. To best of our knowledge, it presents a novel perspective on modeling in the session-based recommendation scenario.
- To generate session-based recommendations, we do not rely on user representations, but use the session embedding, which can be obtained merely based on latent vectors of items involved in each single session.
- Extensive experiments conducted on real-world datasets show that SR-GNN evidently outperforms the state-of-art methods.

2 Related Work

In this section, we review some related work on session-based recommendation systems, including conventional methods, sequential methods based on Markov chains, and RNN-based methods. Then, we introduce the neural networks on graphs.

Conventional recommendation methods. Matrix factorization (Mnih and Salakhutdinov 2007; Koren, Bell, and Volinsky 2009; Koren and Bell 2011) is a general approach to recommendation systems. The basic objective is to factorize a user-item rating matrix into two low-rank matrices, each of which represents the latent factors of users or items. It is not very suitable for the session-based recommendation, because the user preference is only provided by some positive clicks. The item-based neighborhood methods (Sarwar et al. 2001) is a natural solution, in which item similarities are calculated on the co-occurrence in the same session. These methods have difficulty in considering the sequential order of items and generate prediction merely based on the last click.

Then, the sequential methods based on Markov chains are proposed, which predict users' next behavior based on the previous ones. Treating recommendation generation as a sequential optimization problem, Shani, Brafman, and Heckerman (2002) employ Markov decision processes (MDPs) for the solution. Via factorization of the personalized probability transition matrices of users, FPMC (Rendle, Freudenthaler, and Schmidt-Thieme 2010) models sequential behavior between every two adjacent clicks and provides a more accurate prediction for each sequence. However, the main drawback of Markov-chain-based models is that they combine past components independently. Such an independence assumption is too strong and thus confines the prediction accuracy.

Deep-learning-based methods. Recently, some prediction models, especially language models (Mikolov et al. 2013) are proposed based on neural networks. Among numerous language models, the recurrent neural network (RNN) has been the most successful one in modeling sentences (Mikolov et al. 2010) and has been flourishingly applied in various natural language processing tasks, such as machine translation (Cho et al. 2014), conversation machine (Serban et al. 2016), and image caption (Mao et al. 2015). RNN also has been applied successfully in numerous applications, such as the sequential click prediction (Zhang et al. 2014), location prediction (Liu et al. 2016), and next basket recommendation (Yu et al. 2016).

For session-based recommendation, the work of (Hidasi et al. 2016a) proposes the recurrent neural network approach, and then extends to an architecture with parallel RNNs (Hidasi et al. 2016b) which can model sessions based on the clicks and features of the clicked items. After that, some work is proposed based on these RNN methods. Tan, Xu, and Liu (2016) enhances the performance of recurrent model by using proper data augmentation techniques and taking temporal shifts in user behavior into account. Jannach and Ludewig (2017) combine the recurrent method and the neighborhood-based method together to mix the sequential patterns and co-occurrence signals. Tuan and Phuong (2017) incorporates session clicks with content features, such as item descriptions and item categories, to generate recommendations by using 3-dimensional convolutional neural networks. Besides, A list-wise deep neural network (Wu and Yan 2017) models the limited user behavior within each session, and uses a list-wise ranking model to generate the recommendation for each session. Furthermore, a neural attentive recommendation machine with an encoder-decoder architecture, i.e. NARM (Li et al. 2017a), employs the attention mechanism on RNN to capture users' features of sequential behavior and main purposes. Then, a short-term attention priority model (STAMP) (Liu et al. 2018) using simple MLP networks and an attentive net, is proposed to efficiently capture both users' general interests and current interests.

Neural network on graphs. Nowadays, neural network has been employed for generating representation for graph-structured data, e.g., social network and knowledge bases. Extending the word2vec (Mikolov et al. 2013), an unsupervised algorithm DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) is designed to learn representations of graph nodes based on random walk. Following DeepWalk, unsupervised network embedding algorithms LINE (Tang et al. 2015) and node2vec (Grover and Leskovec 2016) are most representative methods. On the another hand, the classical neural network CNN and RNN are also deployed on graph-structured data. (Duvenaud et al. 2015) introduces a convolutional neural network that operates directly on graphs of arbitrary sizes and shapes. A scalable approach (Kipf and Welling 2016) chooses the convolutional architecture via a localized approximation of spectral graph convolutions, which is an efficient variant and can operate on graphs directly as well. However, these methods can only be implemented on undirected graphs. Previously, in form of recurrent neural networks, Graph Neural Networks (GNNs) (Gori, Monfardini, and Scarselli 2005; Scarselli et al. 2009) are proposed to operate on directed graphs. As a modification of GNN, gated GNN (Li et al. 2015) uses gated recurrent units and employs back-propagation through time (BPTT) to compute gradients. Recently, GNN is broadly applied for the different tasks, e.g., script event prediction (Li, Ding, and Liu 2018), situation recognition (Li et al. 2017b), and image classification (Marino, Salakhutdinov, and Gupta 2017).

3 The Proposed Method

In this section, we introduce the proposed SR-GNN[1] which applies graph neural networks into session-based recommendation. We formulate the problem at first, then explain how to construct the graph from sessions, and finally describe the SR-GNN method thoroughly.

Notations

Session-based recommendation aims to predict which item a user will click next, solely based on the user's current sequential session data without accessing to the long-term preference profile. Here we give a formulation of this problem as below.

In session-based recommendation, let $V = \{v_1, v_2, \ldots, v_m\}$ denote the set consisting of all unique items involved in all the sessions. An anonymous session sequence s can be represented by a list $s = [v_{s,1}, v_{s,2}, \ldots, v_{s,n}]$ ordered by timestamps, where $v_{s,i} \in V$ represents a clicked item of the user within the session s. The goal of the session-based recommendation is to predict the next click, i.e. the sequence label, $v_{s,n+1}$ for the session s. Under a session-based recommendation model, for the session s, we output probabilities $\hat{\mathbf{y}}$ for all possible items, where an element value of vector $\hat{\mathbf{y}}$ is the recommendation score of the corresponding item. The items with top-K values in $\hat{\mathbf{y}}$ will be the candidate items for recommendation.

Constructing Session Graphs

Each session sequence s can be modeled as a directed graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$. In this session graph, each node represents an item $v_{s,i} \in V$. Each edge $(v_{s,i-1}, v_{s,i}) \in \mathcal{E}_s$ means that a user clicks item $v_{s,i}$ after $v_{s,i-1}$ in the session s. Since several items may appear in the sequence repeatedly, we assign each edge with a normalized weighted, which is calculated as the occurrence of the edge divided by the outdegree of that edge's start node. We embed every item $v \in V$ into an unified embedding space and the node vector $\mathbf{v} \in \mathbb{R}^d$ indicates the latent vector of item v learned via graph neural networks, where d is the dimensionality. Based on node vectors, each session s can be represented by an embedding vector $\mathbf{s}$, which is composed of node vectors used in that graph.

Learning Item Embeddings on Session Graphs

Then, we present how to obtain latent vectors of nodes via graph neural networks. The vanilla graph neural network is

[1]To make our results fully reproducible, all source codes have been made public at https://github.com/CRIPAC-DIG/SR-GNN.

proposed by Scarselli et al. (2009), extending neural network methods for processing the graph-structured data. Li et al. (2015) further introduce gated recurrent units and propose gated GNN. Graph neural networks are well-suited for session-based recommendation, because it can automatically extract features of session graphs with considerations of rich node connections. We first demonstrate the learning process of node vectors in a session graph. Formally, for the node $v_{s,i}$ of graph $\mathcal{G}_s$, the update functions are given as follows:

$$\mathbf{a}_{s,i}^t = \mathbf{A}_{s,i:} \left[\mathbf{v}_1^{t-1}, \ldots, \mathbf{v}_n^{t-1}\right]^\top \mathbf{H} + \mathbf{b}, \tag{1}$$

$$\mathbf{z}_{s,i}^t = \sigma\left(\mathbf{W}_z \mathbf{a}_{s,i}^t + \mathbf{U}_z \mathbf{v}_i^{t-1}\right), \tag{2}$$

$$\mathbf{r}_{s,i}^t = \sigma\left(\mathbf{W}_r \mathbf{a}_{s,i}^t + \mathbf{U}_r \mathbf{v}_i^{t-1}\right), \tag{3}$$

$$\widetilde{\mathbf{v}}_i^t = \tanh\left(\mathbf{W}_o \mathbf{a}_{s,i}^t + \mathbf{U}_o \left(\mathbf{r}_{s,i}^t \odot \mathbf{v}_i^{t-1}\right)\right), \tag{4}$$

$$\mathbf{v}_i^t = \left(1 - \mathbf{z}_{s,i}^t\right) \odot \mathbf{v}_i^{t-1} + \mathbf{z}_{s,i}^t \odot \widetilde{\mathbf{v}}_i^t, \tag{5}$$

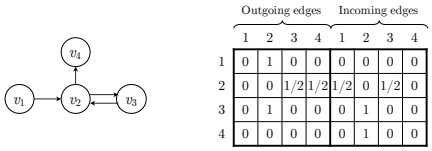
where $\mathbf{H} \in \mathbb{R}^{d \times 2d}$ controls the weight, $\mathbf{z}_{s,i}$ and $\mathbf{r}_{s,i}$ are the reset and update gates respectively, $\left[\mathbf{v}_1^{t-1}, \ldots, \mathbf{v}_n^{t-1}\right]$ is the list of node vectors in session s, $\sigma(\cdot)$ is the sigmoid function, and $\odot$ is the element-wise multiplication operator. $\mathbf{v}_i \in \mathbb{R}^d$ represents the latent vector of node $v_{s,i}$. The connection matrix $\mathbf{A}_s \in \mathbb{R}^{n \times 2n}$ determines how nodes in the graph communicate with each other and $\mathbf{A}_{s,i:} \in \mathbb{R}^{1 \times 2n}$ are the two columns of blocks in $\mathbf{A}_s$ corresponding to node $v_{s,i}$.

Here $\mathbf{A_s}$ is defined as the concatenation of two adjacency matrices $\mathbf{A}_s^{(\text{out})}$ and $\mathbf{A}_s^{(\text{in})}$, which represents weighted connections of outgoing and incoming edges in the session graph respectively. For example, consider a session $s = [v_1, v_2, v_3, v_2, v_4]$, the corresponding graph $\mathcal{G}_s$ and the matrix $\mathbf{A}_s$ are shown in Figure 2. Please note that SR-GNN can support different connection matrices $\mathbf{A}$ for various kinds of constructed session graphs. If different strategies of constructing the session graph are used, the connection matrix $\mathbf{A}_s$ will be changed accordingly. Moreover, when there exists content features of node, such as descriptions and categorical information, the method can be further generalized. To be specific, we can concatenate features with node vector to deal with such information.

For each session graph $\mathcal{G}_s$, the gated graph neural network proceeds nodes at the same time. Eq. (1) is used for information propagation between different nodes, under restrictions given by the matrix $\mathbf{A}_s$. Specifically, it extracts the latent vectors of neighborhoods and feeds them as input into the graph neural network. Then, two gates, i.e. update and reset gate, decide what information to be preserved and discarded respectively. After that, we constructs the candidate state by the previous state, the current state, and the reset gate as described in Eq. (4). The final state is then the combination of the previous hidden state and the candidate state, under the control of the update gate. After updating all nodes in session graphs until convergence, we can obtain the final node vectors.

Generating Session Embeddings

Previous session-based recommendation methods always assume there exists a distinct latent representation of user



	1	2	3	4	1	2	3	4
1	0	1	0	0	0	0	0	0
2	0	0	1/2	1/2	1/2	0	1/2	0
3	0	1	0	0	0	1	0	0
4	0	0	0	0	0	1	0	0

Figure 2: An example of a session graph and the connection matrix $\mathbf{A}_s$

for each session. On the contrary, the proposed SR-GNN method does not make any assumptions on that vector. Instead, a session is represented directly by nodes involved in that session. To better predict the users' next clicks, we plan to develop a strategy to combine long-term preference and current interests of the session, and use this combined embedding as the session embedding.

After feeding all session graphs into the gated graph neural networks, we obtain the vectors of all nodes. Then, to represent each session as an embedding vector $\mathbf{s} \in \mathbb{R}^d$, we first consider the local embedding $\mathbf{s}_\text{l}$ of session s. For session $s = [v_{s,1}, v_{s,2}, \ldots, v_{s,n}]$, the local embedding can be simply defined as $\mathbf{v}_n$ of the last-clicked item $v_{s,n}$, i.e. $\mathbf{s}_\text{l} = \mathbf{v}_n$.

Then, we consider the global embedding $\mathbf{s}_\text{g}$ of the session graph $\mathcal{G}_s$ by aggregating all node vectors. Consider information in these embedding may have different levels of priority, we further adopt the soft-attention mechanism to better represent the global session preference:

$$\begin{aligned} \alpha_i &= \mathbf{q}^\top \sigma(\mathbf{W}_1 \mathbf{v}_n + \mathbf{W}_2 \mathbf{v}_i + \mathbf{c}), \\ \mathbf{s}_\text{g} &= \sum_{i=1}^n \alpha_i \mathbf{v}_i, \end{aligned} \tag{6}$$

where parameters $\mathbf{q} \in \mathbb{R}^d$ and $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d \times d}$ control the weights of item embedding vectors.

Finally, we compute the hybrid embedding $\mathbf{s}_\text{h}$ by taking linear transformation over the concatenation of the local and global embedding vectors:

$$\mathbf{s}_\text{h} = \mathbf{W}_3 \left[\mathbf{s}_\text{l}; \mathbf{s}_\text{g}\right], \tag{7}$$

where matrix $\mathbf{W}_3 \in \mathbb{R}^{d \times 2d}$ compresses two combined embedding vectors into the latent space $\mathbb{R}^d$.

Making Recommendation and Model Training

After obtained the embedding of each session, we compute the score $\hat{\mathbf{z}}_i$ for each candidate item $v_i \in V$ by multiplying its embedding $\mathbf{v}_i$ by session representation $\mathbf{s}_\text{h}$, which can be defined as:

$$\hat{\mathbf{z}}_i = \mathbf{s}_\text{h}^\top \mathbf{v}_i. \tag{8}$$

Then we apply a softmax function to get the output vector of the model $\hat{\mathbf{y}}$:

$$\hat{\mathbf{y}} = \text{softmax}\left(\hat{\mathbf{z}}\right), \tag{9}$$

where $\hat{\mathbf{z}} \in \mathbb{R}^m$ denotes the recommendation scores over all candidate items and $\hat{\mathbf{y}} \in \mathbb{R}^m$ denotes the probabilities of nodes appearing to be the next click in session s.

For each session graph, the loss function is defined as the cross-entropy of the prediction and the ground truth. It can

Table 1: Statistics of datasets used in the experiments

Statistics	*Yoochoose 1/64*	*Yoochoose 1/4*	*Diginetica*
# of clicks	557,248	8,326,407	982,961
# of training sessions	369,859	5,917,745	719,470
# of test sessions	55,898	55,898	60,858
# of items	16,766	29,618	43,097
Average length	6.16	5.71	5.12

be written as follows:

$$\mathcal{L}(\hat{\mathbf{y}}) = -\sum_{i=1}^{m} \mathbf{y}_i \log(\hat{\mathbf{y}}_i) + (1-\mathbf{y}_i)\log(1-\hat{\mathbf{y}}_i), \quad (10)$$

where $\mathbf{y}$ denotes the one-hot encoding vector of the ground truth item.

Finally, we use the Back-Propagation Through Time (BPTT) algorithm to train the proposed SR-GNN model. Note that in session-based recommendation scenarios, most sessions are of relatively short lengths. Therefore, it is suggested to choose a relatively small number of training steps to prevent overfitting.

4 Experiments and Analysis

In this section, we first describe the datasets, compared methods, and evaluation metrics used in the experiments. Then, we compare the proposed SR-GNN with other comparative methods. Finally, we make detailed analysis of SR-GNN under different experimental settings.

Datasets

We evaluate the proposed method on two real-world representative datasets, i.e. *Yoochoose*[2] and *Diginetica*[3]. The Yoochoose dataset is obtained from the RecSys Challenge 2015, which contains a stream of user clicks on an e-commerce website within 6 months. The Diginetica dataset comes from CIKM Cup 2016, where only its transactional data is used.

For fair comparison, following (Li et al. 2017a; Liu et al. 2018), we filter out all sessions of length 1 and items appearing less than 5 times in both datasets. The remaining 7,981,580 sessions and 37,483 items constitute the Yoochoose dataset, while 204,771 sessions and 43097 items construct the Diginetica dataset. Furthermore, similar to (Tan, Xu, and Liu 2016), we generate sequences and corresponding labels by splitting the input sequence. To be specific, we set the sessions of subsequent days as the test set for Yoochoose, and the sessions of subsequent weeks as the test set for Diginetiva. For example, for an input session $s = [v_{s,1}, v_{s,2}, \ldots, v_{s,n}]$, we generate a series of sequences and labels $([v_{s,1}], v_{s,2}), ([v_{s,1}, v_{s,2}], v_{s,3}), \ldots, ([v_{s,1}, v_{s,2}, \ldots, v_{s,n-1}], v_{s,n})$, where $[v_{s,1}, v_{s,2}, \ldots, v_{s,n-1}]$ is the generated sequence and $v_{s,n}$ denotes the next-clicked item, i.e. the label of the sequence. Following (Li et al. 2017a; Liu et al. 2018), we also use the most recent fractions 1/64 and 1/4 of the training sequences of Yoochoose. The statistics of datasets are summarized in Table 1.

[2]http://2015.recsyschallenge.com/challege.html

[3]http://cikm2016.cs.iupui.edu/cikm-cup

Baseline Algorithms

To evaluate the performance of the proposed method, we compare it with the following representative baselines:

- **POP** and **S-POP** recommend the top-N frequent items in the training set and in the current session respectively.
- **Item-KNN** (Sarwar et al. 2001) recommends items similar to the previously clicked item in the session, where similarity is defined as the cosine similarity between the vector of sessions.
- **BPR-MF** (Rendle et al. 2009) optimizes a pairwise ranking objective function via stochastic gradient descent.
- **FPMC** (Rendle, Freudenthaler, and Schmidt-Thieme 2010) is a sequential prediction method based on markov chain.
- **GRU4REC** (Hidasi et al. 2016a) uses RNNs to model user sequences for the session-based recommendation.
- **NARM** (Li et al. 2017a) employs RNNs with attention mechanism to capture the user's main purpose and sequential behavior.
- **STAMP** (Liu et al. 2018) captures users' general interests of the current session and current interests of the last click.

Evaluation Metrics

- **P@20** (Precision) is widely used as a measure of predictive accuracy. It represents the proportion of correctly recommended items amongst the top-20 items.
- **MRR@20** (Mean Reciprocal Rank) is the average of reciprocal ranks of the correctly-recommended items. The reciprocal rank is set to 0 when the rank exceeds 20. The MRR measure considers the order of recommendation ranking, where large MRR value indicates that correct recommendations in the top of the ranking list.

Parameter Setup

Following previous methods (Li et al. 2017a; Liu et al. 2018), we set the dimensionality of latent vectors $d = 100$ for both datasets. Besides, we select other hyper-parameters on a validation set which is a random 10% subset of the training set. All parameters are initialized using a Gaussian distribution with a mean of 0 and a standard deviation of 0.1. The mini-batch Adam optimizer is exerted to optimize these parameters, where the initial learning rate is set to 0.001 and will decay by 0.1 after every 3 epochs. Moreover, the batch size and the L2 penalty is set to 100 and 10^{-5} respectively.

Comparison with Baseline Methods

To demonstrate the overall performance of the proposed model, we compare it with other state-of-art session-based recommendation methods. The overall performance in terms of P@20 and MRR@20 is shown in Table 2, with the best results highlighted in boldface. Please note that, as in (Li et al. 2017a), due to insufficient memory to initialize FPMC, the performance on Yoochoose 1/4 is not reported.

SR-GNN aggregates separated session sequences into graph-structured data. In this model, we jointly consider the

Table 2: The performance of SR-GNN with other baseline methods over three datasets

Method	Yoochoose 1/64		Yoochoose 1/4		Diginetica	
	P@20	MRR@20	P@20	MRR@20	P@20	MRR@20
POP	6.71	1.65	1.33	0.30	0.89	0.20
S-POP	30.44	18.35	27.08	17.75	21.06	13.68
Item-KNN	51.60	21.81	52.31	21.70	35.75	11.57
BPR-MF	31.31	12.08	3.40	1.57	5.24	1.98
FPMC	45.62	15.01	–	–	26.53	6.95
GRU4REC	60.64	22.89	59.53	22.60	29.45	8.33
NARM	68.32	28.63	69.73	29.23	49.70	16.17
STAMP	68.74	29.67	70.44	30.00	45.64	14.32
SR-GNN	**70.57**	**30.94**	**71.36**	**31.89**	**50.73**	**17.59**

global session preference as well as the local interests. According to the experiments, it is obvious that the proposed SR-GNN method achieves the best performance on all three datasets in terms of P@20 and MRR@20. This verifies the effectiveness of the proposed method.

The performance of traditional algorithms like POP and S-POP is relatively poor. Such simple models make recommendations solely based on repetitive co-occurred items or successive items, which is problematic in session-based recommendation scenarios. Even so, S-POP still outperforms its opponents such as POP, BPR-MF, and FPMC, demonstrating the importance of session contextual information. Item-KNN achieves better results than FPMC which is based on Markov chains. Please note that, Item-KNN only utilizes the similarity between items without considering sequential information. This indicates that the assumption on the independence of successive items, which traditional MC-based methods mostly rely on, is not realistic.

Neural-network-based methods, such as NARM and STAMP, outperform the conventional methods, demonstrating the power of adopting deep learning in this domain. Short/long-term memory models, like GRU4REC and NARM, use recurrent units to capture a user's general interest while STAMP improves the short-term memory by utilizing the last-clicked item. Those methods explicitly model the users' global behavioral preferences and consider transitions between users' previous actions and the next click, leading to superior performance against these traditional methods. However, their performance is still inferior to that of the proposed method. Compared with the state-of-art methods like NARM and STAMP, SR-GNN further considers transitions between items in a session and thereby models every session as a graph, which can capture more complex and implicit connections between user clicks. Whereas in NARM and GRU4REC, they explicitly model each user and obtain the user representations through separated session sequences, with possible interactive relationships between items ignored. Therefore, the proposed model is more powerful to model session behavior.

Besides, SR-GNN adopts the soft-attention mechanism to generate a session representation which can automatically select the most significant item transitions, and neglect noisy and ineffective user actions in the current session. On the contrary, STAMP only uses the transition between the last-clicked item and previous actions, which may not be sufficient. Other RNN models, such as GRU4REC and NARM, fail to select impactful information during the propagation process as well. They use all previous items to obtain a vector representing the user's general interest. When a user's behavior is aimless, or his interests drift quickly in the current session, conventional models are ineffective to cope with noisy sessions.

Comparison with Variants of Connection Schemes

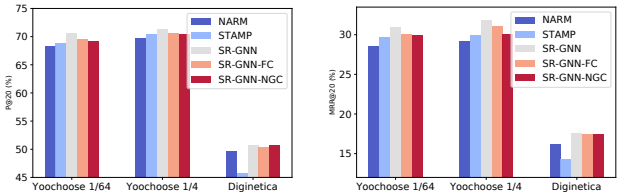
The proposed SR-GNN method is flexible in constructing connecting relationships between items in the graph. We show another two connection variants in order to evaluate relationships between items in each session graph. Firstly, we aggregate all session sequences together and model them as a directed whole item graph, which is termed as the global graph hereafter. In the global graph, each node denotes a unique item, and each edge denotes a directed transition from one item to another. Secondly, we model all high-order relationships between items within one session as direct connections explicitly. In summary, the following two connection schemes are proposed to compare with SR-GNN:

- SR-GNN with normalized global connections (SR-GNN-NGC) replaces the connection matrix with edge weights extracted from the global graph on the basis of SR-GNN.
- SR-GNN with full connections (SR-GNN-FC) represents all higher-order relationships using boolean weights and appends its corresponding connection matrix to that of SR-GNN.

The results of different connection schemes are shown in Figure 3. From the figures, it is seen that all three connection schemes achieve better or almost the same performance as the state-of-the-art STAMP and NARM methods, confirming the usefulness of modeling sessions as graphs.

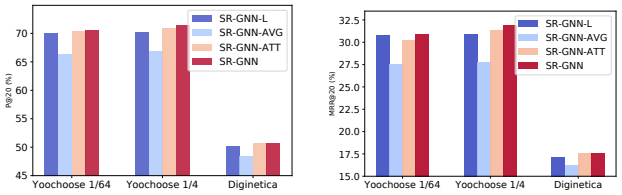
Compared with SR-GNN, for each session, SR-GNN-NGC takes the impact of other sessions into considerations in addition to items in the current session, which subsequently reduces the influence of edges that are connected to nodes with high degree within the current session graph. Such a fusion method notably affects the integrity of the current session, especially when the weight of the edge in the graph varies, leading to performance downgrade.

In regard to SR-GNN and SR-GNN-FC, the former one only models the exact relationship between consecutive items, and the latter one further explicitly regards all high-order relationships as direct connections. It is reported that SR-GNN-FC performs worse than SR-GNN, though the experimental results of the two methods are not of much difference. Such a small difference in results suggests that in most recommendation scenarios, not every high-order transitions can be directly converted to straight connections and intermediate stages between high-order items are still necessities. For instance, considering that the user has viewed the following pages when browsing a website: $A \rightarrow B \rightarrow C$, it is not appropriate to recommend page C directly after A without intermediate page B, due to the lack of a direct connection between A and C.



(a) P@20 (b) MRR@20

Figure 3: The performance of different connection schemes



(a) P@20 (b) MRR@20

Figure 4: The performance of different session representations

Comparison with Different Session Embeddings

We compare the session embedding strategy with the following three approaches: (1) local embedding only (SR-GNN-L), (2) global embedding with average pooling (SR-GNN-AVG), and (3) global embedding with the attention mechanism (SR-GNN-ATT). The results of methods with three different embedding strategies are given in Figure 4.

From the figures, it can be observed that the hybrid embedding method SR-GNN achieves best results on all three datasets, which validates the importance of explicitly incorporating current session interests with the long-term preference. Furthermore, the figures show that SR-GNN-ATT performs better than SR-GNN-AVG with average pooling on three datasets. It indicates that the session may contain some noisy behavior, which cannot be treated independently. Besides, it is shown that attention mechanisms are helpful in extracting the significant behavior from the session data to construct the long-term preference.

Please note that SR-GNN-L, a downgraded version of SR-GNN, still outperforms SR-GNN-AVG and achieves almost the same performance as that of SR-GNN-ATT, supporting that both the current interest and long-term preference are crucial for session-based recommendation.

Analysis on Session Sequence Lengths

We further analyze the capability of different models to cope with sessions of different lengths. For comparison, we partition sessions of Yoochoose 1/64 and Diginetica into two groups, where "Short" indicates that the length of sessions is less than or equal to 5, while each session has more than 5 items in "Long". The pivot value 5 is the closest integer to the average length of sessions in all datasets. The percentages of session belonging to short group and long group are 0.701 and 0.299 on the Yoochoose data, and 0.764 and 0.236 on the Diginetica data. For each method, we report the results evaluated in terms of P@20 in Table 3.

Table 3: The performance of different methods with different session lengths evaluated in terms of P@20

Method	Yoochoose 1/64		Diginetica	
	Short	Long	Short	Long
NARM	**71.44**	60.79	**51.22**	45.75
STAMP	70.69	64.73	47.26	40.39
SR-GNN-L	70.11	69.73	49.04	50.97
SR-GNN-ATT	70.31	70.64	50.35	51.05
SR-GNN	70.47	**70.70**	50.49	**51.27**

Our proposed SR-GNN and its variants perform stably on two datasets with different session lengths. It demonstrates the superior performance of the proposed method and the adaptability of graph neural networks in session-based recommendation. On the contrary, the performance of NARM and STAMP changes greatly in short and long groups. STAMP explains such a difference according to replicated actions. It adopts the attention mechanism, so replicated items can be ignored when obtaining user representations. Similar to STAMP, NARM achieves good performance on the short group, but the performance drops quickly with the length of the sessions increasing, which is partially because RNN models have difficulty in coping with long sequences.

Then we analyze the performance of SR-GNN-L, SR-GNN-ATT, and SR-GNN with different session representations. These three methods achieve promising results comparing with STAMP and NARM. It is probably because that based on the learning framework of graph neural networks, our methods can attain more accurate node vectors. Such node embedding not only captures the latent features of nodes but also models the node connections globally. On such basis, the performance is stable among variants of SR-GNN, while the performance of two state-of-art methods fluctuate considerably on short and long datasets. Moreover, the table shows that SR-GNN-L can also achieve good results, although this variant only uses local session embedding vectors. It is maybe because that SR-GNN-L also implicitly considers the properties of the first-order and higher-order nodes in session graphs. Such results are also validated by Figure 4, where both SR-GNN-L and SR-GNN-ATT achieve the close-to-optimal performance.

5 Conclusions

Session-based recommendation is indispensable where users' preference and historical records are hard to obtain. This paper presents a novel architecture for session-based recommendation that incorporates graph models into representing session sequences. The proposed method not only considers the complex structure and transitions between items of session sequences, but also develops a strategy to combine long-term preferences and current interests of sessions to better predict users' next actions. Comprehensive

experiments confirm that the proposed algorithm can consistently outperform other state-of-art methods.

Acknowledgements

The authors Shu Wu and Yuyuan Tang contribute to this work equally. Yanqiao Zhu is the corresponding author. This work is supported by National Natural Science Foundation of China (61772528) and National Key Research and Development Program (2016YFB1001000).

References

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *EMNLP* 1724–1734.

Duvenaud, D.; Maclaurin, D.; Aguilera-Iparraguirre, J.; Gómez-Bombarelli, R.; Hirzel, T.; Aspuru-Guzik, A.; and Adams, R. P. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*.

Gori, M.; Monfardini, G.; and Scarselli, F. 2005. A new model for learning in graph domains. In *IJCNN*, volume 2, 729–734.

Grover, A., and Leskovec, J. 2016. Node2vec: Scalable feature learning for networks. In *KDD*, 855–864.

Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; and Tikk, D. 2016a. Session-based recommendations with recurrent neural networks. In *ICLR*.

Hidasi, B.; Quadrana, M.; Karatzoglou, A.; and Tikk, D. 2016b. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *RecSys*, 241–248.

Jannach, D., and Ludewig, M. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *RecSys*, 306–310.

Kipf, T. N., and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Koren, Y., and Bell, R. 2011. Advances in collaborative filtering. In *Recommender Systems Handbook*. Springer. 145–186.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer* 42(8):30–37.

Li, Y.; Tarlow, D.; Brockschmidt, M.; and Zemel, R. S. 2015. Gated graph sequence neural networks. In *ICLR*.

Li, J.; Ren, P.; Chen, Z.; Ren, Z.; Lian, T.; and Ma, J. 2017a. Neural attentive session-based recommendation. In *CIKM*, 1419–1428.

Li, R.; Tapaswi, M.; Liao, R.; Jia, J.; Urtasun, R.; and Fidler, S. 2017b. Situation recognition with graph neural networks. In *ICCV*, 4183–4192.

Li, Z.; Ding, X.; and Liu, T. 2018. Constructing narrative event evolutionary graph for script event prediction.

Liu, Q.; Wu, S.; Wang, L.; and Tan, T. 2016. Predicting the next location: A recurrent model with spatial and temporal contexts. In *AAAI*, 194–200.

Liu, Q.; Zeng, Y.; Mokhosi, R.; and Zhang, H. 2018. Stamp: Short-term attention/memory priority model for session-based recommendation. In *KDD*, 1831–1839.

Mao, J.; Xu, W.; Yang, Y.; Wang, J.; Huang, Z.; and Yuille, A. 2015. Deep captioning with multimodal recurrent neural networks (m-rnn). *ICLR*.

Marino, K.; Salakhutdinov, R.; and Gupta, A. 2017. The more you know: Using knowledge graphs for image classification. In *CVPR*, 20–28.

Mikolov, T.; Karafiát, M.; Burget, L.; Cernockỳ, J.; and Khudanpur, S. 2010. Recurrent neural network based language model. In *INTERSPEECH*, volume 2, 3.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, 3111–3119.

Mnih, A., and Salakhutdinov, R. 2007. Probabilistic matrix factorization. In *NIPS*, 1257–1264.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *KDD*, 701–710.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, 452–461.

Rendle, S.; Freudenthaler, C.; and Schmidt-Thieme, L. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, 811–820. ACM.

Sarwar, B.; Karypis, G.; Konstan, J.; and Riedl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2009. The graph neural network model. *TNN* 20(1):61–80.

Serban, I. V.; Sordoni, A.; Bengio, Y.; Courville, A.; and Pineau, J. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, 3776–3784.

Shani, G.; Brafman, R. I.; and Heckerman, D. 2002. An mdp-based recommender system. In *UAI*, 453–460.

Tan, Y. K.; Xu, X.; and Liu, Y. 2016. Improved recurrent neural networks for session-based recommendations. In *DLRS*, 17–22.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *WWW*, 1067–1077.

Tuan, T. X., and Phuong, T. M. 2017. 3d convolutional networks for session-based recommendation with content features. In *RecSys*, 138–146.

Wu, C., and Yan, M. 2017. Session-aware information embedding for e-commerce product recommendation. In *CIKM*, 2379–2382.

Yu, F.; Liu, Q.; Wu, S.; Wang, L.; and Tan, T. 2016. A dynamic recurrent basket recommendation model. In *SIGIR*. ACM.

Zhang, Y.; Dai, H.; Xu, C.; Feng, J.; Wang, T.; Bian, J.; Wang, B.; and Liu, T.-Y. 2014. Sequential click prediction for sponsored search with recurrent neural networks. In *AAAI*, 1369–1376.