Joint Modeling of Local and Global Behavior Dynamics for Session-based Recommendation*

Yong Xu¹ and Jiahui Chen² and Chao Huang³ and Bo Zhang⁴ and Hao Xing⁵ and Peng Dai⁶ and Liefeng Bo⁷

Abstract. Session-based recommendation is critical in modern recommender systems, which aims to predict the next interested item given anonymous behavior sequences of users. While prior works have made efforts to addressing the session-based recommendation problem, two significant limitations exist: i) They ignore the fact that items may be correlated with other across different session units; ii) existing solutions are also limited in their assumption of rigidly ordered pattern over intra-session item transition, which may not be true in practice. To address these above limitations, we propose a Local-Global Session-based Recommendation framework—LGSR which generalizes the modeling of behavior dynamics from two perspectives: we first design a cross-session item dependency encoder to learn the inter-session item relationship structures from a global perspective. Additionally, a dual-stage attentive aggregation module is developed to capture local item transition dynamics, without the restriction of rigid sequential process for jointly modeling user’s current interest and intra-session purpose. With the exploration of both complex intra- and inter-session interest transitional regularities, our LGSR model enables the representation learning of user behavior dynamics via jointly mapping local and global signals into the same latent space. The experimental results on two real-world datasets demonstrate the superiority of the proposed LGSR framework over state-of-the-art methods.

1 Introduction

To alleviate the information explosion and identify the items for users with their personalized interests, modeling user’s preferences over items based on their historical interactions has become increasing popular in recent real-world recommender systems, such as e-commerce platforms [15, 14], online movie sites [2] and location-based services [37]. Under the realistic circumstances that specific user information is not always available (due to privacy issues), conventional recommendation strategies (e.g., collaborative filtering-based methods [11, 36]) can hardly generate promised results. In such cases, session-based recommendation has become a key task with the aim of predicting the next item and making recommendations based on anonymous behavior sequences (i.e., clicked items) from a short-term period [12, 20, 23, 35].

To model sequential dynamics of user behaviors, many session-based recommendation methods have been developed to capture various sequential transition regularities of user behavioral data. In particular, recurrent neural networks (e.g., GRU) have been utilized to model non-linear sequential correlations between past and future user behavior [12]. To extract user’s main purpose in the current session, attention mechanisms serve as key techniques to be integrated with recurrent framework as a hybrid encoder for modeling users’ sequential preferences [23, 20]. In addition, another line of session-based recommendation model leverages the graph neural networks to capture complex transition relations between items for modeling structured session data [35].

Despite the effectiveness of the aforementioned approaches, we argue that two key limitations exist in these methods. First, they only focus on the item sequential transitions within a single session, which makes them insufficient to distill cross-session collaborative signals from the users’ collective behaviors. In real-world session-based recommendations, any pair of user’s interested items could potentially be related across different session units [32]. For example, item v₁ and v₂ is clicked in chronological order (v₁ → v₂) in session A. In another session B, item v₃ is browsed right after v₂ (v₂ → v₃). While there is no explicit intra-session sequential transitions between item v₁ and v₃, they are no longer independent with each other, and implicit relationship between v₁ and v₃ should be considered to accurately capture user’s dynamic interests. Hence, the complex item sequential transition regularities are often exhibited with high-order relation structure from not only the intra-session dependencies (local transitional information) but also the inter-session correlations (global transitional information) [34]. The failure of jointly modeling local and global item transitional signals leads to suboptimal recommendation results.

Second, another deficiency of existing session-based recommendation models lies in the rigid order assumption of item transitional relationships. However, user’s dynamic preferences are affected by many complex unobservable factors [4] and may not follow a rigid order assumption in practical recommendation scenarios [26, 30]. The utilization of current recurrent framework and its extensions
(e.g., attentive recurrent network) assume that a rigidly temporally ordered pattern for item sequences, i.e., user’s preference is propagated in a sequential manner. This assumption limits the representation ability of existing deep recommendation techniques, and it is likely that the learned dynamic users’ behavioral patterns are inaccurate. Therefore, it would be really valuable if a session-based recommendation model could recognize such behavior dynamics without the rigid order assumption of item transition regularities.

With the consideration of existing session-based recommendation methods, we believe that it is of critical importance to develop a framework that allows the modeling of both local and global user behavior dynamics in an explicit and end-to-end manner. Towards this end, we propose a framework, Local-Global Session-based Recommendation (LGSR), to jointly perform global item relation structure learning and local dynamic item transition modeling for accurate session-based recommendations. Specifically, LGSR is equipped with two design choices to correspondingly address the challenges in local-global behavior dynamics modeling: i) cross-session item dependency encoder, which aims to learn item contextual representations with the preservation of implicit correlations between items across different sessions. ii) hierarchical attentive aggregation module, which is a dual-stage attention network with the cooperation of the self-attention mechanism and another attentive aggregation layer, in order to capture both user’s current preference and session-specific purpose. Our LGSR is conceptually advantageous to existing methods in that both the local (intra-session item transitions) and global (inter-session item dependencies) item high-order relations are factored into the recommendation model.

The contributions of this work are summarized as follows:

- We provide a principled way to exploit both local and global behavior dynamics of users in the session-based recommendation framework.
- We propose a new framework LGSR, which simultaneously performs global item relation structure learning by maximizing the likelihood of preserving cross-session item correlations, and local dynamic item transition modeling via a hierarchically structured attentive aggregation module.
- We perform extensive experiments on two real-world datasets for session-based recommendation to validate the rationality by joint learning of local-global item transition relationships. Experimental results demonstrate the effectiveness and interpretability of our developed LGSR framework.

2 Methodology

We first present the problem formulation and the model overview. Then, we explain key modules of LGSR in details.

2.1 Problem Formulation

The goal of the session-based recommendation is to predict the item that the user will interested in (e.g., click) at the next time step based on their historical behavior sequences. Generally, it recommends $k$ items that users may be interested in from the item candidate set $V = \{v_1, v_2, ..., v_n\}$, where $n$ is the number of items. This problem is formalized as follows: Given a temporally-ordered item sequence $s = [v_{s,1}, v_{s,2}, ..., v_{s,t}]$, where $v_{s,i} \in V$ denotes the $i$-th item clicked by the user in the session $s$, and $t$ denotes the length of $s$, the session-based recommendation aims to output a list $Y = [y_1, y_2, ..., y_n]$ based on the session $s$, where $y_i$ denotes the probability of item $v_i$ will be clicked by user. The recommendation result is a set of items with top-$k$ probability values in $Y$.

2.2 Framework Overview

Our developed LGSR framework consists of two major modules: cross-session item dependency encoder and hierarchically structured attentive aggregation module. The architecture of LGSR is shown in Figure 1. We first devise a cross-session item dependency encoder to model the global item relation structures. This module aims to learn global context-aware item representations based on the cross-session item graph, by maximizing the likelihood of preserving item correlations across session units. Furthermore, we propose a dual-stage attention network to capture user’s dynamic preferences and session-specific main purpose. In the architecture of LGSR, these two modules cooperate with each other by sharing a embedding layer.

2.3 Cross-Session Item Dependency Encoder

We formulate a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with items as nodes from all historical sessions $\mathcal{S} = \{s_1, s_2, ..., s_m\}$, where $m$ is the number of historical sessions. Each session can be regarded as a path which starts from the first item and ends at the last item in $\mathcal{G}$. The global context of item relations in graph $\mathcal{G}$ helps us to learn inter-session item transitions. As shown in Figure 1, we can see that before clicking $v_4$, the click on item $v_1$, $v_2$ appear in different sessions, which indicates that European distance among the continuous feature representation of $v_4$ and $v_1$, $v_2$ should be relatively small. Previous session-based recommendation systems only focus on the item relations in a single session but ignore the complex item inter-dependencies in different sessions. In order to obtain the low-dimensional vector representation of the items in graph $\mathcal{G}$ and maintain the homogeneity of items, we utilize a cross-session item dependency encoder to generate item embedding. The output is the vector representation of the items on the graph, i.e., $\mathcal{V} = \{v_1, v_2, ..., v_n\}$, where $v_i \in \mathbb{R}^d$, $d$ is the dimensionality of latent item representations.

In order to distinguish the importance of the different adjacent items, we assign the weight of each edge according to the number of the occurrence in all sessions. After constructing graph $\mathcal{G}$, the item’s corpus is generated by truncating random walk on the graph, and then we train a skip-gram model on the corpus. The random walk traverses all items and generates the context of each item. To fully exploit the contextual signals of item relationships on the graph, we generating the context, it will sample a node from the neighborhood of the current node according to the weight of edge until the length equals to $L$ with the number of walks as $T$ (both parameters are studied in Section 3). After conducting the random walk process, we obtain a plurality of item sequences, as well as the corpus of items $\mathcal{C} = \{c_1, c_2, ..., c_{|\mathcal{C}|}\}$. Then, the skip-gram model [27], which maximizes the co-occurrence probability of two words that appear simultaneously in a window, is utilized to model the item co-occurrence on the corpus $\mathcal{C}$:

$$\text{maximize} \prod_{v_i \in \mathcal{C}} \prod_{v_j \in \text{context}(v_i)} P(v_j|v_i)$$  \hspace{1cm} (1)$$

where $\text{context}(v_i)$ denotes the context items of item $v_i$ in sequence $c$ in a window size $N_w$. The conditional probability $P(v_j|v_i)$, which denotes how likely $v_i$ is observed in the contexts of $v_j$, is computed by the inner product kernel with softmax for output:

$$P(v_j|v_i) = \frac{\exp(v_i^T \theta_j)}{\sum_{k=1}^{|\mathcal{V}|} \exp(v_i^T \theta_k)}$$ \hspace{1cm} (2)$$

where $\theta$ is the learned item embedding from the cross-session item dependency encoder.
The corresponded loss function in our cross-session dependency encoder is defined as follows:

$$L_g = \sum_{v_i \in C} \sum_{v_c \in C} \log \frac{\exp(v_i^T \theta_c)}{\sum_{k=1}^{w_i} \exp(v_i^T \theta_k)}$$  \hspace{1cm} (3)$$

where $v_i$ is the vector representation of $v_i$, $\theta_c \in \mathbb{R}^d$ denotes the role of $v_c$ as a context. Nevertheless, minimizing $L_g$ is non-trivial because the denominator term in (3) is very time-consuming. Negative sampling is an effective strategy to optimize the training complexity. The idea of negative sampling is to approximate the costly denominator in (3) with some sampled negative instance. NCELoss [10] applies a binary classifier to discriminate the target item and the sampled negative items. The conditional probability $P(v_c|v_i)$ is computed by (4):

$$P(v_c|v_i) = \begin{cases} \sigma(v_i^T \theta_c) & \text{if } v_c \in C(v_i) \\ 1 - \sigma(v_i^T \theta_c) & \text{if } v_c \in N(v_i) \end{cases}$$  \hspace{1cm} (4)$$

Hence, the updated loss function is shown as below:

$$L_g = -\sum_{v_i \in C} \sum_{v_c \in C(v_i)} \log \sigma(v_i^T \theta_c) + \sum_{v_c \in N(v_i)} \log \sigma(-v_i^T \theta_c)$$  \hspace{1cm} (5)$$

where the $N_i(v_i)$ denotes the set of negative samples for current item $v_i$. $\sigma$ is the sigmoid function $1/(1 + e^{-x})$.

### 2.4 Hierarchical Attentive Aggregation Module

This aggregation network serves as the recommendation module in our LGSR framework, taking a single session $s = [v_{s,1}, v_{s,2}, ..., v_{s,t}]$ as input and outputting the relevance probability for all items. As shown in Figure 2, this module is equipped with two attention networks: (1) self-attention network: models the complex structures in sessions and captures the complicated transition between items, and (2) session aggregation network: capture the long-term preference and current interest of users.

#### 2.4.1 Self-Attention Network

We map the items in $s$ into a unified vector space via the item embedding layer which shared with the cross-session dependency encoder module and get $V_s = [v_{s,1}, v_{s,2}, ..., v_{s,t}]$. Since the self-attention model is not aware of the item positions in the session, we add a position embedding $P \in \mathbb{R}^{t \times d}$ into the item embedding, and get $E_s = [e_{s,1}, e_{s,2}, ..., e_{s,t}]$, where $e_{s,i} = v_{s,i} + p_i$. Motivated by [16], we utilize a learnable position embedding rather than a fixed position-aware vectors. Self-attention mechanism is a special case of the dot-product attention which calculates a weighted sum of all values($V$), where the weight relates to queries($Q$) and keys($K$) and defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$  \hspace{1cm} (6)$$

where $d_k$ is the dimension of $K$, and $\sqrt{d_k}$ denotes a scale factor to avoid overly large values of the inner product. When $Q, K, V$ are the same, the dot-product attention becomes so-called self-attention. In our case, they all equal to $E_s$. The self-attention (SA) is defined as:

$$S_s = \text{SA}(E_s) = \text{Attention}(E_s W_q, E_s W_k, E_s W_v)$$  \hspace{1cm} (7)$$

where $W_q, W_k, W_v \in \mathbb{R}^{d \times d}$ are the learnable parameters. Since self-attention is a linear operation, it is necessary to add a feedforward network to endow the model with nonlinearity. In this work, a two-layer feedforward network is applied to $S_s$:

$$F_s = \text{FFN}(S_s) = \text{ReLU}(S_s W^{(1)} + b^{(1)}) W^{(2)} + b^{(2)}$$  \hspace{1cm} (8)$$

where $\text{ReLU}(x) = \text{max}(0, x)$ is the activation function which aims to add nonlinearities to the model. $W^{(1)}, W^{(2)} \in \mathbb{R}^{d \times d}$ and $b^{(1)}, b^{(2)} \in \mathbb{R}^d$ are the learnable parameters.

In the self-attention layer, when calculating the attentive weight of the $i$-th item, it should only consider the first $(i-1)$-th items due to the nature of sequences. So, we forbid all links between $Q_i$ and $K_j$ for all index which $j$ is larger than $i$. Additionally, the multi-head attention, which jointly attend to information from different representation subspaces at different positions, can enhance the expression ability of self-attention. However, in our case, the experiment result shows that it isn’t as effective as expected. The main reason may be the value of $d$ is quite small in our case and it is no need to project them into the multiple learning subspace.

#### 2.4.2 Session Aggregation Layer

After self-attention, we obtain $F_s = [f_{s,1}, ..., f_{s,t}]$. Each $f_{s,i}$ adaptively extracts information from previous items. We apply another attention layer to generate session embedding by aggregating...
the learned sequential signals. The representation of current session
composed of two parts: long-term preference and the current inter-
est. We define the current interest \( s_i \), as \( f_{s,i} \), and employ attention
mechanisms on \( F_{s} \) to capture the long-term preference:

\[
\alpha_i = \text{softmax}(q^T \sigma(W_1 f_{s,i} + W_2 f_{s,i}))
\]  

(9)

where \( q, c \in \mathbb{R}^d \), and \( W_1, W_2 \in \mathbb{R}^{d \times d} \). The long-term pre-
ference \( s_t = \sum_{i=1}^{\alpha_i} f_{s,i} \), where the softmax function makes sure
that the sum of all weights equals to 1. The final session embedding
\( s_f = W_3[s_c; s_t] \) where \( W_3 \in \mathbb{R}^{d \times 2d} \) is the linear transforma-
tion of the concatenation of \( s_c \) and \( s_t \). Next, the scores of each candidate
item \( z_i = s_f^T v_i \), the inner product of the item embedding \( v_i \) and ses-

tion embedding \( s_f \). The probability of all candidate items to be next
clicked by the user in the current sessions is calculated by softmax
function.

\[
\hat{y} = \text{softmax}(z)
\]  

(10)

The loss function of the recommendation module with hierarchical
attentive aggregation network is defined based on the cross-entropy:

\[
L_{\text{rec}} = - \sum_{i} y_{i} \log(\hat{y}_{i}) + (1 - y_{i}) \log(1 - \hat{y}_{i})
\]  

(11)

where \( y_{i} \) denotes the label of \( i \)-th instance, which is the one-hot en-
coding vector of the ground truth.

2.5 Model Optimization

By integrating the introduce two key modules, we define our joint
loss function as follows:

\[
\mathcal{L} = L_{\text{rec}} + \lambda_1 L_{\theta} + \lambda_2 \|\Theta\|_2^2
\]  

(12)

\( \lambda_1 \) balances the loss from two tasks, \( \Theta \) is the parameter set of LGSR,
and the last term of (12) is the regularization term. \( \lambda_2 \) is another
balancing parameter for preventing over-fitting. Since the input of the
recommendation module and the cross-session dependency encoder
are different, so we employ mini-batch Adam [18] to optimize \( L_{\text{rec}} \)
and \( L_{\theta} \) alternatively. It is challenging to use \( \lambda_1 \) to adjust the weight
of two losses in alternatively optimizing. Inspired by [32], we use an
additional parameter \( g \), which denotes the training frequency of \( L_{\theta} \)
g in each epoch, to balance two losses.

3 Evaluation

We perform experiments on two real-world datasets to comprehen-
sively evaluate our proposed LGSR method. In particular, we aim to
answer the following research questions:

- **RQ1**: How does LGSR perform as compared to state-of-the-art
  session-based recommendation methods?
- **RQ2**: How is the performance of LGSR's variants with different
designed modules in the joint framework?
- **RQ3**: How do different hyperparameter settings affect the recom-
  mendation performance of LGSR?
- **RQ4**: How is the interpretation of our LGSR framework in captur-
ing dynamic correlation weights between items?

### 3.1 Experimental Settings

#### 3.1.1 Data Description

To validate the effectiveness of LGSR, we utilize two real-world
datasets from Diginetica\(^8\) and Yoochoose\(^9\). We summarize the sta-
tistical information of experimented datasets in Table 1 and present
data details as follows:

- **Diginetica Data**: This dataset contains the click records of users
over different items from an e-commerce service spanning the
time period of six months. For fair comparison, we follow the
same data preprocessing strategy as [20] and filter out the ses-
sions which include only one item (i.e., with the session length of
1). We also remove items whose frequency of appearance is less
than 5. There are 43097 items and 204771 sessions remaining in
the Diginetica dataset after preprocessing.

- **Yoochoose Data**: This data records users' clicked item logs from
another online retailing site. By performing the same preprocess-
ing steps as the Diginetica data, 37483 items and 798150 sessions
are included in the final Yoochoose data.

In our experiments, we split the data into training and test set in
chronological order. Considering different data scales of Diginetica
and Yoochoose, we follow the same experimental settings in [20, 35],
and construct the test set of Diginetica and Yoochoose data by
selecting sessions from the last week and last day, respectively. To
be consistent with the settings in [20, 23], we report the evaluation
results on the recent fractions with \( \frac{1}{nt} \) and \( \frac{1}{nt} \) of temporally ordered
session from the generated training sequences. We also present the
average session length of Diginetica and Yoochoose in Table 1.

#### 3.1.2 Methods for Comparison

In our experiments, LGSR is evaluated against the following vari-
ous state-of-the-art baselines: (i) popularity-based recommendation
strategy (i.e., POP and S-POP); (ii) K-nearest neighbor modeling al-
gorithm (i.e., item-KNN); (iii) recurrent recommendation technique
(i.e., GRU4Rec); (iv) session-based recommendation with graph
neural network (i.e., SR-GNN); (v) attentive recommendation mod-
els (i.e., NARM and STAMP).

- **POP**: It makes recommendations based on the popularity of items.
  For all sessions, POP recommends the most frequent items from the
  historical clicked item logs.
- **S-POP**: It is another popularity-based recommendation strategy
  by recommending most popular items in the current session.
- **item-KNN**: This baseline leverages the K-Nearest Neighbors
  algorithm and uses the cosine similarity to estimate the correla-
tions between items.
- **GRU4Rec**: This session-based recommendation approach
  utilizes the recurrent neural network (i.e., GRU) to encode se-
quential transitional regularities of user preferences.

### Table 1. Statistics of Experimented Datasets

<table>
<thead>
<tr>
<th>Datasets</th>
<th>#train</th>
<th>#test</th>
<th>#item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yoochoose-1/64</td>
<td>369859</td>
<td>5917745</td>
<td>719470</td>
</tr>
<tr>
<td>Yoochoose-1/4</td>
<td>55898</td>
<td>55898</td>
<td>60855</td>
</tr>
<tr>
<td>Diginetica</td>
<td>17376</td>
<td>30444</td>
<td>43097</td>
</tr>
</tbody>
</table>

Average Length: 6.16 (Yoochoose-1/64), 5.71 (Yoochoose-1/4), 5.13 (Diginetica)

\(^8\) http://cikm2016.cs.iupui.edu/cikm-cup
3.2 Performance Comparison (RQ1)

We present the evaluation results of all compared methods on all datasets in Table 2 and summarize the following key observations:

- **SR-GNN** [35]: This method incorporates graph neural networks to model transitions between items of session sequences, and capture users’ current interests within the session.
- **NARM** [20]: It is an integrative recommendation model with the attention mechanism and recurrent neural network based on an encoder-decoder learning architecture.
- **STAMP** [23]: This method utilizes the multi-layer perceptron network and attention mechanism to extract users’ preferences from the long-term session contextual signals.

3.3 Model Ablation Study of LGSR (RQ2)

We also perform ablation experiments over the key components of LGSR so as to have a better understanding of their impacts, i.e., cross-session item dependency encoder and hierarchical attentive aggregation module. We design the following model variants corresponds to different perspectives:

- **Efficacy of cross-session item dependency encoder. LGSR-C**: In order to investigate the impact of incorporating the implicit item relations across different session units, we design this model variant (i.e., without the dependency encoder between sessions) to capture dynamic user preferences via pure attention networks to model item transitional relationships.
- **Effectiveness of attentive aggregation layer. LGSR-A**: another variant of LGSR which makes recommendations only by performing the self-attentive operation over the time-ordered item sequences. We regard the learned latent representations from the last the best performance, it still generates very competitive results. We attribute the performance improvement to the reason that the proposed LGSR jointly considers local and global item inter-dependencies, which can help to model intra- and inter-session transitional regularities of user’s dynamic preference simultaneously for more accurate recommendation results.

3.1.3 Parameters Settings and Reproductivity

We implement our LGSR with TensorFlow. The regulation penalty term is set as $\lambda_2 = 10^{-6}$. During the learning process, we perform the parameter inference using the Adam optimizer with the batch size and learning rate as 512 and $10^{-3}$, respectively. We set the training frequency $L_0$ in each epoch as 2. The path length $L$ and the walks per node $T$ are set to 50 and 13, respectively. We further set the window size $N_w$ to 8 and the number of the negative samples $N_n$ to 512. In addition, we apply the dropout layers with the dropout rate as 30% to alleviate the overfitting issue during the training phase. To make our results fully reproducible, all the relevant source codes have been made public at https://github.com/chenjh1988/LGSR.

### Table 2. Performance Comparison on Yoochoose-1/64, Yoochoose-1/4 and Diginetica.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Yoochoose-1/64</th>
<th>Yoochoose-1/4</th>
<th>Diginetica</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P@20(%) MRR@20(%)</td>
<td>P@20(%) MRR@20(%)</td>
<td>P@20(%) MRR@20(%)</td>
</tr>
<tr>
<td>POP</td>
<td>6.71 1.65</td>
<td>1.33 0.30</td>
<td>0.89 0.20</td>
</tr>
<tr>
<td>S-POP</td>
<td>30.44 18.35</td>
<td>27.08 17.75</td>
<td>21.06 13.68</td>
</tr>
<tr>
<td>Item-KNN</td>
<td>51.60 21.81</td>
<td>52.31 21.70</td>
<td>35.75 11.57</td>
</tr>
<tr>
<td>GRU4Rec</td>
<td>60.64 22.89</td>
<td>59.53 22.60</td>
<td>29.45 8.33</td>
</tr>
<tr>
<td>NARM</td>
<td>68.32 28.63</td>
<td>69.73 29.23</td>
<td>49.70 16.17</td>
</tr>
<tr>
<td>STAMP</td>
<td>68.74 29.67</td>
<td>70.44 30.00</td>
<td>45.64 14.32</td>
</tr>
<tr>
<td>SR-GNN</td>
<td>70.57 30.94</td>
<td>71.36 31.89</td>
<td>50.73 17.59</td>
</tr>
<tr>
<td>LGSR</td>
<td><strong>71.97 31.29</strong></td>
<td><strong>72.23 31.39</strong></td>
<td><strong>53.77 18.88</strong></td>
</tr>
</tbody>
</table>

Note that larger P@20 and MRR@20 values indicates better recommendation performance.

3.4 Evaluation Metrics

We use the following metrics that are widely used in the session-based recommendation [20, 23] to evaluate all compared methods.

- **P@20** (Precision): it represents the proportion of correctly recommended items in the top-20 items among all test instances.
- **MRR@20** (Mean Reciprocal Rank): it takes average on the reciprocal ranks of users’ desired items. We set the reciprocal rank as 0 when the desired item is not among the top-20 recommended items. This metric measures the position of the top relevant recommendations.

Note that larger P@20 and MRR@20 values indicates better recommendation performance.

3.2 Performance Comparison (RQ1)

We present the evaluation results of all compared methods on all datasets in Table 2 and summarize the following key observations:

- In general, we can observe that LGSR consistently yields the best performance in terms of precision and mean reciprocal rank in most evaluation cases. In the occasional cases that LGSR misses
Hyperparameter Study of LGSR (RQ3)

We now study how the different hyperparameter settings affect the recommendation performance. The key parameters involve the hidden state dimensionality \( d \), training frequency \( g \), walks per item \( T \), path length \( L \) and the number of negative samples \( N_s \). Except for the parameter being tested, we set other parameters at the default values we described before.

Impact of hidden state dimensionality \( d \). We vary the value of \( d \) from 60 to 140 to study how the dimension of item embedding affects the model performance. The results on both Yoochoose-1/64 and Yoochoose-1/4 in terms of P@20 are shown in Figure 4. We can observe that the larger hidden state dimensionality brings better representation capability for item relations at the earlier stage, and the performance tends to saturate as \( d \) reaches 100. In our experiments, \( d \) is set to 100.

Impact of training frequency \( g \). We investigate the influence of the training frequency of \( L_g \) by varying \( g \) from 1 to 4. Figure 4 indicates that a higher \( g \) value will mislead the objective function of LGSR and take more training time, while a small \( g \) can make full use of the global information of item dependencies. Hence, we set \( g = 2 \) to obtain better performance.

Impact of walks \( T \) and path length \( L \) per item. The truncated random walk generates \( T \) walks with length of \( L \) for each item. We vary \( L \) and \( T \) to investigate their effect in our cross-session item dependency encoder of LGSR. From Figure 5, we can observe the similar trend of these two parameters, i.e., the performance first increases and then remains stable. The larger value of \( L \) and \( T \) indicates that the cross-session item correlations are more fully excavated. However, when \( L \) and \( T \) is large, the model may overstate the role of global relation signals while reducing the importance of intra-session item transitions.

Impact of window size \( N_w \). The window size \( N_w \) of the skip-gram determines the number of co-occurrence item pairs to be considered in our cross-session dependency encoder, i.e., the larger the window size is, the more items pairs will be optimized in training process. We vary \( N_w \) from 3 to 9 and show the evaluation results in Figure 6. We can notice that the larger window size first improves the model performance but hurts the recommendation accuracy later.

Impact of the number of negative samples \( N_s \). An appropriate sampling strategy and size can accelerate the training phase while maintaining satisfactory results. Hence, another key hyperparameter in our LGSR is the number of the negative samples \( N_s \). We evaluate the performance and time cost of different sample sizes. As \( N_s \) increases, the performance enhances and the more computationally training time is required. However, when \( N_s \) further increases (i.e.,...
weights of case 2

Figure 6. Impact of Nw and Ns on Yoochoose-1/64

≤ 512), the performance becomes relatively stable while the time cost still increases. Therefore, we set Ns = 512 by considering the trade-off between model accuracy and computational cost.

Figure 7. Visualization of self-attention weights in two modeled sessions. The depth of the color corresponds to the importance scores of items.

3.5 Model Interpretation Study (RQ4)

Apart from the superior forecasting performance, another key advantage of LGSR is its ability in interpreting the importance weights of item correlations. To demonstrate this, we perform case studies to show the interpretability of our model by visualizing the attention weights obtained from LGSR. Particularly, we visualize both the self-attention weights in sequential modeling and the attention weights in long-term preference capturing of extracted samples on Yoochoose-1/64 data, to illustrate the explainability of attention mechanism intuitively. Figure 7 shows two heatmaps of self-attention weights of two samples sessions with 15 items. There are few previous items related to current item (deeper color) in most cases. This may owe to the interest transfer of users. As showed in Figure 8, we present some attention weights when calculating the long-term preference of the user in the current session. Overall, a few consecutive items are related to the next click in current session and the most important items often appear in the end of the session. Hence, we could observe that LGSR enables the dynamic modeling of correlations between the target item and other relevant items.

Figure 8. Visualization of the attention weights in capturing users’ long-term preference. The depth of the color corresponds to the importance scores of items. The numbers indicate case indexes.

4 Related Work

4.1 Session-based Recommendation

Conventional recommendation techniques. The primary purpose of the session-based recommendation is to make predictions on users’ interests based on anonymous user’s behavior sequences (e.g., clicked item sequences) [13, 9]. Without the availability of user’s profile information results in the failure of traditional collaborative filtering approaches [19]. There exists conventional frequency-based methods are developed to study the session-based recommendation problem, such as neighborhood search-based methods [29] and Markov chain-based methods [28]. However, the aforementioned conventional recommendation techniques are difficult to be adapted to capture the time-evolving user’s preferences and are expected to perform poorly when the user’s online behavior is highly dynamic.

Recurrent recommendation methods. Recurrent neural networks (RNNs), which is specifically designed for sequence modeling (e.g., machine translation [3] and image caption [25]), have received a great amount of attention due to their capability in modeling nonlinear sequential correlations [24, 6]. In session-based recommendation scenarios, RNN-based methods have been proposed to explore sequential patterns of user behavior [12]. However, RNN-based method is designed for modeling item sequential transitions from single view, which goes against the hierarchical item inter-dependencies and may not fit the true distributions of user behavior data.

Attention-based learning models. Based on the architecture of recurrent neural networks, attention-based neural models have been successfully used in session-based recommendation tasks [20, 23]. For example, NARM [20] regarded the last hidden state as user behavior representation, and weighted combined all hidden states as the main purpose of the user in the current session. While attention mechanism has addressed the limitation of RNN without the fixed length internal representation [17, 31], these methods aimed to assign weights to intra-session item relations. Different from those models, our LGSR framework jointly learns intra- and inter-session item dependencies in a fully automatic manner.

4.2 Multi-Task Learning

Multi-task learning has been applied to enhance performance of many learning tasks [1], such as sequence modeling [21, 22] and multi-modal behavior modeling [7]. Recent work has demonstrated the effectiveness of multi-task learning in recommendations. For example, PACE [38] and BiNE [8] enhance personalized recommendation by user-item bipartite graph node embedding. KGAT [33] employs the graph attention network to perform representation learning on the knowledge graph and simultaneously makes recommendation. Motivated by the insights from these work, we design a multi-task learning framework LGSR for the session-based recommendation, by simultaneously performs global item relation structure learning and local dynamic item transition modeling.

5 Conclusion

In this work, we explore both the local and global user behavior dynamics for session-based recommendations. We devise a new framework LGSR, which explicitly models the intra- and inter-session item transition signals. At its core is the integration of cross-session dependency encoder and a dual-stage attentive aggregation network, which learns effective item representations—preserving the item relation heterogeneity. Extensive experiments on two real-world datasets demonstrate the rationality and effectiveness of LGSR.
REFERENCES


[41] Carl Yang, Lanxiao Bai, Chao Zhang, Quan Yuan, and Jiawei Han, ‘Bridge collaborative filtering and semi-supervised learning: a neural approach for poi recommendation’, in Proc. of the 23rd ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining, pp. 1245–1254. ACM, (2017).