SEPARATION AND FUSION GRAPH NETWORKS FOR SESSION-BASED RECOMMENDATION

Jingyuan HE[12], Bailong YANG[3]*, A. RUHAN[4], Jinjin ZHANG[5]

Previous models of session-based recommendation simply concatenate incoming graph and outgoing graph as a joint graph, and they model item representation with the relationship of item graphs but ignore the item-self information. Additionally, the incoming graph and outgoing graph indicate different relational patterns, they should be modeled separately, and how to balance their importance in a flexible and end-to-end manner could be crucial for performance enhancement. This paper presents Separation and Fusion graph networks (SEFU) for session-based recommendation. For each session, SEFU first represents the incoming and outgoing graphs with two separate graph neural encoders and item-self information to generate item representation, then leverages an attention-based gating mechanism to selectively fuse representations of incoming graph and outgoing graph. Extensive experiments present SEFU greatly outperforms other models, verifying efficacy of our proposed method on session modeling.

Keywords: graph neural network; incoming graph; item-self information; outgoing graph; session-based recommendation

1. Introduction

Recommender systems (RS) have evolved in modern society for helping users to make choice from the large number of products and services. The main core of current recommender systems, such as collaborative filtering RS, is modeling the long-term static user preference. However, in these systems, the user's intent after a certain period of time may be easily submerged by his/her historical behaviors, making the recommendations inopportune and/or inappropriate. To address this, Session-based Recommendation (SR) models a transaction with multiple purchased items in one shopping event as a session, and then recommends next clicked item based on interaction of items, without using user identification information [1]. As a result, SR can better model user's preference dynamically and is applied to e-commerce and online-searching systems.

[1] Lecturer, School of Mathematics and Computer Science, Yan'an University, Yan'an, China, e-mail: yau_hejingyuan@163.com
[2] Xi'an Research Institute of Hi-Tech, Xi'an, China, e-mail: yau_hejingyuan@163.com
[3] Prof., Xi'an Research Institute of Hi-Tech, Xi'an, China, e-mail: xa_403@163.com
[4] Lecturer, School of Accounting and Finance, Xi'an Peihua University, Xi'an, China, e-mail: aruhan@peihua.edu.cn
[5] Lecturer, Xi'an Technological University, Xi'an, China, e-mail: 598791278@qq.com

Current SR approaches can be classified into three categories in terms of how they model the session. One classical approach is the Markov Decision Process (MDP) modeling. The MDP-based models introduce the Markov assumption simplifies the recommendation model. To better model user interaction histories, the second line of approaches model the session as sequences [1], [2]. Sequence modeling methods always treat the step-backed items as new items without characterizing the vacillation behaviors of users. To address this drawback, recent methods consider sessions as graphs, and recommend items with the help of prevailing Graph Neural Networks (GNN) [3], [4].

When we look more closely at the connections between items in each session (see for example in Fig. 1), there are two kinds of connections: the incoming connections from other items to each target item and the outgoing connections from each target item to others. Although existing GNN-based methods have achieved excellent performance, these methods simply combine the incoming connections and the outgoing connections into a joint graph, neglecting their distinct effects in item representation learning. In addition, previous methods ignore the information of the item self when constructing the item representation. It indicates that item vector which is built by GNN from previous methods lacks item-self information and is incomplete.

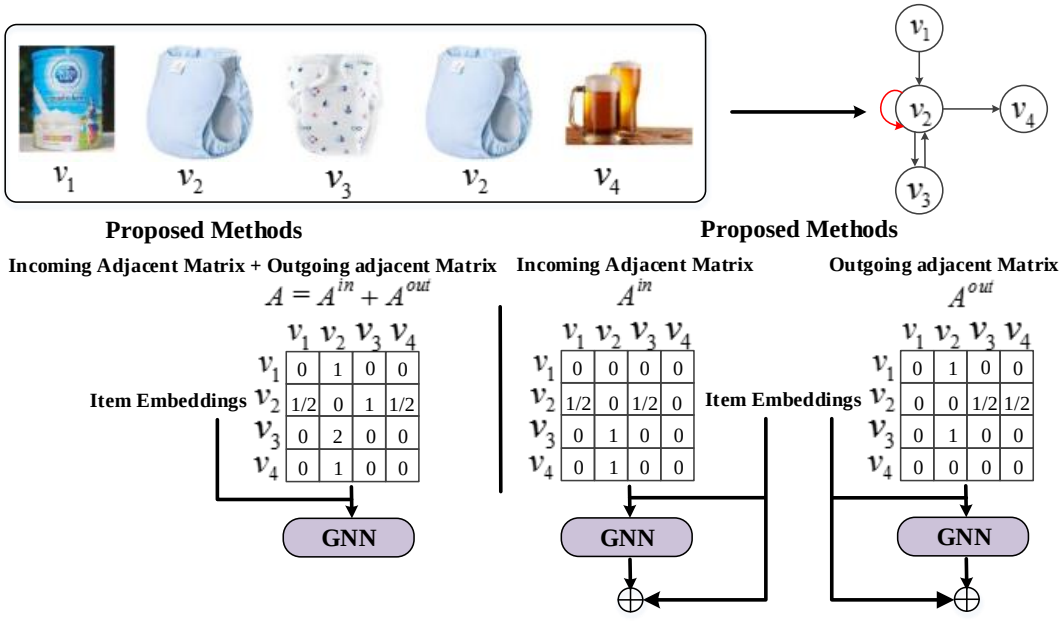

Fig. 1. A sketch of a session and its corresponding session graph

In this paper, we propose a Separation and Fusion graph networks (SEFU) for SR. For each session, we consider the incoming connections and outgoing connections as two separate graphs which are independently processed via two

gated GNN-based encoders, and we integrate them with item-self information to deliver two representations of the session. Then, a gating mechanism is developed to fuse the two individual session representations by flexibly weighting their influences on the prediction of different items.

2. Related work

Traditional SR methods. The traditional methods for SR are mainly based on the idea of item-to-item or co-occurrences of the items. Rendle [5] combine Matrix Factorization and MC to model sequential behaviors for the next basket recommendation. However, these methods neglect the previous clicks and discard the useful information in the sequence.

RNN-based SR methods. Recurrent neural network (RNN) has attracted great concern since its capabilities in modeling sequential behavior. Li et al. [2] designed NARM to build user's main purpose by item-level attention mechanism (AM). Pan et al. [6] consider importance of items to improve recommendation performance.

GNN-based SR methods. GNN is increasingly used in SR recently. Li et al. [7] proposed Disen-GNN for SR. [4] employ both GNN and the self-attention mechanism to learn latent vectors for all nodes and long-range dependencies between the distant items for recommendation. Some researchers incorporate graph-structured data and target-aware attention module for SR. Xia et al. [8] proposed SHT to enhance user representations and robustness of recommender systems by exploring the global collaborative relationships in an explicit way.

Our work is related to SR with GNN (SR-GNN) [3]. Indeed, our work shares the same backbone with SR-GNN. The major difference between SR-GNN and our work is that we emphasize the different roles of the incoming and outgoing graphs in session representation learning via two independent GNN encoders and flexibly adjust their importance weights by using the attention mechanism. SR-GNN also separates the incoming graph from the outgoing graph, but it simply concatenates the corresponding adjacency matrices into a joint matrix column-wisely, where the computation will degenerate to a naive GNN encoding with a directed adjacency matrix. Additionally, we consider the role of item-self information in modeling item representation, which further strengthens the accuracy of session representation.

3. Methods

In SR, $\mathrm{V}=\{\mathrm{v}_1,\mathrm{v}_2,\cdots,\mathrm{v}_{|\mathrm{V}|}\}$ denotes the set of unique items. We use $[v_1,v_2,\cdots,v_n]$ to denote an interaction session, in which v_i is the i-th interaction item during session.

3.1. Framework

As shown in Fig. 2, SEFU consists of an incoming session encoder (ISE), an outgoing session encoder (OSE), and a recommendation decoder. In ISE, item embeddings and incoming adjacent matrix are converted into two high dimensional representations with the help of the GNN and soft-attention mechanism: one is an incoming local embedding and the other is an incoming global embedding. In OSE, it converts item embeddings and outgoing adjacent matrix into two high dimensional representations with help of another GNN and soft-attention mechanism: one is an outgoing local embedding and the other is an outgoing global embedding. Finally, outputs of ISE and OSE are fed into recommendation decoder. The output of the framework is a recommendation score.

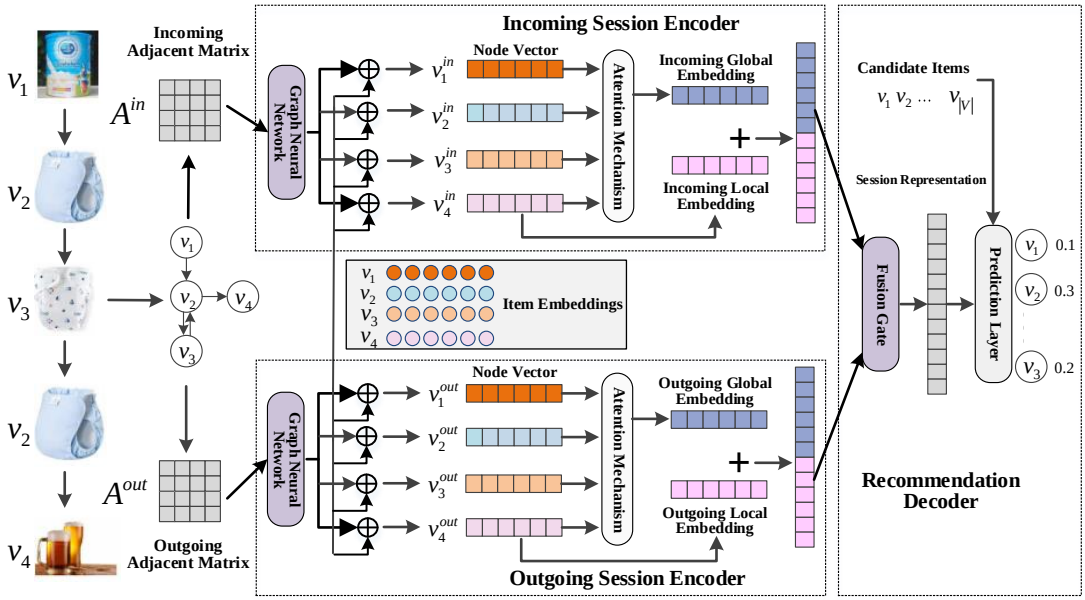

Fig. 2. The framework of SEFU

3.2. Graph construction

Given an input session $[v_1,\cdots,v_i,\cdots,v_n]$, we firstly model this session as a direct graph where v_i is denoted as a node and each edge (v_i,v_{i+1}) represents the item v_{i+1} is clicked after item v_i in this session. We can build individual incoming adjacent matrix $\boldsymbol{A}^{in}$ and outgoing adjacent matrix $\boldsymbol{A}^{out}$ according to the connection edges. For example, the corresponding graph and the adjacent matrices for the session $[v_1,v_2,v_3,v_2,v_4]$ are shown in Fig. 1. Additionally, we embed each item v_i into a space $\boldsymbol{v}_i \in \mathrm{R}^d$. Then, incoming node vector $\boldsymbol{v}_i^{in}$ can be learnt by using a GNN with the help of the incoming graph which contains all the item embeddings $[\boldsymbol{v}_1,\boldsymbol{v}_2,\cdots,\boldsymbol{v}_n]$ and $\boldsymbol{A}_i^{in}$

that is the i-th row of A^{in} corresponding to item v_i. Similarly, the outgoing node vector v_i^{out} of item v_i can be learnt by using another GNN with the help of the outgoing graph which contains all the item embeddings $[v_1, v_2, \cdots, v_n]$ and A_i^{out} that is i-th row of A^{out} corresponding to item v_i.

3.3. Incoming session encoder

For a session, we learn the latent vectors of nodes via gated GNN. [9] proposed a gated GNN based on the vanilla GNN. Many researchers propose utilize the gated GNN as encoders to learn node vectors in a session graph. The gated GNN structure in ISE is shown like below:

$$a_i^t = \mathbf{A}_i^{in}[v_1^{t-1}, \cdots, v_n^{t-1}]^{\mathrm{T}}\mathbf{H} + \mathrm{b} \tag{1}$$

$$\mathbf{z}_i^t = \sigma\left(\mathbf{W}_z a_i^t + \mathbf{U}_z v_i^{t-1}\right) \tag{2}$$

$$r_i^t = \sigma\left(\mathbf{W}_r a_i^t + \mathbf{U}_r v_i^{t-1}\right) \tag{3}$$

$$\tilde{\mathbf{v}}_i^t = \tanh(\mathbf{W}_o a_i^t + \mathbf{U}_o(r_i^t \ \mathrm{e} \ v_i^{t-1})) \tag{4}$$

$$v_i^t = \left(1 - \mathbf{z}_i^t\right) \mathrm{e} \ v_i^{t-1} + \mathbf{z}_i^t \ \mathrm{e} \ \tilde{\mathbf{v}}_i^t \tag{5}$$

Where $\mathbf{z}_i^t$ is update gate, r_i^t is reset gate. $\tilde{\mathbf{v}}_i^t$ is a candidate state. $\mathbf{W}_z, \mathbf{U}_z, \mathbf{W}_r, \mathbf{U}_r, \mathbf{W}_o$, and $\mathbf{U}_o$ are learnable parameters. e denotes element-wise multiplication. $\sigma(\cdot)$ is the sigmoid function. Additionally, $[v_1^{t-1}, \cdots, v_n^{t-1}]$ is the list of the node embedding. Equation (1) is used for information propagation between different nodes with respect to A_i^{in}.

For a session graph, the gated GNN handles the nodes at the same time. More concretely, a_i^t firstly extracts the latent vectors of the nodes which interact with the i-th node and then they are fed as the network's input. Then, update gate and reset gate use sigmoid function to determine what information to be retained or discarded, respectively. Thirdly, candidate state is constructed by current state, reset gate, and previous state. Finally, previous state and candidate state are combined with consideration of the update gate to construct the final state. We can obtain all the node vectors $[v_1^{in\_gnn}, v_2^{in\_gnn}, \cdots, v_n^{in\_gnn}]$ after all the nodes in the session are processed into the gated GNN with respect to the incoming graph. However, the node-self information loss occur in gated GNN operations. Thus we add the node-self information on the node vectors which are produced by gated GNN, which makes the item vectors much more accurate. Then, the final node vectors with respect to the incoming graph and item-self information are denoted as $[v_1^{in}, v_2^{in}, \cdots, v_n^{in}]$, which is computed as:

$$v_i^{in} = \mathbf{W}_{in}\mathbf{v}_i^{in\_gnn} \oplus \mathbf{W}_i\mathbf{v}_i \tag{6}$$

Since a session is directly composed by nodes which are ordered by timestamp, we plan to generate local embedding and global embedding with

consideration of node vectors. Then, we take local embedding and global embedding to build session representation.

For modeling incoming local embedding, since there is a very strong causality relationship between user's last action and next interacted item, we use node vector $\boldsymbol{v}_n^{in}$ of the last clicked item as incoming local embedding, namely $\boldsymbol{s}_l^{in} = \boldsymbol{v}_n^{in}$.

For modeling the outgoing global embedding, we consider the whole sequential behavior in the session. Since not all the items reflect equally for modeling the global embedding, we use soft AM to extract global preference specific items that are significant to the global embedding. Finally, the incoming global embedding $\boldsymbol{s}_g^{in}$ is aggregated by the vectors of those informative items, which is computed as:

$$\boldsymbol{s}_g^{in} = \sum_{i=1}^{n} \alpha_i^{in} \boldsymbol{v}_i^{in} \tag{7}$$

Where $\boldsymbol{v}_n^{in}$ is node vector of the i-th item, weighting factor α_i^{in} models relationship between $\boldsymbol{v}_n^{in}$ and previous clicked items $\boldsymbol{v}_i^{in}$ by computing their similarity, which can be defined as:

$$\alpha_i^{in} = \mathbf{q}^{\mathrm{T}} \sigma(\mathbf{W}_1^{\mathrm{in}} \boldsymbol{v}_n^{in} + \mathbf{W}_2^{\mathrm{in}} \boldsymbol{v}_i^{in} + \boldsymbol{c}^{in}) \tag{8}$$

Where $\mathbf{W}_1^{\mathrm{in}}$ and $\mathbf{W}_2^{\mathrm{in}}$ transform $\boldsymbol{v}_n^{in}$ and $\boldsymbol{v}_i^{in}$ into a latent space, respectively. $\sigma(\cdot)$ is the sigmoid function. $\mathbf{q}$ is a weight vector and $\mathbf{q}^{\mathrm{T}}$ denotes its transpose.

Then, we concatenate $\boldsymbol{s}_l^{in}$ and $\boldsymbol{s}_g^{in}$ as the hybrid incoming session representation $\boldsymbol{s}^{in}$:

$$\boldsymbol{s}^{in} = [\boldsymbol{s}_l^{in} : \boldsymbol{s}_g^{in}] \tag{9}$$

3.4. Outgoing session encoder

Similar to the incoming session encoder, we use another gated GNN to compute the node vectors in a session with the help of item embedding $[\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_n]$ and outgoing adjacent matrix $\boldsymbol{A}^{out}$. After adding the item-self information on the corresponding node vectors which with respect to outgoing graph, the final node vectors are denoted as $[\boldsymbol{v}_1^{out}, \boldsymbol{v}_2^{out}, \cdots, \boldsymbol{v}_n^{out}]$.

For modeling the outgoing local embedding, we also use the node vector $\boldsymbol{v}_n^{out}$ of the last clicked item as outgoing local embedding, namely $\boldsymbol{s}_l^{out} = \boldsymbol{v}_n^{out}$

For modeling the outgoing global embedding, since every node contributes different information to global embedding, the soft AM is also introduced to generate outgoing global embedding $\boldsymbol{s}_g^{out}$. The process is similar to the process of building the incoming global embedding. It is defined as:

$$\alpha_i^{out} = \mathbf{q}^{\mathrm{T}} \sigma(\mathbf{W}_1^{out} \boldsymbol{v}_n^{out} + \mathbf{W}_2^{out} \boldsymbol{v}_i^{out} + \boldsymbol{c}^{out}) \tag{10}$$

$$\boldsymbol{s}_g^{out} = \sum_{i=1}^{n} \alpha_i^{out} \boldsymbol{v}_i^{out} \tag{11}$$

Finally, the hybrid outgoing session representation $\boldsymbol{s}^{out}$ is designed as:

$$\boldsymbol{s}^{out}=[\boldsymbol{s}_l^{out} : \boldsymbol{s}_g^{out}] \tag{12}$$

3.5. Recommendation decoder

Recommendation decoder is to fuse hybrid incoming and outgoing representations of the session with an adaptive importance proportion, and compute the probability of the next clicked item. Since incoming graph and outgoing graph represent different information and have different roles for building session representation, we build the final session representation $\boldsymbol{s}^{final}$ through an attention-based fusion gating mechanism which controls the information flow from outputs of ISE and OSE according to their importance:

$$\boldsymbol{s}^{final} = f_t \boldsymbol{s}^{in} + (1- f_t)\boldsymbol{s}^{out} \tag{13}$$

where the fusion gate f_t is determined by:

$$f_t = \sigma(\boldsymbol{W}^{in}\boldsymbol{s}^{in} + \boldsymbol{W}^{out}\boldsymbol{s}^{out}) \tag{14}$$

When the incoming graph plays a more important role in the session, f_t is adjusted to be larger adaptively, whereas $(1- f_t)$ decreases. Conversely, f_t decreases and $(1- f_t)$ increases when outgoing graph plays key role in modeling final session representation.

Then, we calculate recommendation score $\hat{y}_i$ for candidate item v_i:

$$\hat{y}_i = \text{softmax}(\boldsymbol{v}_i \boldsymbol{B}\boldsymbol{s}^{final}) \tag{15}$$

Where $\boldsymbol{B} \in R^{d\times 2d}$ transforms session representation into latent space R^d.

In our model, loss is defined as:

$$Loss = \sum_{i=1}^{|v|} y_i \log(\hat{y}_i) + (1- y_i)\log(1- \hat{y}_i) \tag{16}$$

4. Experiments

4.1. Datasets and experiment settings

We conduct experiments on Yoochoose and Diginetica datasets from RecSys Challenge2015 and CIKM Cup 2016, respectively. The statistics of them are shown in Table 1.

Table 1

Statistics of datasets

Statistics	Yoochoose 1/64	Yoochoose 1/4	Diginetica
# of training sessions	430,328	6,145,883	719,470
# of test sessions	55,464	55,861	60,858
# of items	37,484	37,484	43,098

We implement SEFU with Tensorflow and carry out experiments on a Nvidia T4. Processing of dataset is followed by [3]. We firstly filter out the session

whose length is 1 and where item occurrence is less than 5 in two datasets. Following previous works [2], [3], the item embedding on each dataset is 100. We use Adam to update parameters when training and initial learning rate is 0.001 and will decay by 0.1 after every 3 epochs. Additionally, batch size is 100, and L2 penalty is 10^{-5} in our model. We use Recall@20 and MRR@20 to measure prediction accuracy and order of recommendation ranking.

4.2. Model comparisons

Since our framework only uses information from current session, we compare SEFU with some mainstream SR models which are based on the current session. The performances of baseline methods and our proposed SEFU are shown in Table 2.

Table 2

Results with different methods

Methods	Yoochoose 1/64		Yoochoose 1/4		Diginetica	
	Recall@20	MRR@20	Recall@20	MRR@20	Recall@20	MRR@20
FPMC	45.62	15.01	51.52	21.21	26.53	8.95
Item-KNN	51.60	21.44	52.34	21.69	28.75	9.36
GRU4Rec	66.70	22.89	65.66	28.24	44.56	14.32
NARM	68.32	28.34	69.10	29.13	48.32	16.29
GC-SAN	69.61	30.18	70.39	30.07	49.05	16.62
SR-GNN	69.17	30.12	69.37	29.47	51.03	17.07
Disen-GNN	71.46	31.36	-	-	53.79	18.99
SEFU-IO	71.28	31.31	70.68	30.28	51.38	17.19
SEFU	**71.64**	**31.64**	**70.95**	**30.42**	**53.85**	**19.07**

From Table 2, we can observe that:

In traditional methods, the result of FPMC [10] indicates consecutive items have dependency relationships. Besides, the performance of Item-KNN [11] is better than FPMC. It denotes the last clicked item very significant in recommendation.

Since neural network methods have the ability to model the complex contextual information, the RNN-based methods of GRU4Rec and NARM excessively outperform the traditional methods. It indicates that modeling the session representation considers the transitions between the consecutive items are helpful for SR. Compared to the methods based on RNNs (GRU4Rec [12] and NARM [2]), the models based on GNNs (SR-GNN [3], GC-SAN [4] and Disen-GNN [7]) perform better. This may be because the graph-structured data is able to capture more complex item transition patterns, and it verifies that GNN is friendly to capture the information in a given session.

Data scale does not affect the performance. Compared with Disen-GNN, SEFU gets improvement of 0.18 and 0.28, respectively for a small dataset like Yoochoose 1/64, and gets improvement of 0.06 and 0.08, respectively for a dataset like Diginetica. Especially, compared with SR-GNN which has a similar

construction to us, SEFU-IO obviously gets large improvement on all datasets. We suppose the reason is that the previous methods cannot distinguish roles of the incoming graph and the outgoing graph to model the session representation. Additionally, SEFU achieves the best performance. The results confirm our proposed methodology of introducing item-self information in modeling item representation, separating incoming graph and outgoing graph, and balancing their importance in a flexible and end-to-end manner are reasonable and effective for the GNN based methods.

4.3. Effect of different connections

To illustrate the effect of every encoder in our model, we compare SEFU with two variants IC-SR and OC-SR. IC-SR denotes the final session representation is directly generated only from the ISE. OC-SR refers that the final session representation is built merely from the OSE. Fig. 3 shows the results of IC-SR, OC-SR, and SEFU for SR.

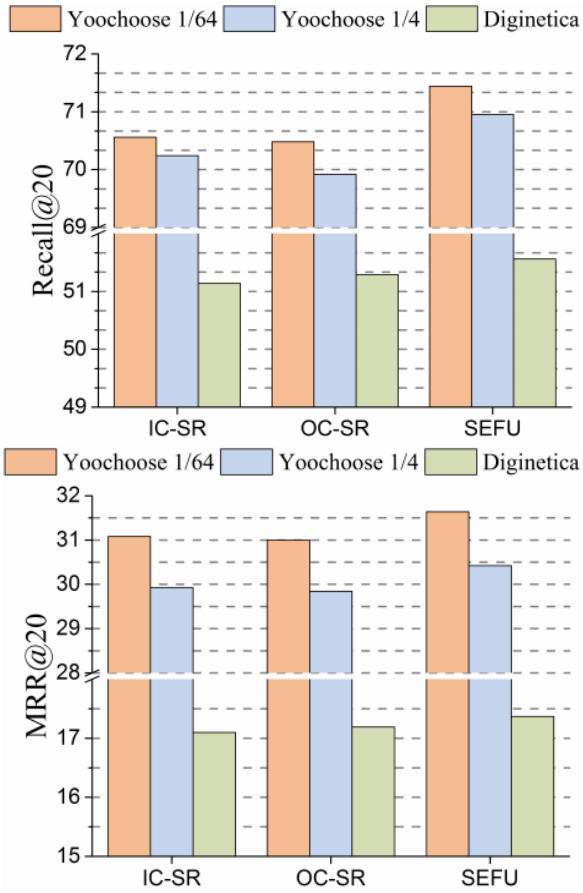

Fig. 3. Effect of the different connections for modeling the final session representation for SR

From Fig. 3, we can observe that:

IC-SR and OC-SR both perform better than GRU4Rec and NARM. Their differences are: IC-SR and OC-SR convert the session into a graph structure, and the others directly use sequential information to build the session representation. The results indicate that modeling session into a graph structure is friendly to recommendation. Moreover, the incoming graph with item-self information or the outgoing graph with item-self information can capture more information related to the session.

For the same data source of Yoochoose, results of IC-SR are lightly higher than those of OC-SR. It indicates the incoming graph very important in modeling session representation for Yoochoose dataset, whereas the outgoing graph is more effective than the incoming graph to improve the recommendation accuracy for Diginetica. The results demonstrate that the incoming graph and the outgoing graph play different role in modeling the session representation, and their performances associate with the recommendation scenario.

Compared with the IC-SR and OC-SR, our model shows the best results for recommendation. The results confirm the usefulness of adaptively selecting the information from the different sources and the effectiveness of item-self information for better recommendation.

4.4. Effect of different session embeddings

In this part, we construct two frameworks, SEFU-L and SEFU-G, to explore the roles of local embedding and global embedding for modeling session representation. SEFU-L models the session representation only with the incoming and outgoing local embeddings in each encoder. SEFU-G models the session representation only with the incoming and outgoing global embeddings in each encoder. Their recommendation performances are shown in Fig. 4.

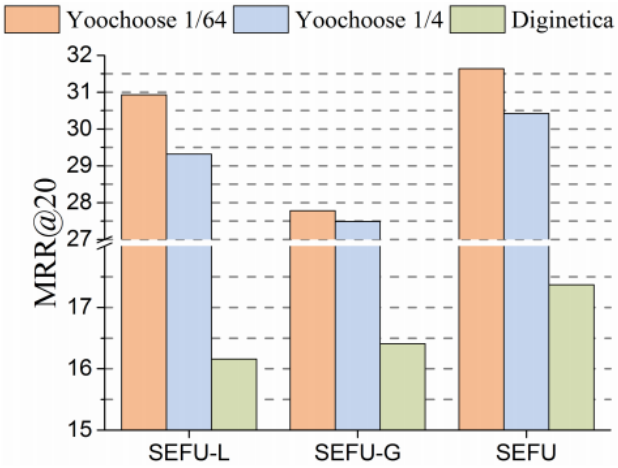

Fig. 4. Effect of different session embeddings for SR

From the Figs., the results of SEFU-L are higher than SEFU-G on Yoochoose dataset, which means that the last clicked item has strong relation with the next interactive item, and the local embedding is consistent with the user's preference. Conversely, SEFU-G shows better than SEFU-L on Diginetica dataset, which shows that inherent dependency among previous clicked items. Therefore, we suppose that when the last item is an unintentional or improvised click, only using local embedding may obviously cause deviation, and introducing the global embedding can correct the deviation. If one session consists of dispersed item catalogs, only using global embedding may introduce noise to mislead the prediction into a mistaken preference, and introducing the local embedding can enhance the current preference.

SEFU improves the accuracy over SEFU-L and SEFU-G on all the datasets, and the performances accord with the approaches which are not involved in item-self information and separated incoming and outgoing graph with fusion gating mechanism. The results indicate that SEFU improves the session representation from the nature of the problem.

4.5. Effect of aggregation operations

We use different aggregation operations to build the final session representation from the incoming session representation and the outgoing session representation. The details are as follows.

Max pooling and average pooling respectively use maximum value and average value of each dimension of the two session representations as the final session representation, namely:

$$\boldsymbol{s}^{final} = max(\boldsymbol{s}^{in}, \boldsymbol{s}^{out}) \tag{17}$$

$$\boldsymbol{s}^{final} = \frac{1}{2}(\boldsymbol{s}^{in} + \boldsymbol{s}^{out}) \tag{18}$$

For the concatenation, the final session representation is designed as:

$$\boldsymbol{s}^{out} = [\boldsymbol{s}_l^{out} : \boldsymbol{s}_g^{out}] \tag{19}$$

From Table 3, we can see that fusion gating mechanism outperforms the others. This demonstrates that fusion gating mechanism works better in modeling session representation from ISE and OSE. Additionally, the results support the thought of our work and indicate the incoming graph and outgoing graph play different role in determining the representation of the session. In most cases, the aggregation operation with max-pooling achieves better performance than the aggregation operation with average-pooling or concatenation. This indicates that max-pooling has an advantage over average-pooling and concatenation in modeling the final session representation. We think the reason may be the average-pooling and concatenation are not capable of selecting meaningful information from ISE

and OSE, namely the effective information cannot be distinguished. And introducing the noise information to meaningful information reduces the effectiveness of meaningful information.

Table 3

Effect of different aggregation operations for incoming and outgoing session representation

Models	Yoochoose 1/64		Yoochoose 1/4		Diginetica	
	Recall@20	MRR@20	Recall@20	MRR@20	Recall@20	MRR@20
Max-pooling	70.49	30.63	70.71	30.22	51.42	17.29
Average-pooling	70.05	30.24	70.49	30.17	51.29	17.13
Concatenation	70.03	30.30	70.35	30.14	51.21	17.11
Fusion gating	**71.44**	**31.64**	**70.95**	**30.42**	**51.56**	**17.37**

4.6. Effect of session length

To explore the scalability of the recommendation performance with different session lengths, there are Short and Long groups. For Short group and Long group, the separation point of Yoochoose 1/64 and Diginetica are 5 and 4. From Table 4, we can observe that: (1) compared with SR-GNN, OC-SR performs better on Short group and IC-SR performs better on Long group. This indicates that short session representations and long session representations relay on the incoming graph and the outgoing graph heavily, respectively. For short sessions, the possibility of user repeatedly browsing the same item is relatively smaller and the interaction items show a progressive relationship. For long sessions, the user may struggle for a fraction of items browsed repeatedly and the interaction items show a comparison relationship. Therefore, incoming graph and outgoing graph play different role in modeling session representation. (2) It demonstrates that separately encoding the incoming graph with item-self information and the outgoing graph with item-self information helps to learn a more precise node representation, and adaptively selecting the different sources of the session representations promotes the precise of the final session representation. (3) SEFU performs consistently better than SR-GNN within both short and long sessions, which indicates generation of our proposed main thought.

Table 4

Comparison of different session lengths

Models	Yoochoose 1/64		Diginetica	
	Short	Long	Short	Long
SR-GNN	27934	10927	21991	9102
IC-SR	28137	10936	22082	9014
OC-SR	28115	10959	21869	9176
SEFU	**28427**	**11210**	**22117**	**9215**

5. Conclusions

We developed a novel framework SEFU to model different connections in session graph for SR. We conduct thorough empirical experiments on the public datasets to investigate SEFU. Specifically, the experimental results of the model performance comparison, different connections for modeling the final session representation for SR, different session embeddings for SR, different aggregation operations for incoming and outgoing session representation, and the comparison of different session lengths fully demonstrate the effectiveness of our model. Experimental results show that our model outperforms existing methods on the three public datasets. In addition, the ablation study validated the validity of each component of our model. In addition, the ablation studies validate the validity of each component of our model.

REFERENCES

[1]. J. Song, H. Shen, Z. Ou, J. Zhang, T. Xiao, and S. Liang, "Islf: Interest shift and latent factors combination model for session-based recommendation", in Proceedings of the 28th International Joint Conference on Artificial Intelligence, pp. 5765-5771, 2019a.

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

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

[4]. C. Xu, P. Zhao, Y. Liu, V. S. Sheng, J. Xu, F. Zhuang, J. Fang, and X. Zhou, "Graph contextualized self-attention network for session-base recommendation," in Proceedings of the 28th International Joint Conference on Artificial Intelligence, pp. 3940–3946, 2019.

[5]. S. Rendle, "Factorization machines with libfm," ACM Transactions on Intelligent Systems and Technology, vol. 3, no. 3, pp. 57:1–57:22, 2012.

[6]. Z. Pan, F. Cai, Y. Ling, and M. de Rijke, "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, 2020.

[7]. A. Li, Z. Cheng, F. Liu, et al. "Disentangled graph neural networks for session-based recommendation," arXiv preprint arXiv:2201.03482, 2022.

[8]. L. Xia, C. Huang, C. Zhang. "Self-supervised hypergraph transformer for recommender systems," in Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2100-2109,2022.

[9]. W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang, "Session-based social recommendation via dynamic graph attention networks," in Proceedings of the 12th ACM International Conference on Web Search and Data Mining, pp. 555–563, 2019b.

[10]. S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in Proceedings of the 19th International Conference on World Wide Web, pp. 811–820, 2010.

[11]. J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, et al., "The youtube video recommendation system," in Proceedings of the 4th ACM conference on Recommender systems, pp. 293–296, 2010.

[12]. B. Hidasi and A. Karatzoglou, "Recurrent neural networks with top-k gains for session-based recommendations," in Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 843–852, 2018.