Neural Computing and Applications
https://doi.org/10.1007/s00521-022-06966-3

ORIGINAL ARTICLE

Session-based recommendation with an importance extraction module

Zhiqiang Pan[1] · Fei Cai[1] · Wanyu Chen[1] · Honghui Chen[1]

Received: 11 January 2021 / Accepted: 14 January 2022

Abstract
The goal of session-based recommendation is to make item predictions at the next timestamp based on the anonymous ongoing session. Previous work mainly models user's preference by exploring the transition pattern between the interacted items in the session. However, they generally fail to pay enough attention to the item importance in terms of the relevance of the items to user's main purpose. This paper proposes a Session-based Recommendation approach with an Importance Extraction Module, i.e., **SR-IEM** $_{\text{improved}}$, which can simultaneously consider user's long-term interactions and recent behavior in current session. Specifically, we modify the self-attention mechanism to avoid introducing bias for estimating the importance of each item in the session. Then, the item importances are utilized to produce user's long-term preference, and the sequential signals are incorporated in the long-term interest modeling. Next, the long-term preference and user's current interest which is conveyed by the last interacted item in the session are combined to obtain user's final preference representation. Finally, item predictions are generated using the user preference, where a normalization layer is adopted to solve the long-tail problem. Extensive experiments are conducted on three public benchmark datasets, i.e., *Yoochoose 1/64*, *Yoochoose 1/4* and *Diginetica*. The experimental results show that SR-IEM$_{\text{improved}}$ can outperform the start-of-the-art baselines in terms of Recall and MRR for session-based recommendation. In addition, compared to the state-of-the-art neural methods, SR-IEM$_{\text{improved}}$ can obviously reduce the computational complexity.

Keywords Recommender systems · Session-based recommendation · Importance extraction module · Self-attention mechanism

1 Introduction

Recommender systems can help people obtain personalized information from the explosively increasing sources on the internet [14, 35, 36, 39]. Most existing recommendation methods recommend future items for user to interact with by relying on user's general preference, which is modeled from the historical interactions with items [21, 29]. However, in scenarios where historical user-item interactions are unavailable, such as the user is newly registered, it is challenging to accurately detect user's intent from the limited items [7]. Thus, the session-based recommendation task has been proposed to make item predictions merely based on the ongoing session.

Actually, Recurrent Neural Networks (RNNs) [7, 15], the attention mechanisms [15], and Graph Neural Networks (GNNs) [32] have been widely investigated in session-based recommendation. First, sequential methods play important roles in session-based recommendation. For

Zhiqiang Pan and Wanyu Chen have contributed equally to this work.

A preliminary version of this paper has appeared in the proceedings of SIGIR 2020 [19]. Compared to the conference version, we (1) propose a new model **SR-IEM** $_{\text{improved}}$ which extends our previous work on accurately obtaining user's preference by utilizing the layer normalization to solve the long-tail problem and adding the position embeddings to incorporate the sequential signal; (2) include a large scale dataset *Yoochoose 1/4* to conduct additional experiments for verifying the effectiveness of our model; (3) conduct an ablation study for validating the utility of each module in the recommendation approach; (4) give a case study to show how the IEM module works on distinguishing the items according to their corresponding importances; and (5) include more related work and present the analysis of our proposal as well as the experimental results in detail.

Extended author information available on the last page of the article instance, Hidasi et al. first propose to model user's sequential behaviors by a Gated Recurrent Unit (GRU) for capturing user's instant and dynamic preference in the session [7]. Li et al. propose Neural Attentive Recommendation Machine (NARM) which utilizes the attention mechanism to capture user's main purpose for making recommendations [15]. On top of NARM, Wang et al. exploit the collaborative information involved in the introduced neighbor sessions for better user intent detection [28]. Second, attention based methods focus on the relative importance of all items in the session so as to distinguish them. For example, Liu et al. propose to estimate user's general preference and current interest by utilizing the long- and short-term memory models, respectively [17]. Additionally, the complex transition relationship between items is taken into consideration. For instance, Wu et al. employ the Gated Graph Neural Networks (GGNNs) to generate accurate representation of the items, which are then aggregated with an attention mechanism for producing item predictions [32].

Even though the aforementioned methods have achieved satisfactory performance, the following issues are still unsolved. For instance, they generally fail to pay enough attention to the importance of each item in the ongoing session. This means that it is difficult to accurately locate the important items and eliminate the unrelated items for detecting user's main intent. After the item embeddings are generated, the existing methods distinguish the importance of items by simply relying on each item's relevance either to the mixture of items in the long-term interactions [15, 28] or the last single item [32] or a combination [17]. Unavoidably, non-relevant items in the session will mislead the recommendation methods to accurately detect user's main purpose. In addition, the RNN or GNN based models like NARM [15], CSRM [28], SR-GNN [32] have an unsatisfying high computational complexity because of the contained complex operations.

Clearly, during a user's interaction process with items, the user will mainly focus on the purpose within the session, such as purchasing some certain goods or browsing a certain webpage. However, because of the uncertainty of user's behavior pattern, the user may occasionally interact with some unrelated items out of curiosity or by accident [15]. Considering an example session in Fig. 1, a user who tends to purchase a phone clicked several phones to look for a product for maintaining his requirements. However, after clicking the *iPhone 11*, the user accidently clicked *Airpods* as they are both *Apple* products and are usually used together. Nevertheless, the *Airpods* is obviously uncorrelated to user's main purpose (i.e., purchasing a phone). Such complex behavior pattern makes the methods which refer to the sequential order [15] or pairwise item-transitions [32] easily introduce bias because of the unrelated items.

We propose an approach for *session-based recommendation with an importance extraction module*, i.e., **SR-IEM** $_{\text{improved}}$, which can effectively capture user's long-term preference and current interest simultaneously. More specifically, to obtain user's long-term preference, we introduce an Importance Extraction Module (IEM) that modifies the self-attention mechanism for estimating the importance of each item in the session. Then, the item embeddings and the position embeddings are merged to generate the final item representations, which are discriminatively combined as user's general preference according to the generated item importance scores. Moreover, to maintain user's instant needs, we resort to the embeddings of the last item in the ongoing session as an expression of user's current interest. Finally, we combine the long-term and recent interests by concatenation as the final user preference for producing item recommendations, where the normalization layer is utilized to deal with the long-tail problem.

Our main contributions can be summarized as follows:

1. We propose an Importance Extraction Module (IEM) to accurately estimate the importance of each item and avoid introducing bias for session-based recommendation. The proposed SR-IEM$_{\text{improved}}$ can simultaneously capture user's long-term preference and current interest for making item predictions.
2. We incorporate the sequential information in the session by considering the position embeddings and introduce a normalization layer to solve the long-tail problem in session-based recommendation.
3. Extensive experiments conducted on three benchmark datasets, i.e., *Yoochoose 1/64*, *Yoochoose 1/4* and *Diginetica*, indicate that SR-IEM$_{\text{improved}}$ can outperform the competitive baselines in terms of Recall and MRR. Moreover, the improvement on boosting the ranking of the target item is especially obvious in scenarios with relatively less training data. In addition, SR-IEM$_{\text{improved}}$ has a lower computational complexity than the state-of-the-art neural models.

We organize the rest of this paper as follows: We review some related work in Sect. 2 and present the details of our proposed SR-IEM$_{\text{improved}}$ model in Sect. 3. In Sect. 4, we give the setting of experiments, and then report the experimental results and conduct deep analysis in Sect. 5. Finally, we conclude our work and suggest the future directions of our research in Sect. 6.

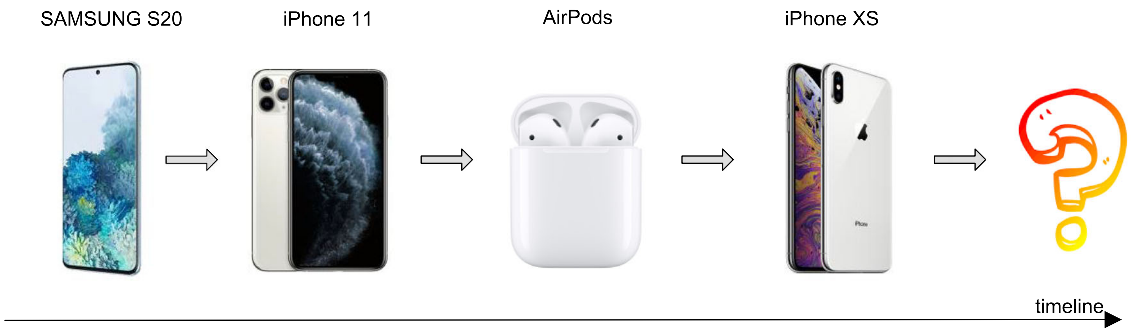


Fig. 1 An example session, where a user who aims to buy a phone interacted with an *Airpods* accidently during the interaction process

2 Related work

In this section, we review the related work about the general recommendation methods, the sequential recommendation methods, the graph neural networks based methods, and the attention based methods.

2.1 General recommendation methods

Most general recommenders are relying on Collaborative Filtering (CF) and can be mainly categorized into neighborhood-based models [16, 23] and latent factor-based methods [3, 6, 13].

Neighborhood-based models produce recommendations mainly based on the similarity between users or items, which correspond to the user-based and item-based KNN methods, respectively. The similarity can be computed by the Pearson correlation coefficient [23] or the cosine similarity. For instance, Jin et al. propose to automatically assign different weights for the items based on their ratings from users in the training set, keeping similar users close to each other [8]. Different from neighborhood-based models, latent factor-based methods focus on the user-item interactions. Specifically, latent factor-based methods first generate a latent factor for each user and item to represent them based on a given rating matrix. Then, the item predictions are generated upon the generated latent factors. Common traditional methods in this category include Singular Value Decomposition (SVD) [11], Matrix Factorization (MF) [12], etc. Moreover, He et al. first propose to extend matrix factorization in a nonlinear way by leveraging a multi-layer perceptron [6]. Furthermore, Chen et al. propose a joint training neural method which can learn deep features of users and items, and simultaneously explore deep interactions between them [3].

However, both neighborhood-based and latent factor-based methods generate recommendations based on user's long-term historical interactions, which are unavailable in the setting of session-based recommendation. In addition, general recommenders can merely detect user's inherent preference, failing to capture user's interest migration. Unlike the work listed above, by modeling the hybrid preference in the ongoing session, our proposal can accurately obtain user's instant interest, so as to generate recommendations to maintain user's current needs.

2.2 Sequential recommendation methods

Considering that items in a session are organized according to the temporal order, many methods focus on the sequential information in the session to obtain user's instant preference.

The traditional methods are proposed to model the sequential signal based on Markov Chains (MC). For instance, Shani et al. regard the generation process of the item recommendations as a sequential optimization problem and utilize Markov Decision Processes (MDPs) to generate item predictions [24]. Moreover, Rendle et al. consider both user's long-term preference and sequential behaviors by using matrix factorization and Markov chains, respectively [22]. Recently, many neural networks like RNNs and GRUs are widely applied to session-based recommendation. For instance, Hidasi et al. first propose GRU4REC, which utilizes GRU to model the sequential signal in the session and adopts a session-parallel mini-batch for training [7]. Furthermore, Li et al. propose NARM where an attention mechanism is utilized for emphasizing user's main purpose [15]. Moreover, considering limited information is contained in current session, similar sessions are incorporated as neighbors for enriching the current session. For instance, Wang et al. propose to incorporate neighbor sessions via the memory networks [31] for helping model the current session on the basis of NARM [28].

However, Markov Chains based methods can merely utilize the information from adjacent items, while neglecting the whole sequence. Regarding RNN based methods, it has been proved that the sequential signal cannot fully represent the transition relationship between items since user's interaction behavior is highly complicated [20, 32]. Different from the sequential methods which rely on the chronological order of items in session, our proposed SR-IEM$_{\text{improved}}$ method focuses on the relative importance of each item. This can avoid introducing bias brought by the accidentally interacted items in the item sequence, so as to concentrate on user's main intent in the current session.

2.3 Graph neural networks based methods

Considering the ability of modeling complex transition relationship between objects, Graph Neural Networks (GNNs) are widely adopted in the field of recommender systems.

For instance, Wang et al. design NGCF to propagate embeddings between users and items to explicitly encode the collaborative signal, so as to exploit the high-order connectivities between them [29]. In addition, He et al. simplify NGCF by removing feature transformation and nonlinear activation in NGCF to merely preserve the neighborhood aggregation component in GCN [10] to generate item recommendations [5]. GNNs have also been widely adopted for modeling the complex transition pattern among items in the session-based recommendation task. For instance, Wu et al. first propose to transform each session into a session graph and then apply the Gated Graph Neural Networks (GGNN) to generate accurate item embeddings [32]. After that, the session representation is obtained using an attention mechanism. Qiu et al. design the Full Graph Neural Networks (FGNN), where the graph attention networks (GAT) is utilized for modeling the transition relationship between items. Then, a Readout function [26] is used to generate the session representation [20]. Moreover, Wang et al. propose to simultaneously exploit the session graph and global graph level item transitions to learn the corresponding contextual information. Then, a reversed position aware attention is utilized to incorporate the sequential information for session aggregation [30].

However, GNN based methods generally have a high computational complexity. Moreover, the session aggregation in GNN based methods mostly relies on the last item [32, 33] or combination of all items in current session [20, 30], which may introduce much bias since the session may include unrelated items. Differently, by comparing each item with other items in current session in the importance extraction module, our proposal can accurately generate the importance score of each item. In addition, the low computational complexity of the attention mechanism makes our proposal much efficient in both training and test.

2.4 Attention based methods

Since user's interacted items have different degree of importance and contribute variously to modeling user's preference [17, 19], many approaches rely on an attention mechanism to generate an accurate user representation.

For instance, Liu et al. adopt the attention mechanism to generate user's general preference as well as the recent interest by relying on the long- and short-term memories in the ongoing session, respectively [17]. Moreover, Kang and McAuley propose a self-attention based method (SASRec) to capture long-term semantics in user's historical interactions for making item predictions in sequential recommendation [9]. In addition, for users whose historical interactions are available, Ying et al. propose to consider both the general and current interests using a two-layer hierarchical attention network [34]. The first attention layer is applied to user's historical interactions to generate user's long-term interest, while the second layer is used to fuse the long-term preference with the items in the short-term interaction set. Furthermore, Chen et al. propose to utilize a co-attention network to take the dynamic connection between user's long-term and recent interactions into consideration for making item predictions [2].

In general, the attention scores generated by the aforementioned methods are mainly produced based on the last item [32] or the hybrid of all items within the session [20, 30] or their combination [17]. However, both the last item or items in the session could be unrelated to user's real intent because of user's uncertain interaction behavior. The inaccurate attention scores will make the user preference unable to be well represented. On top of the previous attention based models, we improve the evaluation of the item importance in the current session using the IEM component. The designed IEM can effectively distinguish the relevance of different items to user's main intent, so as to eliminate the unrelated items.

Moreover, in our previous work [19], we propose to accurately estimate the importance of each item via the proposed importance extraction module (IEM) to avoid introducing bias brought by the unrelated items. In this extension, we additionally incorporate the position embeddings to take the sequential signal in the session into consideration, and we utilize a normalization layer to solve the serious long-tail problem in session-based recommendation.

3 Approach

In this section, we first present the formalization of the session-based recommendation task. Then, we detail our proposed SR-IEM$_{\text{improved}}$ model.

The task of session-based recommendation aims to predict the item for user to click at the next timestamp, which is formulized as follows. Assuming a set of $|V|$ items is represented by $V = \{v_1, v_2, \ldots, v_{|V|}\}$, where $v_i \in V$ is an item. Given a session $S = \{x_1, x_2, \ldots, x_t\}$ consisting of t items that a user clicked, the goal of session-based recommendation is to predict the next item to recommend at timestamp $t+1$. Specifically, we obtain the user preference from the current session and utilize it to make item predictions to obtain the probability of recommending all candidate items $\hat{\mathbf{y}} = \{\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_{|V|}\}$, where $\hat{\mathbf{y}}_i \in \hat{\mathbf{y}}$ indicates the probability of item v_i being clicked by the user. As last, we select the items with the top-K highest scores to recommend to the user.

Figure 2 presents an overview of SR-IEM$_{\text{improved}}$. First, each item x_i in the session is mapped into a d-dimensional embedding vector $\mathbf{x}_i$ to represent the item. Then, we transform the item embeddings into *query* and *key* in the importance extraction module, which are utilized to obtain the attention score of each item. After that, we obtain the long-term preference in the session by combining the item representations according to the attention scores, where the item representations are generated by adding the item embeddings to their respective position embeddings. Then, we concatenate the long-term preference and the recent interest (represented by the last item in the session) to obtain the final user preference. Finally, we apply a normalization layer to deal with the long-tail problem and make item predictions to generate the recommendations. We adopt the cross-entropy as the loss function to train the proposed SR-IEM$_{\text{improved}}$ model.

3.1 Importance extraction

To focus on the important items and filter unrelated items in the session for user intent detection, we propose an Importance Extraction Module (IEM) to generate the item importance, so as to accurately locate the important items.

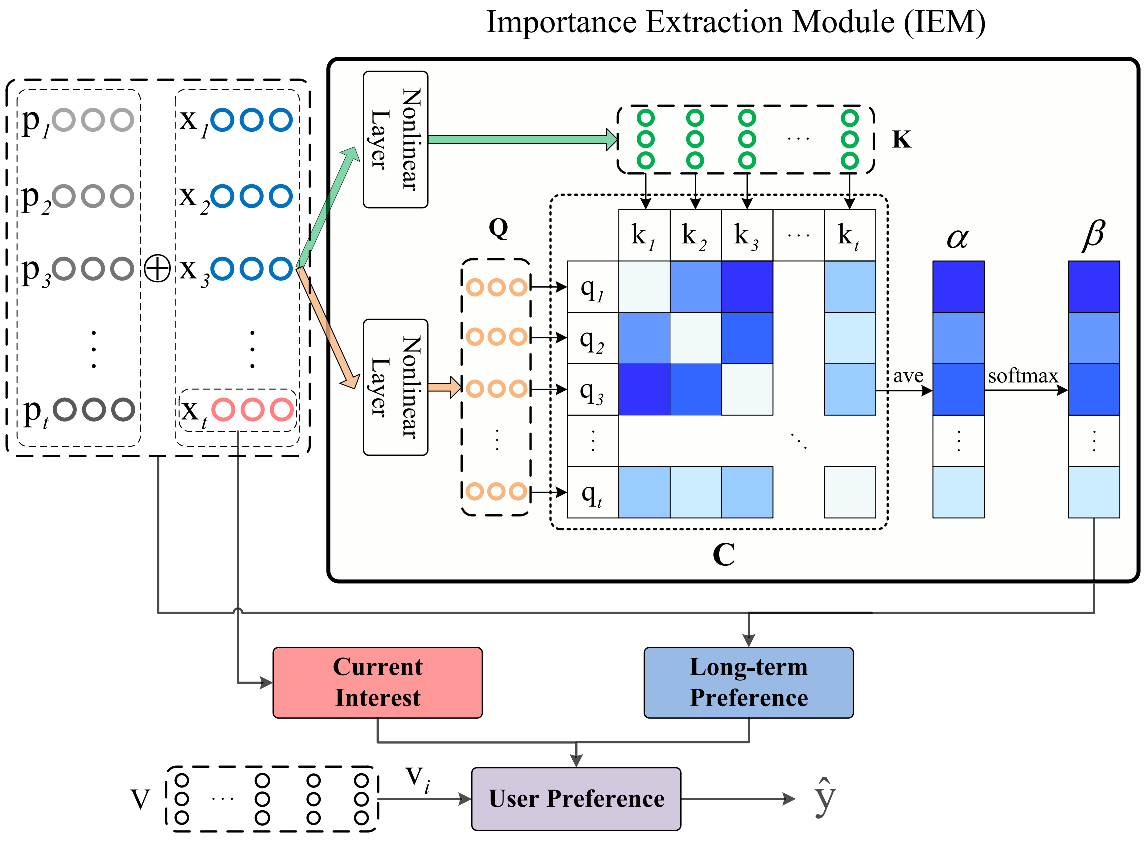


Fig. 2 The framework of SR-IEM$_{\text{improved}}$, which consists of three main components, i.e., the importance extraction module (IEM), the preference fusion module and the item recommendation module. Given a session, we first generate the importance scores of the contained items using IEM to obtain user's long-term preference. Then, we combine user's long-term and recent interests as the hybrid preference. Finally, we produce the prediction scores on all candidate items to make recommendations

First, we apply an embedding layer to embed each item x_i in $S = \{x_1, x_2, \ldots, x_t\}$ into a d dimensional representation $\mathbf{x}_i \in \mathbb{R}^d$. Then, borrowing the merits from self-attention mechanism [25], we transform the item embeddings $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t\}$ into different spaces via the nonlinear functions to generate *query* $\mathbf{Q}$ and *key* $\mathbf{K}$, respectively.

$$\mathbf{Q} = \mathrm{sigmoid}(\mathbf{W}_q\mathbf{X}), \tag{1}$$

$$\mathbf{K} = \mathrm{sigmoid}(\mathbf{W}_k\mathbf{X}), \tag{2}$$

where $\mathbf{W}_q \in \mathbb{R}^{l\times d}$ and $\mathbf{W}_k \in \mathbb{R}^{l\times d}$ are the respective learnable parameters for *query* and *key*, l denotes the dimension of the item representation in the attention mechanism, and the sigmoid function is utilized to learn features from *query* and *key* nonlinearly [4, 6, 37].

After obtaining *query* $\mathbf{Q}$ and *key* $\mathbf{K}$, we modify the self-attention mechanism to estimate the importance of each item. Specifically, we define the affinity matrix $\mathbf{C}$ between $\mathbf{Q}$ and $\mathbf{K}$ by calculating the similarity of every pair of two items as:

$$\mathbf{C} = \frac{\mathrm{sigmoid}(\mathbf{Q}\mathbf{K}^{\mathsf{T}})}{\sqrt{d}}, \tag{3}$$

where $\sqrt{d}$ is utilized for scaling the attention scores.

We would like to utilize the generated affinity matrix $\mathbf{C}$ to accurately estimate the item importance. Specifically, if the similarity scores of item x_i with the other items in the ongoing session, i.e., the scores in the i-row of $\mathbf{C}$, are generally large, we can conclude that this item conforms to user's main intent within the session. In contrast, if an item is not similar to the other items in the ongoing session, this item may be an unrelated item that user interacted with by accident or out of curiosity. Inspired by the aforementioned intuition, we define the importance of each item as the average similarity score between the item and the other items in the session. In addition, to avoid high matching scores between identical items in *query* and *key*, a masking operation is employed to mask the diagonal of the affinity matrix as in [38]. Then, we can compute the *importance score* α_i of item x_i in the session as:

$$\alpha_i = \frac{1}{t-1}\sum_{j=1, j\neq i}^{t} \mathbf{C}_{ij}, \tag{4}$$

where $\mathbf{C}_{ij} \in \mathbf{C}$. Then, we utilize a softmax layer to normalize the scores so as to get the final *importance* β_i of item i as:

$$\beta_i = \frac{\exp(\alpha_i)}{\sum_i \exp(\alpha_i)}, \quad \forall i = 1, 2, \ldots, t. \tag{5}$$

3.2 Preference fusion

Through the importance extraction module, the importance of each item is generated, which denotes the relevance of each item to user's main intent in the ongoing session.

Moreover, considering that the item embeddings $\mathbf{X}$ do not include the sequential information in the session. Here, we add the position embeddings upon the item embeddings to incorporate the sequential signal:

$$\mathbf{X}^p = \mathbf{X} + \mathbf{P}, \tag{6}$$

where $\mathbf{P} \in \mathbb{R}^{t\times d}$ are the corresponding position embeddings.

Then, we combine all item representation $\mathbf{x}_i^p \in \mathbf{X}^p$ according to their importance scores as user's long-term preference $\mathbf{z}_l$ as:

$$\mathbf{z}_l = \sum_{i=1}^{t} \beta_i \mathbf{x}_i^p. \tag{7}$$

For the current interest, denoted as $\mathbf{z}_s$, we directly adopt the last item embeddings[1] by:

$$\mathbf{z}_s = \mathbf{x}_t. \tag{8}$$

Finally, user's preference are generated by combining the long-term preference $\mathbf{z}_l$ and the current interest $\mathbf{z}_s$ for producing item predictions:

$$\mathbf{z}_h = \mathbf{W}_0[\mathbf{z}_l; \mathbf{z}_s], \tag{9}$$

where $[\cdot]$ is the concatenating operation. $\mathbf{W}_0 \in \mathbb{R}^{d\times 2d}$ transforms the concatenated representation from a latent space $\mathbb{R}^{2d}$ into $\mathbb{R}^d$.

3.3 Item recommendation

After obtaining the user preference $\mathbf{z}_h$, we adopt it to produce item predictions by calculating the probability of clicking all candidate items in the item set V.

As session-based recommendation typically faces a serious long-tail problem, which means that items of low popularity are relatively less likely to be recommended. Here, we adopt a layer normalization to obtain the session representation $\tilde{\mathbf{z}}_h$ and candidate item embeddings $\tilde{\mathbf{v}}_i$ to solve this problem as:

$$\tilde{\mathbf{z}}_{\mathrm{h}} = \mathrm{LayerNorm}(\mathbf{z}_{\mathrm{h}}) = \frac{\mathbf{z}_{\mathrm{h}} - \mu_{\mathrm{z}}}{\sqrt{\sigma_{\mathrm{z}}^2 + \epsilon}}, \tag{10}$$

$$\tilde{\mathbf{v}}_i = \mathrm{LayerNorm}(\mathbf{v}_i) = \frac{\mathbf{v}_i - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \quad \forall i = 1, 2, \ldots, |V|, \tag{11}$$

[1] In [32], it has been proved that the last item can represent user's instant interest and contribute much to capturing user's main intent.

where σ_z and μ_z are the standard deviation and mean of the session representation, respectively. Similarly, σ_i and μ_i are the standard deviation and mean of the embeddings of item x_i, and ϵ is a small positive value used for preventing dividing zero.

Then, we utilize a multiplication operation to compute the preference score $\tilde{\mathbf{y}}_i$ on each item v_i as:

$$\tilde{\mathbf{y}}_i = \tilde{\mathbf{z}}_h^{\mathsf{T}} \tilde{\mathbf{v}}_i, \tag{12}$$

Note that after normalization, it will face a convergence problem since the softmax loss will be trapped at a very high value on training set [27]. To solve the problem, a scale coefficient γ is multiplied with the obtained scores in the softmax layer so as to generate the final score of each item as:

$$\hat{\mathbf{y}} = \frac{\exp(\gamma \tilde{\mathbf{y}}_{\mathbf{i}})}{\sum_i \exp(\gamma \tilde{\mathbf{y}}_{\mathbf{i}})}, \quad \forall i = 1, 2, \ldots, |V|. \tag{13}$$

where $\hat{\mathbf{y}} = (\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_{|V|})$. Finally, items with the highest scores constitute the recommendation list for the user.

We adopt the cross-entropy loss as the objective function to optimize the parameters in SR-IEM$_{\text{improved}}$ as:

$$L(\hat{\mathbf{y}}) = -\sum_{i=1}^{|V|} \mathbf{y}_i \log(\hat{\mathbf{y}}_i) + (1 - \mathbf{y}_i) \log(1 - \hat{\mathbf{y}}_i), \tag{14}$$

where $\mathbf{y}$ reflects the one-hot encoding vector of the ground truth. Finally, the Back-Propagation Through Time algorithm is applied to train our model.

We detail the training process of our SR-IEM$_{\text{improved}}$ model in Algorithm 1. We first initialize the parameters in SR-IEM$_{\text{improved}}$ in step 1. Then for each training session, we calculate the item importance scores using IEM in step 4. After that, we obtain the user preference from step 5 to 8, where the sequential signal is incorporated by step 5. From step 9 to 12, we normalize the user preference as well as the candidate item embeddings and then make the item predictions. Finally, we adopt the cross-entropy loss in Eq. (14) as the objective function and use back propagation to optimize the network parameters from step 14 and 15.

Algorithm 1 SR-IEM$_{improved}$

```
Input: Epochs: the number of training iterations
       $U$: all training sessions
Output: $\mathbf{V}$: embeddings of all items
       $\mathbf{P}$: position embeddings
       $\mathbf{W}_q$: transformation weights for query in IEM
       $\mathbf{W}_k$: transformation weights for key in IEM
       $\mathbf{W}_0$: transformation weights for preference fusion
 1: Randomly initialize $\mathbf{V}, \mathbf{P}, \mathbf{W}_q$, $\mathbf{W}_k$ and $\mathbf{W}_0$
 2: for epoch in range(Epochs) do
 3:     for session $S$ in $U$ do
 4:         $\beta_i = IEM(\mathbf{X}, \mathbf{W}_q, \mathbf{W}_k)$
 5:         $\mathbf{X}^p = \mathbf{X} + \mathbf{P}$
 6:         $\mathbf{z}_l = \sum_{i=1}^{t} \beta_i \mathbf{x}_i^p$
 7:         $\mathbf{z}_s = \mathbf{x}_t$
 8:         $\mathbf{z}_h = \mathbf{W}_0[\mathbf{z}_l; \mathbf{z}_s]$
 9:         $\tilde{\mathbf{z}}_h = LayerNorm(\mathbf{z}_h)$
10:         $\tilde{\mathbf{v}}_i = LayerNorm(\mathbf{v}_i)$
11:         $\tilde{\mathbf{y}}_i = \tilde{\mathbf{z}}_h^{\mathsf{T}} \tilde{\mathbf{v}}_i$
12:         $\hat{\mathbf{y}} = \frac{\exp(\gamma \tilde{\mathbf{y}}_i)}{\sum_i \exp(\gamma \tilde{\mathbf{y}}_i)}, \quad \forall i = 1, 2, \ldots, |V|$
13:     end for
14:     $L(\hat{\mathbf{y}}) = -\sum_{i=1}^{|V|} \mathbf{y}_i \log(\hat{\mathbf{y}}_i) + (1 - \mathbf{y}_i) \log(1 - \hat{\mathbf{y}}_i)$
15:     use back propagation to optimize the parameters;
16: end for
17: return $\mathbf{V}, \mathbf{P}, \mathbf{W}_q$, $\mathbf{W}_k$, and $\mathbf{W}_0$
```

4 Experiments

4.1 Research questions

To examine the performance of SR-IEM$_{\text{improved}}$, we address six research questions:

(RQ1) Can SR-IEM$_{\text{improved}}$ beat the state-of-the-art baselines for the session-based recommendation task?

(RQ2) Can SR-IEM$_{\text{improved}}$ achieve a lower computational complexity to reduce the training and test time compared with other neural methods?

(RQ3) How is the performance of SR-IEM$_{\text{improved}}$ on sessions of various length?

(RQ4) Can IEM perform better than other importance extraction methods on distinguishing the importance of items?

(RQ5) How is the contribution of each module in SR-IEM$_{\text{improved}}$ to the recommendation performance?

(RQ6) How does the IEM component of SR-IEM$_{\text{improved}}$ work in different cases?

4.2 Datasets

We utilize the benchmark datasets, i.e., *Yoochoose*[2] and *Diginetica*[3] for evaluation to validate the effectiveness of our proposal. *Yoochoose* contains the click streams within a 6-month period from an e-commerce website, which is obtained by the RecSys Challenge 2015. *Diginetica* comes from the transaction data released by CIKM Cup 2016.

Following [32], for *Yoochoose*, sessions of length 1 and items appearing less than 5 times are filtered out. Then, we split the sessions for training and test, where the sessions of the last day are adopted for test and the remaining part is used for training, respectively. Furthermore, items not included in the training set are removed. And as for *Diginetica*, the only difference is that the sessions of the last week are utilized for test. After preprocessing, 7,981,580 sessions with 37,483 items are remained in *Yoochoose*, and 204,771 sessions with 43,097 items constitute *Diginetica*.

As in [32], we adopt the sequence splitting operation to augment the training samples. More specifically, for session $S = \{x_1, x_2, \ldots, x_t\}$, we generate the sequences and corresponding labels as $([x_1], x_2), ([x_1, x_2], x_3), \ldots, ([x_1, \ldots, x_{t-1}], x_t)$ for both training and test. Moreover, since *Yoochoose* is too large, following [32], we only utilize the recent 1/64 and 1/4 fractions of the training sequences, denoted as *Yoochoose 1/64* and *Yoochoose 1/4*, respectively. Note that comparing to our previous work [19], we add experiments on *Yoochoose 1/4* to examine the robustness of SR-IEM$_{\text{improved}}$ on large scale datasets since *Yoochoose 1/4* is much larger than *Yoochoose 1/64* and *Diginetica*. Detailed statistics of three datasets are summarized in Table 1. In addition, we plot the distributions of sessions with different length on the training set of the three datasets in Fig. 3. As we can see, most of the sessions merely contain a few items, and the number of sessions drops rapidly with the session length increasing, especially in the *Yoochoose* datasets.

4.3 Model summary

The models discussed in this paper are listed as following. 1. Three traditional methods, i.e., S-POP [1], Item-KNN [23] and FPMC [22]; 2. Five neural models, i.e., GRU4REC [7], NARM [15], STAMP [17], CSRM [28] and SR-GNN [32]; and 3. Our proposals, i.e., our previous work SR-IEM$_{\text{basic}}$ [19] and the proposed model in this paper SR-IEM$_{\text{improved}}$.

- **S-POP** recommends the most frequent items of the ongoing session.
- **Item-KNN** recommends items similar to the current session based on the similarity between items in the current session and the candidate items.
- **FPMC** is a Markov Chains based hybrid method for sequential recommendation. Here, the user representation is omitted because it's unavailable in session-based recommendation.
- **GRU4REC** models the sequential information in session-based recommendation using a GRU.
- **NARM** employs GRUs to model the sequential signal and utilizes an attention mechanism to capture user's main purpose.
- **CSRM** incorporates the neighbor sessions as auxiliary information for the current session with a parallel memory module on the basis of NARM.
- **STAMP** obtains the general preference by an attention mechanism and combines it with the recent interest in the current session for recommendation.
- **SR-GNN** utilizes GGNN to generate the embeddings of items, which are aggregated to obtain the user representation using an attention mechanism.
- **SR-IEM** $_{\text{basic}}$ designs an importance extraction module to calculate the item importance for generating user's long-term preference, which is combined with user's recent interest to make recommendations.
- **SR-IEM** $_{\text{improved}}$ extends **SR-IEM** $_{\text{basic}}$ by utilizing the layer normalization to solve the long-tail problem and incorporating the position embeddings to model the sequential signal.

4.4 Experimental setup

The hyper parameters are selected on the validation set which is randomly selected from the training set with a proportion of 10%. Specifically, the dimension of the item embeddings (i.e., d) and the item representation in attention (i.e., l) are both searched in $\{50, 100, 150, 200, 250\}$ and then set as $d = 200$ and $l = 100$, respectively. Moreover, the scale coefficient γ is tuned in $\{8, 10, 12, 14, 16\}$ and then set to 12 on all three datasets. We adopt the $L2$ regularization to prevent overfitting which is ranged in $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ and finally set to $L2 = 10^{-5}$. Furthermore, we set the batch size to 128, and the Adam is adopted as the optimizer, where the initial learning rate is set to 0.001 and decays by 0.1 after every 3 epochs. In

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

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

Table 1 Statistics of the datasets used in our experiments

Statistics	*Yoochoose 1/64*	*Yoochoose 1/4*	*Diginetica*
# Cicks	557,248	8,326,407	982,961
# Training sessions	369,859	5,917,746	719,470
# Test sessions	55,898	55,898	60,858
# Items	16,766	29,618	43,097
Average session length	6.16	5.71	5.12

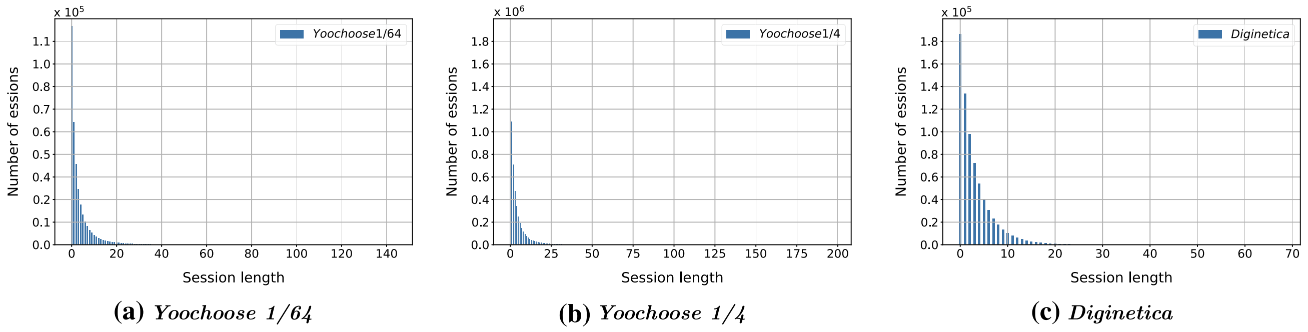


Fig. 3 Distribution of sessions with various length in the training set of three datasets

addition, the maximum session length is searched in $\{5, 10, 15, 20\}$ and then set to 10 as in [9]. This means that we only consider the 10 most recent items for long sessions. All parameters are initialized using a Gaussian distribution with a mean of 0 and a standard deviation of 0.1.

4.5 Evaluation metrics

Following previous work [32], **Recall@K** and **MRR@K** are adopted for evaluating the recommendation performance.

Recall@K measures whether the desired item occurs in the top-K positions in the ranking list.

$$\text{Recall@K} = \frac{n_{\text{hit}}}{N}, \tag{15}$$

where N and n_{hit} are the number of all test cases and the cases that the target item is amongst top-K in recommendation, respectively.

MRR@K is a normalized hit which considers the position of the target item. It is set to 0 if the target item ranks out of top-K positions in the recommendation list, and otherwise can be calculated as follows:

$$\text{MRR@K} = \frac{1}{N} \sum_{v_{\text{target}} \in S_{\text{test}}} \frac{1}{\text{Rank}(v_{\text{target}})}, \tag{16}$$

where v_{target} is the target item and $\text{Rank}(v_{\text{target}})$ is the ranked position of the target item.

5 Results and discussion

5.1 Overall performance

5.1.1 General comparison

To answer RQ1, we compare SR-IEM$_{\text{basic}}$ and SR-IEM$_{\text{improved}}$ to the baselines in terms of Recall@20 and MRR@20 on three datasets. The results are presented in Table 2. Here, the experimental results of the baselines are directly copied from [32] as all the experiments are conducted in the same setting except CSRM which we obtain the results by re-running its released code[4] on our used datasets.

As shown in Table 2, first of all, for the traditional baselines, we can observe that Item-KNN outperforms S-POP and FPMC on most cases in terms of Recall@20 and MRR@20 on three datasets. This indicates the utility of collaborative information from other items for promoting the recommendation accuracy for current session. Although FPMC takes the sequential signal into consideration, the performance is still poor, which may be due to that FPMC can merely consider the adjacent items, ignoring the overall session sequence.

In addition, from Table 2, we can see that the neural methods generally outperform the traditional models. Moreover, among the RNN based methods, i.e., GRU4REC, NARM and CSRM, we can observe that the emphasis of user's main purpose (comparing NARM to GRU4REC) and the incorporation of neighbor sessions

[4] https://github.com/wmeirui/CSRM_SIGIR2019.

Table 2 Model performance. The results of the best baseline and the best performer in each column are underlined and boldfaced, respectively

Method	*Yoochoose 1/64*		*Yoochoose 1/4*		*Diginetica*	
	Recall@20	MRR@20	Recall@20	MRR@20	Recall@20	MRR@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
FPMC	45.62	15.01	–	–	31.55	8.92
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
CSRM	69.85	29.71	70.63	29.48	51.69	16.92
SR-GNN	70.57	30.94	71.36	31.89	50.73	17.59
SR-IEM $_{\text{basic}}$	71.15	31.71	71.67	31.82	52.35	17.64
SR-IEM $_{\text{improved}}$	**71.95** $^{▲}_{△}$	**32.09** $^{▲}_{△}$	**72.61** $^{▲}_{△}$	**31.92**	**55.61** $^{▲}_{△}$	**18.93** $^{▲}_{△}$

Statistical significance of pairwise differences of **SR-IEM** $_{\text{improved}}$ against the best baseline (▲) and of **SR-IEM** $_{\text{improved}}$ against **SR-IEM** $_{\text{basic}}$ (△) are determined by a *t*-test ($p < 0.01$)

(comparing CSRM to NARM) can indeed help make accurate recommendations. Regarding the attention based baseline STAMP, we can see that its performance is not satisfying, especially on *Diginetica*. This indicates that the bias introduced in the attention mechanism will seriously affect the recommendation performance. Among the baselines, SR-GNN performs best in terms of both metrics on all three datasets with an exception that it loses the competition against CSRM on *Diginetica* in terms of Recall@20. It could be due to that SR-GNN is able to explore complex transition relationship between the items to accurately detect user's intent by applying a GGNN. And CSRM extends NARM by incorporating the neighbor sessions, leading to a better performance than other baselines. Thus, CSRM and SR-GNN are chosen for comparison in later experiments.

Next, we zoom in on the performance of our proposals SR-IEM$_{\text{basic}}$ and SR-IEM$_{\text{improved}}$. In general, SR-IEM$_{\text{basic}}$ can outperform the baselines in terms of both metrics on three datasets except that it underperforms SR-GNN in terms of MRR@20 on *Yoochoose 1/4*. This may be explained by the fact that GNN faces an overfitting problem, which is partially alleviated in scenarios with much training data. Moreover, by incorporating the sequential signal and solving the overfitting problem, SR-IEM$_{\text{improved}}$ can improve the performance on the basis of SR-IEM$_{\text{basic}}$ and achieves the best performance in terms of two metrics for all cases on three datasets. In addition, the improvement of SR-IEM$_{\text{improved}}$ over the best baseline SR-GNN in terms of Recall@20 and MRR@20 on *Yoochoose 1/64* is 1.96% and 3.72%, respectively. While on *Yoochoose 1/4* the improvement in terms of Recall@20 (i.e., 1.75%) is similar to that on *Yoochoose 1/64*. However, it is quite different from that in terms of MRR@20 (i.e., 0.09%). We analysis this difference is caused by the size of the training set. This indicates that our proposal can achieve stable improvement in terms of Recall@20 on different scales of training set, while presenting relatively more improvement in terms of MRR@20 for cases with less training data.

5.1.2 Recommendation hits at top positions

To further validate the recommendation ability of our proposal for a limited length of recommendation list, we evaluate the performance of SR-IEM$_{\text{basic}}$ and SR-IEM$_{\text{improved}}$ as well as the outstanding baselines, i.e., CSRM and SR-GNN. The results in terms of Recall@K and MRR@K with the recommendation number K = 5 and 10 are shown in Table 3.

From Table 3, we can observe that SR-IEM$_{\text{improved}}$ can consistently achieve the best performance in terms of both Recall and MRR when K = 5 and 10 on three datasets. This verifies the effectiveness of our proposal in scenarios with different number of items recommended to the user. Moreover, among the baselines, different from the phenomenon that SR-GNN underperforms CSRM in terms of Recall@20 on Diginetica observed in Sect. 5.1.1, SR-GNN performs best among the baselines in all cases. This validates the utility of GNNs on ranking the target item at top positions. For our proposals, comparing SR-IEM$_{\text{basic}}$ and SR-GNN, we can see that SR-IEM$_{\text{basic}}$ can generally outperform SR-GNN in most cases and slightly underperform SR-GNN in terms of MRR@5 as well as MRR@10 on *Yoochoose 1/4*. This is consistent with the observations in Sect. 5.1.1. However, SR-IEM$_{\text{improved}}$ can significantly outperform the baselines in all cases on three datasets, indicating the necessarily of modeling the sequential signal and preventing overfitting.

Table 3 Model performance in terms of Recall@K and MRR@K where $K = 5$ and 10

K	Method	*Yoochoose 1/64*		*Yoochoose 1/4*		*Diginetica*	
		Recall@K	MRR@K	Recall@K	MRR@K	Recall@K	MRR@K
5	CSRM	45.75	27.17	45.51	26.84	25.95	14.39
	SR-GNN	<u>47.35</u>	<u>28.32</u>	<u>48.21</u>	<u>29.47</u>	<u>26.63</u>	<u>15.17</u>
	SR-IEM $_{\text{basic}}$	48.05	29.26	48.44	29.37	27.09	15.18
	SR-IEM $_{\text{improved}}$	**48.49** ▲	**29.56** ▲	**48.62** ▲	**29.49** ▲	**29.00** ▲	**16.21** ▲
10	CSRM	58.88	28.94	59.27	28.68	37.88	15.97
	SR-GNN	<u>60.23</u>	<u>30.06</u>	<u>60.84</u>	<u>31.16</u>	<u>37.89</u>	<u>16.66</u>
	SR-IEM $_{\text{basic}}$	60.76	30.97	61.18	31.08	38.79	16.70
	SR-IEM $_{\text{improved}}$	**61.34** ▲	**31.29** ▲	**61.68** ▲	**31.18** ▲	**41.44** ▲	**17.87** ▲

The results of the best performing baseline and the best performer when $K = 5$ and 10 in each column are underlined and boldfaced, respectively. ▲ denotes a significant improvement of SR-IEM$_{\text{improved}}$ over the best baseline using a paired t-test ($p < 0.01$)

5.2 Computational complexity

To answer RQ2, we analyze the computational complexity of SR-IEM$_{\text{basic}}$ and SR-IEM$_{\text{improved}}$ as well as CSRM and SR-GNN, until generating user preference. The respective computational complexity of CSRM and SR-GNN are $O(td^2 + dM + d^2)$ and $O(s(td^2 + t^3) + d^2)$, where t is the session length and d denotes the item embedding dimension. Here, M is the number of introduced neighbors in CSRM and s indicates the layer number of GGNNs in SR-GNN. For SR-IEM$_{\text{basic}}$, the computational complexity is $O(t^2d + d^2)$, which mainly comes from the importance extraction module $O(t^2d + d^2)$ and from the other components $O(d^2)$. Comparing SR-IEM$_{\text{improved}}$ to SR-IEM$_{\text{basic}}$, the computational complexity is similar, except that SR-IEM$_{\text{improved}}$ has an additional computational complexity of td from the layer normalization and position embeddings, which is negligible. As $t < d$ and $d \ll M$ [28], the complexity of SR-IEM$_{\text{basic}}$ and SR-IEM$_{\text{improved}}$ are clearly lower than SR-GNN and CSRM.

To empirically confirm it, we measure the training and test time of SR-IEM$_{\text{basic}}$ and SR-IEM$_{\text{improved}}$, as well as CSRM and SR-GNN on a single GeForce RTX 2080 Ti GPU. Here, we adopt the Tensorflow version code released by CSRM and SR-GNN[5] for fair comparison. The relative runtime results are presented in Table 4. To make the comparisons more clear, we set the training and test time of SR-IEM$_{\text{improved}}$ as 1 unit in all cases on three datasets, and report the relative time cost of other models. We find that both SR-IEM$_{\text{basic}}$ and SR-IEM$_{\text{improved}}$ have a clearly lower time cost than CSRM and SR-GNN. Moreover, comparing SR-IEM$_{\text{improved}}$ against SR-IEM$_{\text{basic}}$, SR-IEM$_{\text{improved}}$ presents a slight more time cost while their time costs are especially close for test. The aforementioned results indicate that comparing to the state-of-the-art baselines, SR-IEM$_{\text{improved}}$ can achieve the best performance on both recommendation accuracy and computational complexity. This proves its practicability in potential applications. In addition, from Table 4, we can observe that for both CSRM and SR-GNN, the relative training time is higher than the relative test time. This indicates that our proposed SR-IEM$_{\text{improved}}$ can achieve more computational improvement in the training process than that in the test stage.

5.3 Impact of session length

To answer RQ3, we plot the results of SR-IEM$_{\text{basic}}$, SR-IEM$_{\text{improved}}$, CSRM and SR-GNN in terms of Recall@20 and MRR@20 under various session lengths on three datasets in Fig. 4.

As for Recall@20, we can observe that with the session length increasing, the performance of four methods on three datasets consistently first increases and then shows an overall downward trend. This could be explained that for relatively short sessions, with the length increasing, more information about user preference is brought by the interacted items, making user's intent detection more accurate. However, when the length exceeds a threshold value (e.g., 3 on the *Yoochoose* datasets and 2 on *Diginetica*), unrelated items may be included because of user's complex behavior pattern. Moreover, the improvement of SR-IEM$_{\text{improved}}$ over other methods is more obvious for relatively short lengths, especially on *Yoochoose 1/64* and *Yoochoose 1/4*. This can be explained that our importance extraction module is effective on relatively short sessions, since unrelated items in long sessions will increase the difficulty for IEM to extract the item importance accurately.

For MRR@20, on *Yoochoose 1/64* and *Yoochoose 1/4*, different from the trend in terms of Recall@20, all models display a consistent downward trend with the session

[5] https://github.com/CRIPAC-DIG/SR-GNN.

Table 4 Computational complexity and time cost

Method	Complexity	*Yoochoose 1/64*		*Yoochoose 1/4*		*Diginetica*	
		Training	Test	Training	Test	Training	Test
CSRM	$O(td^2 + dM + d^2)$	9.02	6.23	8.66	6.26	7.16	4.90
SR-GNN	$O(s(td^2 + t^3) + d^2)$	6.01	2.66	6.18	2.74	4.99	2.54
SR-IEM$_{\text{basic}}$	$O(t^2d + d^2)$	0.84	0.95	0.83	0.92	0.82	0.95
SR-IEM$_{\text{improved}}$	$O(t^2d + d^2)$	1.00	1.00	1.00	1.00	1.00	1.00

The training and test time of SR-IEM$_{\text{improved}}$ are set to 1 unit, respectively. Then, we can obtain the relative time cost of each model against SR-IEM$_{\text{improved}}$

length increasing. This indicates that the negative impact brought by the increasing of the session length is larger in terms of MRR@20 than Recall@20. However on *Diginetica*, similar to the trend in terms of Recall@20, the performance in terms of MRR@20 still goes up first then decreases with the session length increasing. The difference of results between the *Yoochoose* datasets and *Diginetica* may be due to the fact that the distributions of session lengths are different. As we can observe from Fig. 3, the number of sessions drops sharply with the session length increasing on the *Yoochoose* datasets while it is more gently on *Diginetica*. It means that *Yoochoose* has a larger proportion of sessions with length 1 than *Diginetica*, leading to more training sessions with length 1 and a better performance accordingly. In addition, we can observe that the result gap between CSRM and other methods is becoming larger when the session length increases, which indicates that sequential recommenders are hard to model the long sessions. It is consistent with the findings in Sect. 2.2.

5.4 Analysis on importance extraction module

To answer RQ4, we replace IEM in SR-IEM$_{\text{basic}}$ with two alternatives and then compare the model performance in terms of Recall@20 and MRR@20. The variants can be denoted as follows:

SR-STAMP replaces IEM with an attention mechanism proposed by [17] in our framework. Here, the last item and the mixture of all items in the session are deemed as "key". Specifically, given a session $S = \{x_1, x_2, \ldots, x_t\}$, the external memory is represented as $\mathbf{m}_l = \frac{1}{t}\sum_{i=1}^{t}\mathbf{x}_i$, and the last click is deemed as the short-term memory, i.e., $\mathbf{m}_s = \mathbf{x}_t$. Then, the item importance is decided by both the long- and short-term memories:

$$\beta_i = \mathbf{W}_0^{st}\sigma(\mathbf{W}_1^{st}\mathbf{x}_i + \mathbf{W}_2^{st}\mathbf{m}_l + \mathbf{W}_3^{st}\mathbf{m}_s + \mathbf{b}), \tag{17}$$

where $\mathbf{W}_0^{st} \in \mathbb{R}^{1\times d}, \mathbf{W}_1^{st}, \mathbf{W}_2^{st}, \mathbf{W}_3^{st} \in \mathbb{R}^{d\times d}, \mathbf{b} \in \mathbb{R}^d$ are the trainable parameters (here *st* is short for STAMP), and σ indicates the sigmoid function.

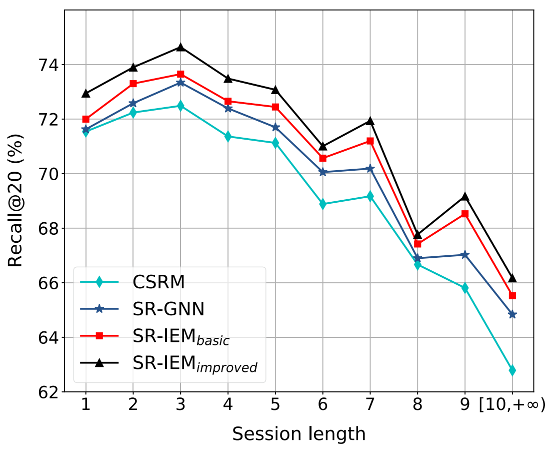
SR-SAT utilizes a self-attention mechanism [25] to distinguish the item importance as

$$\begin{cases} \mathbf{X}^{sa} = softmax(\frac{\mathbf{Q}^{sa}\mathbf{K}^{sa\top}}{\sqrt{d}})\mathbf{V}^{sa} \\ \mathbf{Q}^{sa} = \mathbf{W}_q^{sa}\mathbf{X}, \mathbf{K}^{sa} = \mathbf{W}_k^{sa}\mathbf{X}, \mathbf{V}^{sa} = \mathbf{W}_v^{sa}\mathbf{X} \end{cases}, \tag{18}$$

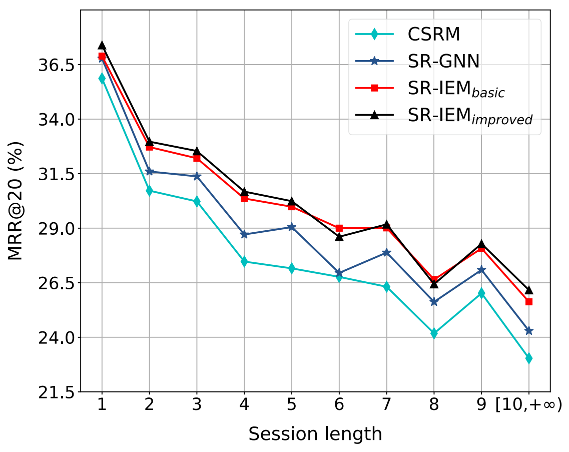
where $\mathbf{W}_q^{sa}, \mathbf{W}_k^{sa}, \mathbf{W}_v^{sa} \in \mathbb{R}^{d\times d}$ are the trainable parameters (here *sa* is short for Self-Attention). We apply an average pooling strategy [38] on $\mathbf{X}^{sa}$, i.e., $\mathbf{z} = \frac{1}{t}\sum_{i=1}^{t}\mathbf{x}_i^{sa}$, to generate the user preference.

The results are shown in Table 5, where we can observe that SR-IEM$_{\text{basic}}$ beats SR-STAMP and SR-SAT in terms of both Recall@20 and MRR@20 on three datasets. And SR-IEM$_{\text{improved}}$ further improves the recommendation accuracy on the basis of SR-IEM$_{\text{basic}}$. Moreover, comparing SR-STAMP to SR-SAT, SR-SAT outperforms SR-STAMP for all cases on three datasets. This could be due to that SR-SAT can explore the contextual signal by modeling the item-item relationship in the session, thus can help to model user's preference. However, SR-STAMP merely relies on the last item and the mixture of all items to distinguish different items, and fails to estimate the importance of each item accurately. In addition, both SR-SAT and SR-STAMP have difficulty in eliminating non-relevant items, which results in a negative effect on capturing user's main intent. In contrast, our proposed IEM component can accurately locate important items and assign relatively high weights to them for user intent detection.

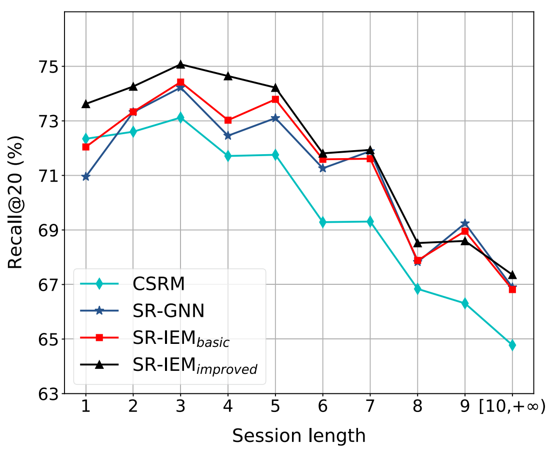
Furthermore, the improvement achieved by SR-IEM$_{\text{improved}}$ against the best variant SR-SAT in terms of Recall@20 and MRR@20 is 1.85% and 3.28% on *Yoochoose 1/64*; 9.97% and 10.77% on *Diginetica*, respectively. Here, a large improvement is achieved in terms of MRR@20 on both two datasets. Differently on *Yoochoose 1/4*, the improvement in terms of Recall@20 (i.e., 1.99%) is higher than MRR@20 (i.e., 0.24%). The difference may be attributed to the size of the dataset. That is, the proposed IEM component is able to rank target item at an early position for cases with relatively few training data, while it is more effective on hitting the target item on datasets of relatively larger scale.



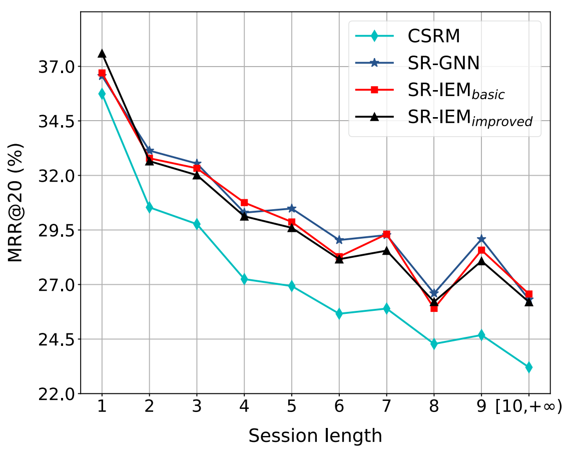
(a) **Recall@20 on *Yoochoose 1/64*.**



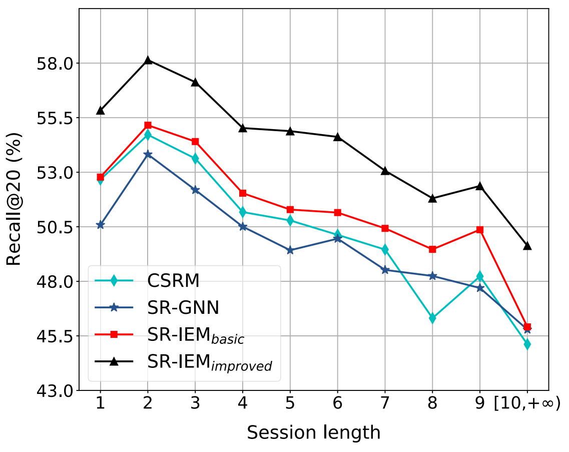
(b) **MRR@20 on *Yoochoose 1/64*.**



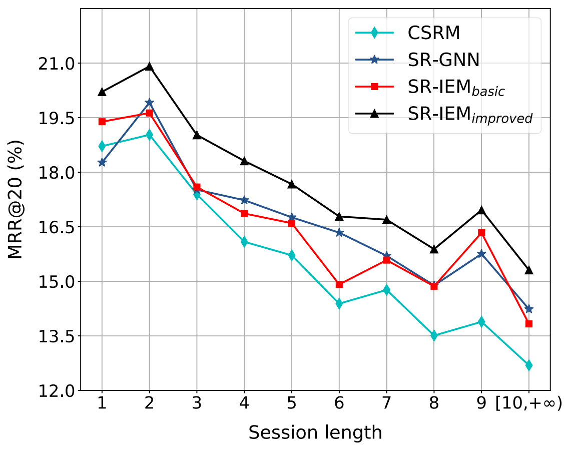
(c) **Recall@20 on *Yoochoose 1/4*.**



(d) **MRR@20 on *Yoochoose 1/4*.**



(e) **Recall@20 on *Diginetica*.**



(b) **MRR@20 on *Diginetica*.**

Fig. 4 Model performance under varying session lengths

Table 5 Model performance of various importance extraction methods used in our framework on three datasets

Method	*Yoochoose 1/64*		*Yoochoose 1/4*		*Diginetica*	
	Recall@20	MRR@20	Recall@20	MRR@20	Recall@20	MRR@20
SR-STAMP	70.02	30.80	70.51	31.24	50.37	16.79
SR-SAT	70.64	31.07	71.19	31.53	50.57	17.09
SR-IEM$_{\text{basic}}$	71.15	31.71	71.67	31.82	52.35	17.64
SR-IEM$_{\text{improved}}$	**71.95**	**32.09**	**72.61**	**31.92**	**55.61**	**18.93**

5.5 Ablation study

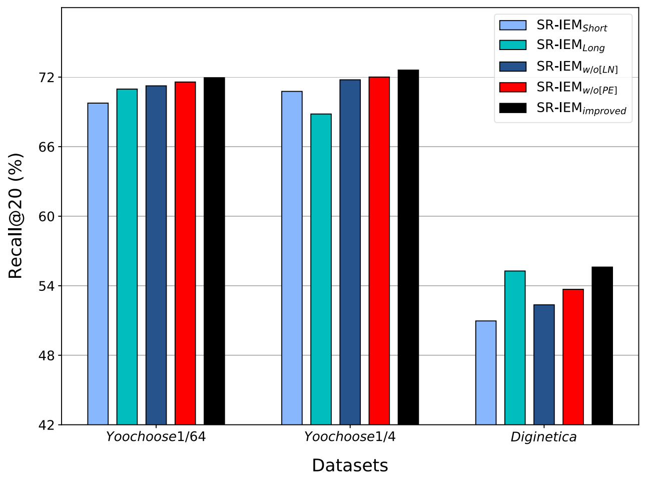
To answer RQ5, we compare our proposal SR-IEM$_{\text{improved}}$ with four variants of SR-IEM$_{\text{improved}}$ to verify the effectiveness of each module in SR-IEM$_{\text{improved}}$:

- **SR-IEM** $_{\text{Short}}$ merely takes the recent interest obtained by Eq. (8) in SR-IEM$_{\text{improved}}$ as the user preference.
- **SR-IEM** $_{\text{Long}}$ merely regards the long-term preference obtained by Eq. (7) in SR-IEM$_{\text{improved}}$ as the user preference.
- **SR-IEM** $_{w/o[\text{PE}]}$ removes the incorporatied position embeddings (i.e., Eq. 6) from SR-IEM$_{\text{improved}}$.
- **SR-IEM** $_{w/o[\text{LN}]}$ removes the layer normalization which is utilized for solving the long-tail problem in SR-IEM$_{\text{improved}}$.

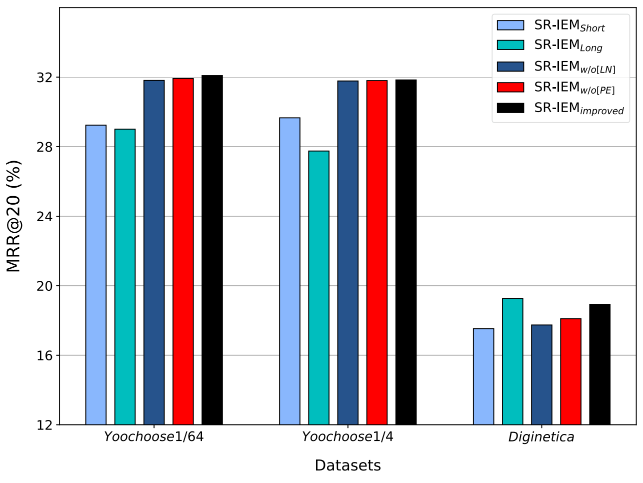
The results of the variants as well as SR-IEM$_{\text{improved}}$ are shown in Fig. 5.

First, as for the long- and short-term preference, from Fig. 5, we can observe that comparing SR-IEM$_{\text{improved}}$ to SR-IEM$_{\text{Short}}$ and SR-IEM$_{\text{Long}}$, SR-IEM$_{\text{improved}}$ can generally outperform these two variants in terms of both metrics on three datasets. This indicates that both the long- and short-term preference can promote the recommendation accuracy. However, SR-IEM$_{\text{improved}}$ loses against SR-IEM$_{\text{Long}}$ in terms of MRR@20 on *Diginetica*. This may be due to that though the last clicked item can represent user's instant need, it may introduce bias in case the last item is unrelated to user's main purpose.

Next, comparing SR-IEM$_{\text{improved}}$ to SR-IEM$_{w/o[\text{PE}]}$ and SR-IEM$_{w/o[\text{LN}]}$, we can observe that both the position embeddings and the layer normalization can improve the recommendation performance by incorporating sequential signal and solving the long-tail problem, respectively. Moreover, the long-tail problem has a higher impact than the sequential information in SR-IEM$_{\text{improved}}$. Furthermore, comparing the improvement brought by SR-IEM$_{\text{improved}}$ over SR-IEM$_{w/o[\text{PE}]}$ and SR-IEM$_{w/o[\text{LN}]}$, we can find that the improvement brought by either the layer normalization or position embeddings is more obvious on *Diginetica* than that on *Yoochoose 1/64* and *Yoochoose 1/4* in terms of both metrics. This may due to two facts: one is that the number of candidate items is relatively large on *Diginetica*, thus the long-tail problem is serious. By solving the long-tail problem using the layer normalization, a relatively large improvement is achieved on *Diginetica*. The other fact is that the average session length of three datasets is different, where it is shorter on *Diginetica* than on the *Yoochoose*



(a) Recall@20



(b) MRR@20

Fig. 5 Results of ablation study on three datasets

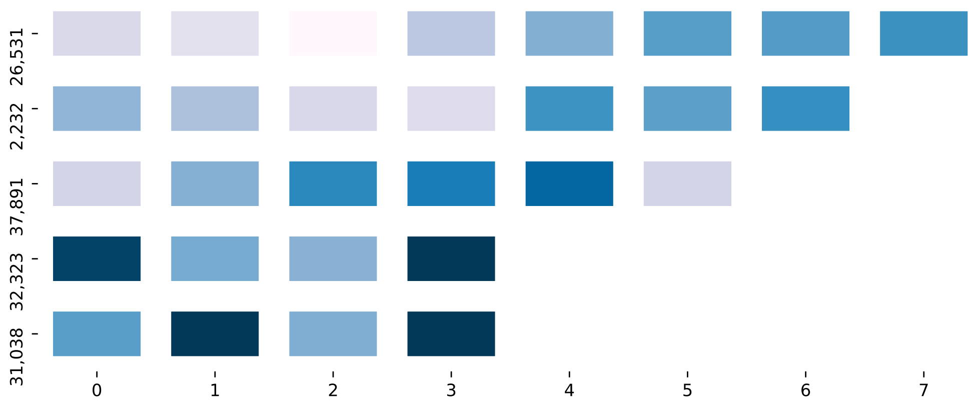


Fig. 6 Visualization of attention weights. The color depth indicates the item importance in each session calculated by Eq. 5, and the numbers on the left denote the session IDs (color figure online)

datasets as shown in Table 1. Through incorporating the position embeddings, the sequential signal can be considered in the SR-IEM$_{\text{improved}}$ model. However, it is relatively hard for long sessions to take the sequential information into consideration, which makes the improvement relatively low on the *Yoochoose* datasets.

5.6 Case study

In order to answer RQ6, as plotted in Fig. 6, we randomly select some example sessions from *Yoochoose 1/64* to intuitively show the effect of the importance extraction module (IEM). Here, the number on the left of each session is the session ID. And a darker color of the block means that the attention weight (calculated by Eq. 5) is larger, which means that the item contributes relatively more to capturing user's main intent. From Fig. 6, we can get the following conclusions:

(1) The item importance does not strictly obey an increased trend according to the sequential order, e.g., recent items may have relatively low importance. In fact, in some sessions, such as session 2232 and 32,323, the former items also have a large importance. However, the IEM component can locate the important items accurately wherever they are.

(2) The last item contributes much to the modeling of user preference, since it can represent user's recent interest. However, not all sessions obey this rule, for example, session 37,891. This indicates that the last item may be an unrelated item. For such cases, the recommendations need to rely on the long-term preference. This indicates the necessity of combining the long- and short-term interests within current session.

(3) For a relatively long session such as session 26,531, the user's interest may migrate because of user's uncertain behavior, making the user preference hard to capture. This is corresponding to the experimental results in Sect. 5.3.

6 Conclusions and future work

We propose an Importance Extraction Module for Session-based Recommendation (**SR-IEM** $_{\text{improved}}$), which incorporates user's long-term preference and current interest for item recommendations. We modify the self-attention mechanism to estimate the item importance for generating the long-term preference in the session. Specifically, we discriminatively combine the representation of the items, which are generated by adding the position embeddings to the item embeddings. Then, the long-term interest is combined with user's current interest released by the last item for producing the item predictions, where a normalization layer is utilized to solve the serious long-tail problem. Experimental results validate the superiority of SR-IEM$_{\text{improved}}$ over state-of-the-art methods in terms of Recall and MRR for session-based recommendation at relatively low time cost. In addition, the importance extraction module (IEM) in our proposal can accurately estimate the item importance and is particularly effective on short sessions.

As to future work, we would like to evaluate SR-IEM$_{\text{improved}}$ on other datasets to further examine its robustness. Moreover, we also have interest in introducing multi behaviors [18] into session-based recommendation to help identify user's main intent in the ongoing session. In addition, we plan to take the time span between items into consideration when estimating the item importance.

Acknowledgements This work was partially supported by the National Natural Science Foundation of China under No. 61702526 and the Postgraduate Scientific Research Innovation Project of Hunan Province under No. CX20200055. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng., 17(6):734–749, 2005. DOI: https://doi.org/10.1109/TKDE.2005.99
2. Chen W, Cai F, Chen H, de Rijke M (2019) A dynamic co-attention network for session-based recommendation. In: Proceedings of the 28th ACM international conference on information and knowledge management, pp 1461–1470. ACM. https://doi.org/10.1145/3357384.3357964
3. Chen W, Cai F, Chen H, de Rijke M (2019) Joint neural collaborative filtering for recommender systems. ACM Trans. Inf. Syst. 37(4):39–39. https://doi.org/10.1145/3343117
4. Du G, Zhang J, Luo Z, Ma F, Ma L, Li S (2020) Joint imbalanced classification and feature selection for hospital readmissions. Knowl. Based Syst. 200:106020. https://doi.org/10.1016/j.knosys.2020.106020
5. He X, Deng K, Wang X, Li Y, Zhang Y, Wang M (2020) Lightgcn: Simplifying and powering graph convolution network for recommendation. In: Proceedings of the 43rd international ACM SIGIR conference on research and development in Information Retrieval, pp 639–648. ACM. https://doi.org/10.1145/3397271.3401063
6. He X, Liao L, Zhang H, Nie L, Hu X, Chua T (2017) Neural collaborative filtering. In: Proceedings of the 26th international conference on world wide web, pp 173–182. ACM. https://doi.org/10.1145/3038912.3052569
7. Hidasi B, Karatzoglou A, Baltrunas L, Tikk D (2016) Session-based recommendations with recurrent neural networks. In: 4th international conference on learning representations. arxiv:1511.06939
8. Jin R, Chai J Y, Si L (2004) An automatic weighting scheme for collaborative filtering. In: Proceedings of the 27th annual international ACM SIGIR conference on research and development in Information Retrieval, pp 337–344. ACM. https://doi.org/10.1145/1008992.1009051
9. Kang W, McAuley J J (2018) Self-attentive sequential recommendation. In: IEEE international conference on data mining, pp 197–206. IEEE Computer Society . https://doi.org/10.1109/ICDM.2018.00035
10. Kipf T N, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: 5th international conference on learning representations. https://openreview.net/forum?id=SJU4ayYgl
11. Koren Y (2008) Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining, pp 426–434. ACM. https://doi.org/10.1145/1401890.1401944
12. Y. Koren, R. M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. Computer, 42(8), 30–37, 2009. DOI: https://doi.org/10.1109/MC.2009.263
13. Lee D D, Seung H S (2000) Algorithms for non-negative matrix factorization. In: Proceedings of the international conference on neural information processing systems, pp 556–562. MIT Press. https://proceedings.neurips.cc/paper/2000/hash/f9d1152547c0bde01830b7e8bd60024c-Abstract.html
14. H. Li, H. Li, S. Zhang, Z. Zhong, and J. Cheng. Intelligent learning system based on personalized recommendation technology. Neural Comput. Appl., 31(9):4455–4462, 2019. DOI: https://doi.org/10.1007/s00521-018-3510-5
15. Li J, Ren P, Chen Z, Ren Z, Lian T, Ma J (2017) Neural attentive session-based recommendation. In: Proceedings of the 2017 ACM on conference on information and knowledge management, pp 1419–1428. ACM. https://doi.org/10.1145/3132847.3132926
16. Linden G, Smith B, York J (2003) Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Comput 7(1):76–80. https://doi.org/10.1109/MIC.2003.1167344
17. Liu Q, Zeng Y, Mokhosi R, Zhang H (2018) STAMP: short-term attention/memory priority model for session-based recommendation. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp 1831–1839. ACM. https://doi.org/10.1145/3219819.3219950
18. Meng W, Yang D, Xiao Y (2020) Incorporating user micro-behaviors and item knowledge into multi-task learning for session-based recommendation. In: Proceedings of the 43rd international ACM SIGIR conference on research and development in Information Retrieval, pp 1091–1100. ACM. https://doi.org/10.1145/3397271.3401098
19. Pan Z, Cai F, Ling Y, de Rijke M (2020) Rethinking item importance in session-based recommendation. In: Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, pp 1837–1840. ACM. https://doi.org/10.1145/3397271.3401274
20. Qiu R, Li J, Huang Z, Yin H (2019) Rethinking the item order in session-based recommendation with graph neural networks. In: Proceedings of the 28th ACM international conference on information and knowledge management, pp 579–588. ACM. https://doi.org/10.1145/3357384.3358010
21. L. Ravi, V. Subramaniyaswamy, M. Devarajan, S. Natarajan, and V. Vijayakumar. Enhancing recommendation stability of collaborative filtering recommender system through bio-inspired clustering ensemble method. Neural Comput. Appl., 32(7):2141–2164, 2020. DOI: https://doi.org/10.1007/s00521-018-3891-5
22. Rendle S, Freudenthaler C, Schmidt-Thieme L (2010) Factorizing personalized markov chains for next-basket recommendation. In: Proceedings of the 19th international conference on world wide web, pp 811–820. ACM. https://doi.org/10.1145/1772690.1772773
23. Sarwar B. M., Karypis G, Konstan J. A., Riedl J (2001) Item-based collaborative filtering recommendation algorithms. In: Proceedings of the tenth international world wide web conference, pp 285–295. ACM. https://doi.org/10.1145/371920.372071
24. Shani G, Heckerman D, Brafman R I (2005) An mdp-based recommender system. J Mach Learn Res 6:1265–1295. http://jmlr.org/papers/v6/shani05a.html
25. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser L, Polosukhin I (2017) Attention is all you need. In: Proceedings of the international conference on neural information processing systems, pp 5998–6008. MIT Press. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
26. Vinyals O, Bengio S, Kudlur M (2016) Order matters: sequence to sequence for sets. In: 4th international conference on learning representations . arxiv:1511.06391
27. Wang F, Xiang X, Cheng J, Yuille A L (2017) Normface: L_2 hypersphere embedding for face verification. In: Proceedings of the 2017 ACM on multimedia conference, pp 1041–1049. ACM. https://doi.org/10.1145/3123266.3123359
28. Wang M, Ren P, Mei L, Chen Z, Ma J, de Rijke M (2019) A collaborative session-based recommendation approach with parallel memory modules. In: Proceedings of the 42nd international

ACM SIGIR conference on research and development in information retrieval, pp 345–354. ACM. https://doi.org/10.1145/3331184.3331210

29. Wang X, He X, Wang M, Feng F, Chua T (2019) Neural graph collaborative filtering. In: Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval, pp 165–174. ACM. https://doi.org/10.1145/3331184.3331267
30. Wang Z, Wei W, Cong G, Li X, Mao X, Qiu M (2020) Global context enhanced graph neural networks for session-based recommendation. In: Proceedings of the 43rd international ACM SIGIR conference on research and development in Information Retrieval, pp 169–178. ACM. https://doi.org/10.1145/3397271.3401142
31. Weston J, Chopra S, Bordes A (2015) Memory networks. In: 3rd international conference on learning representations. arxiv:1410.3916
32. Wu S, Tang Y, Zhu Y, Wang L, Xie X, Tan T (2019) Session-based recommendation with graph neural networks. In: The thirty-third AAAI conference on artificial intelligence, pp 346–353. AAAI Press. https://doi.org/10.1609/aaai.v33i01.3301346
33. Xu C, Zhao P, Liu Y, Sheng V S, Xu J, Zhuang F, Fang J, Zhou X (2019) Graph contextualized self-attention network for session-based recommendation. In: Proceedings of the twenty-eighth international joint conference on artificial intelligence, pp 3940–3946. https://doi.org/10.24963/ijcai.2019/547
34. Ying H, Zhuang F, Zhang F, Liu Y, Xu G, Xie X, Xiong H, Wu J (2018) Sequential recommender system based on hierarchical attention networks. In: Proceedings of the twenty-seventh international joint conference on artificial intelligence, pp 3926–3932. https://doi.org/10.24963/ijcai.2018/546
35. H. Zhang, Y. Ji, J. Li, and Y. Ye. A triple wing harmonium model for movie recommendation. IEEE Trans. Ind. Informatics, 12(1), 231–239, 2016. DOI: https://doi.org/10.1109/TII.2015.2475218
36. H. Zhang, Y. Sun, M. Zhao, T. W. S. Chow, and Q. M. J. Wu. Bridging user interest to item content for recommender systems: An optimization model. IEEE Trans. Cybern., 50(10):4268–4280, 2020. DOI: https://doi.org/10.1109/TCYB.2019.2900159
37. Zhang J, Lin Y, Jiang M, Li S, Tang Y, Tan K C (2020) Multi-label feature selection via global relevance and redundancy optimization. In: Proceedings of the twenty-ninth international joint conference on artificial intelligence, pp 2512–2518. https://doi.org/10.24963/ijcai.2020/348
38. Zhang S, Tay Y, Yao L, Sun A (2018) Next item recommendation with self-attention. arXiv preprint arXiv:1808.06414
39. Zhang S, Yao L, Sun A, Tay Y (2019) Deep learning based recommender system: a survey and new perspectives. ACM Comput Surv 52(1):51–53. https://doi.org/10.1145/3285029

Authors and Affiliations

Zhiqiang Pan[1] · **Fei Cai**[1] · **Wanyu Chen**[1] · **Honghui Chen**[1]

✉ Fei Cai
caifei@nudt.edu.cn

[1] Science and Technology on Information Systems Engineering Labratory, National University of Defense Technology, Changsha 410000, China