Discovering proper neighbors to improve session-based recommendation

Lin Liu[1][0000−0002−3657−7800], Li Wang✉[1][0000−0002−7385−1426], and Tao Lian[1][0000−0002−8941−5143]

Data Science College, Taiyuan University of Technology, Jinzhong, Shanxi, 030600, China wangli@tyut.edu.cn

Abstract. Session-based recommendation shows increasing importance in E-commerce, news and multimedia applications. Its main challenge is to predict next item just using a short anonymous behavior sequence. Some works introduce other close similar sessions as complementary to help recommendation. But users' online behaviors are diverse and very similar sessions are always rare, so the information provided by such similar sessions is limited. In fact, if we observe the data at the high level of coarse granularity, we will find that they may present certain regularity of content and patterns. The selection of close neighborhood sessions at tag level can solve the problem of data sparsity and improve the quality of recommendation. Therefore, we propose a novel model CoKnow that is a collaborative knowledge-aware session-based recommendation model. In this model, we establish a tag-based neighbor selection mechanism. Specifically, CoKnow contains two modules: Current session modeling with item tag(Cu-tag) and Neighbor session modeling with item tag (Ne-tag). In Cu-tag, we construct an item graph and a tag graph based on current session, and use graph neural networks to learn the representations of items and tags. In Ne-tag, a memory matrix is used to store the representations of neighborhood sessions with tag information, and then we integrate these representations according to their similarity with current session to get the output. Finally, the outputs of these two modules are combined to obtain the final representation of session for recommendation. Extensive experiments on real-world datasets show that our proposed model outperforms other state-of-the-art methods consistently.

Keywords: Neighborhood sessions · Tag sequence graph · Memory network · Graph neural network · Session-based recommendation.

1 Introduction

Session-based recommendation has become more and more popular in the field of recommendation systems, which aims to predict the next behavior based on anonymous user's behavior sequence in a short time. At the beginning of the research, item-to-item recommendations[1] are mainly used for this task. Since this method ignores the sequential information that plays a necessary role in the session-based recommendation, Markov chain-based methods follow to predict

the user's next behavior based on the previous one[2]. But this kind of method only models local sequential behavior, and when we try to include all sequences of the user, the state space will quickly become unmanageable.

With the development of deep learning, recurrent neural networks which are good at processing sequential data, gradually become active in session-based recommendation[3–5]. Hidasi et al.[3] apply Gated Recurrent Units (GRUs) to model behavior sequences. Li et al.[4] combines attention mechanism with RNN, while capturing the sequential behavior characteristics and user's main purpose. Although RNN-based methods can capture the sequential dependencies among behaviors in a session, it is difficult to model complex transition patterns in a session. To solve this problem, researchers begin to use graph neural networks for session-based recommendation. Wu et al.[6] use graph neural networks to model session sequences to capture complex transition mode between behaviors. On this basis, Liu et al.[7] introduce category-level information to enhance the expressive ability of the session. In addition, memory networks[8] have also been used in recommendation tasks to store the information of neighborhood sessions. Wang et al.[9] propose CSRM which applies memory network to store the representations of neighborhood sessions. However, this method only models from item level, the sparseness of neighbor information leads to biases in predictions.

Figure 1 shows an example of three sessions. On the one hand, if we analyze the sequence from the item level, we can find that the three sessions contain five, four and five completely different items respectively. The unique ID of each item makes the session sequence more confusing and complicated, which makes it difficult for us to dig out the user's real preferences. From the tag level, it can be found that these three sessions actually correspond to two, three and two types of items respectively, which greatly simplifies our judgment of user's preferences. On the other hand, looking at the three session sequences at the tag level, we can observe that each user has the most clicks on the phone. In other words, the behavior patterns of these three users are very similar. Therefore, we can refer to the first two sessions to find the real intent of the current session(session 3) when predicting the next action of current session. Combined with the information of neighborhood sessions with item tags, we can determine that the current user may want to buy a mobile phone. All brands and models may be selected by this user, rather than just limited to the products that appeared in neighborhood sessions. This approach alleviates the sparsity of neighbor information to a certain extent.

In this paper, we propose a model that improves session-based recommendation performance by discovering proper neighbors, namely CoKnow. We create Cu-tag module and Ne-tag module combined with tag information to model the current session and neighbor session respectively. The main contributions of our work can be summarized as follows:

(1) We establish a tag-aware neighborhood session selection mechanism to assist us in session-based recommendation. This method effectively reduces the sparsity of neighbor information.

(2) The expression of our current session is to integrate the information of

the neighborhood session. Both the current session and neighborhood sessions are modeled in consideration of both the item level and the tag level, and the graph neural networks are used to capture complex transfer relationships.

(3) Extensive experiments conducted on real-world datasets indicate that CoKnow evidently outperforms the state-of-art methods.

Fig. 1. An example of session sequences.

2 Related Work

2.1 Collaborative filtering-based Methods

Collaborative filtering-based recommendation is to discover user's preferences through the mining of user's historical behavior data and group users based on different preferences to recommend items with similar tastes. And it is mainly based on the following assumption: users with similar choices tend to have similar preferences. Recommendation based on collaborative filtering contains memory-based collaborative filtering and model-based collaborative filtering.

Memory-based collaborative filtering aims to find similar users or items by using a specific similarity function to generate recommendations. Such algorithms include user-based recommendations and item-based recommendations. User-based collaborative filtering calculates the similarity between users by analyzing their behaviors, and then recommendation items that similar users like. Jin et al.[10] present an optimization algorithm to automatically compute the weights for different items based on their ratings from training users. Item-based collaborative filtering method calculates the similarity between items, and recommends items that are more similar to the items the user likes. Sarwar et al.[1] analyze different item-based recommendation methods and different techniques of similarity calculation. Model-based collaborative filtering is to train a model

with the help of technology such as machine learning, and then predict the target user's score for the item. Wu et al.[11] generalize collaborative filtering models by integrating a user-specific bias into an auto-encoder. He et al.[12] propose to leverage a multi-layer perceptron to learn the user-item interaction function. Wang et al.[9] propose CSRM to model collaborative filtering in session-based recommendation, using GRU to model the current session, and combining the collaborative information of neighborhood sessions to achieve gratifying results in session-based recommendation. However, the neighbor selection method based on item similarity adopted in this method obtains sparse neighbor information. Therefore, the sparsity of neighbor selection is still a challenge.

2.2 Graph neural networks-based Methods

At present, graph neural networks[13] are widely used because of their strong ability to simulate relationships between different objects. Wu et al.[6] first construct each historical session sequence as a directed graph and use Grated Neural Networks to capture the complex transition mode among behaviors, which achieves better performance than RNN-based methods. Similarly, Xu et al.[14] use a multi-layer self-attention network to capture the global dependencies between items in a session. Qiu et al.[15] utilize graph neural networks to explore the inherent item transition patterns in a session, which is more than plain chronological order. Wang et al.[16] propose a multi-relational graph neural network model which models sessions using other types of user behavior beyond clicks. Liu et al.[7] introduce category-level representations and utilize graph neural networks to receive information at item level and category level. In a word, graph neural networks are good at modeling complex transfer relationships, but the above methods only simulate the behavior of the current session. In many cases, the information provided by the current session is far from enough.

2.3 Memory Network-based Methods

Recently, memory networks for recommendation systems have received much attention because they achieve some good performance. Chen et al.[17] propose a memory-augmented neural network (MANN) integrated with the insights of collaborative filtering for recommendation. Huang et al.[18] propose a knowledge enhanced sequential recommendation method which integrates the RNN-based networks with Key-Value Memory Network (KV-MN). CSRM[9] utilizes memory networks to save the information of neighborhood sessions. These methods(such as CSRM) have improved recommendation performance through collaborative information. But it still has inherent limitations in dealing with the sparsity of neighbor information. Therefore, we try to use more coarse-grained tag information to make up for this defect.

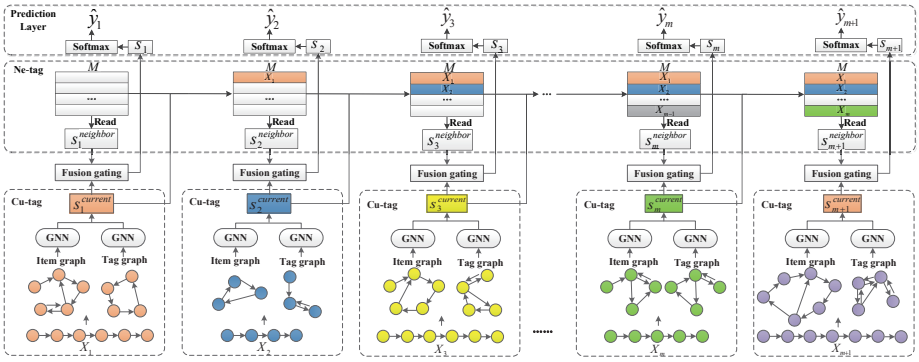

Fig. 2. The framework of CoKnow.

3 The Proposed Method: CoKnow

In this article, we propose CoKnow, which is a method to discover suitable neighbors with the help of tag information to improve session-based recommendation. The model framework is shown in Fig. 2. Specifically, the model contains two modules: Cu-tag and Ne-tag, modeling the current session and neighbor sessions respectively. In this section, we introduce these two modules in detail.

3.1 Problem Definition

Session-based recommendation aims to predict the next item that user will interact with, only based on user's anonymous interactive sequence. Here, we give some notations related to the task of session-based recommendations. Let $V = \{v_1, v_2, \ldots, v_M\}$ denote the set of all unique items involved in all sessions, $C = \{c_1, c_2, \ldots, c_N\}$ denote the set of item tag, where M and N represent the number of items and tags in the dataset respectively, and each item $v_i \in V$ corresponds to a tag $c_k \in C$. We define the session set in a dataset as $X = \{X_1, X_2, \ldots, X_Q\}$, where Q is the number of session sequences. And each session sequence X_i consists of two sequences: item sequence $S_i^m = [v_1, v_2, \ldots, v_n]$ and tag sequence $S_i^c = [c_1, c_2, \ldots, c_n]$. The goal of session-based recommendation is to calculate the recommendation probability $\widehat{y}_i$ of each candidate item, and get a list of probability $\widehat{y} = \{\widehat{y}_1, \widehat{y}_2, \ldots, \widehat{y}_M\}$.

3.2 Current session modeling with item tag (Cu-tag)

To model the current session, we jointly model from the item level and the tag level to get the representation of the current session, shown as Fig.3. And the modeling of these two levels uses graph neural network for the same processing, so here is a unified description.

Learning the embedding of nodes in the graph. Each item sequence $S_i^m = [v_1, v_2, \ldots, v_n]$ and its corresponding tag sequence $S_i^c = [c_1, c_2, \ldots, c_n]$ can

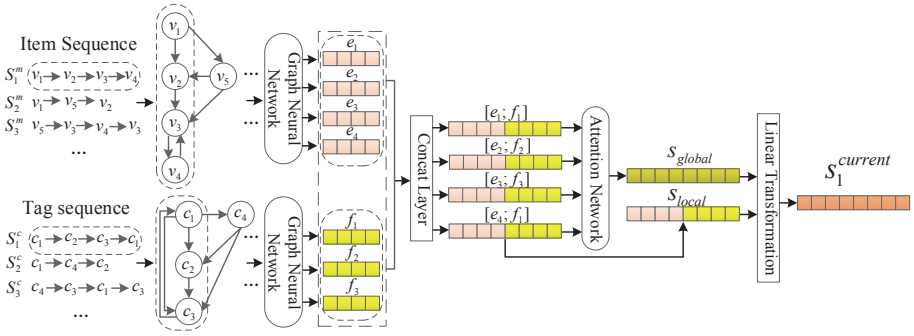

Fig. 3. The framework of the Cu-tag module.

be regarded as graph structure data, and they can be constructed as directed graphs. Here, we use $\mathcal{G}_m = (\nu_m, \varepsilon_m)$ to represent the directed graph constructed by the item sequence, each node refers to an item $v_i \in V$. And $\mathcal{G}_c = (\nu_c, \varepsilon_c)$ represents the directed graph constructed by the tag sequence, each node refers to a tag $c_i \in C$. For example, the item sequence is $S_i^m = [v_1, v_2, v_3, v_4, v_3, v_5]$ and the corresponding tag sequence is $S_i^c = [c_1, c_2, c_1, c_3, c_1, c_4]$. The constructed directed graph and connection matrix are Fig.4(a), Fig.4(b) and Fig.4(c), Fig.4(d). It should be noted that due to the repeated occurrence of a certain item or tag in an sequence, we assign a normalized weight to each edge, which is calculated as the occurrence of the edge divided by the outdegree of that edge's start node.

After constructing the direct graph and connection matrix, we obtain latent vectors of nodes via graph neural networks. At first, we embed every node into a united embedding space. The vector x_i denotes the latent vector of node learned via graph neural networks. The update functions are given as follows:

$$p^t = Concat(A_{In}^i([x_1^{t-1}, \ldots, x_n^{t-1}]^\top H_{In} + b_{In}), A_{Out}^i([x_1^{t-1}, \ldots, x_n^{t-1}]^\top H_{Out} + b_{Out})) \tag{1}$$

$$z^t = \sigma(W_z p^t + U_z x_i^{t-1}) \tag{2}$$

$$r^t = \sigma(W_r p^t + U_r x_i^{t-1}) \tag{3}$$

$$\widetilde{x_i^t} = tanh(W_o p^t + U_o(r^t \odot x_i^{t-1})) \tag{4}$$

$$x_i^t = (1 - z^t) \odot x_i^{t-1} + z^t \odot \widetilde{x_i^t} \tag{5}$$

where W_z, W_r, $W_o \in \mathbb{R}^{2d \times d}$, U_z, U_r, $U_o \in \mathbb{R}^{d \times d}$, H_{In}, $H_{Out} \in \mathbb{R}^{d \times d}$ are learnable parameters. b_{In}, $b_{Out} \in \mathbb{R}^d$ are the bias vectors. A_{In}^i, $A_{Out}^i \in \mathbb{R}^{1 \times n}$ are the corresponding rows in the matrices A_{In} and A_{Out}. $\sigma(\cdot)$ denotes the sigmoid function and $\odot$ represents element-wise multiplication. z^t, r^t are update gate and reset gate respectively, which decide what information to be preserved and discarded. When we update nodes in item graph and tag graph in this way, the obtained item and tag representations are expressed as e_i and f_i.

Current session embedding with tag information. The representation of

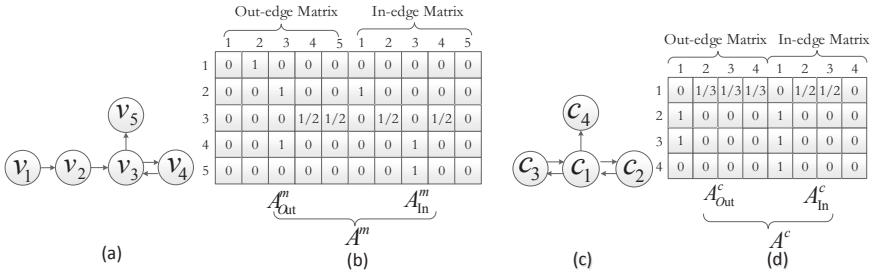

Fig. 4. Examples of item graph, tag graph and their connection matrices.

the current session embedding consists of local representation and global representation. In our model, we suppose that the last interacting item plays a key role, so we use the representation of the last interacting item with tag information as local representation of the current session, i.e. $s_{local} = [e_n; f_n]$.

For global representation, considering the importance of each item in the session sequence is different, so we combine the attention mechanism to get a global representation of the session:

$$\alpha_i = r^\top \sigma(W_1[e_n; f_n] + W_2[e_i; f_i] + c) \tag{6}$$

$$s_{global} = \sum_{i=1}^{n} \alpha_i [e_i; f_i] \tag{7}$$

where $r \in \mathbb{R}^d$ and W_1, $W_2 \in \mathbb{R}^{d \times 2d}$ are learnable parameters.

Finally, we compute the current session embedding $s_i^{current}$ by taking linear transformation over the concatenation of the local embedding and global embedding:

$$s_i^{current} = W_3[s_{local}; s_{global}] \tag{8}$$

where $W_3 \in \mathbb{R}^{d \times 4d}$ is weight matrix.

3.3 Neighbor session modeling with item tag (Ne-tag)

The Cu-tag module only utilizes the information contained in the current session, which ignores the collaborative information in the neighborhood sessions that can supplement the expression of the current session. To address this omission, we design a neighbor session modeling method with tag participation. The Ne-tag module combines tag information to select neighborhood sessions that have the most similar behavior pattern to the current session and use them as auxiliary information of the current session, so as to better judge the user preference of the current session.

After using the Cu-tag module to obtain the representations of m last sessions, they are stored in the memory matrix M. Then we calculate the similarity

between the global embedding s_{global} of the current session and each session m_i stored in the memory matrix M:

$$sim(s_{global}, m_i) = \frac{s_{global} \cdot m_i}{||s_{global}|| \times ||m_i||} \quad \forall m_i \in M \tag{9}$$

According to similarity values, we can get the k largest similarity values $[sim_1, sim_2, \ldots, sim_{k-1}, sim_k]$ and the corresponding k representations of neighborhood sessions $[m_1, m_2, \ldots, m_{k-1}, m_k]$. Since each neighborhood session contributes differently to the representation of current session, we utilize the softmax function to process the k similarity values, and more similar sessions will get greater weight. Finally, the ultimate neighborhood session embedding is obtained by integrating the information of k neighborhood sessions:

$$w_i = \frac{exp(\beta sim_i)}{\sum_{j=1} exp(\beta sim_j)} \tag{10}$$

$$s_i^{neighbor} = \sum_{i=1}^{k} w_i m_i \tag{11}$$

where β denotes the strength parameter.

3.4 Prediction Layer

Here, we apply a fusion gating mechanism to selectively combine the information of the current session and the neighborhood sessions to obtain the ultimate expression of the current session. The fusion gate g_i is given by:

$$g_i = \sigma(W_l s_{local} + W_g s_{global} + W_o s_i^{neighbor}) \tag{12}$$

Therefore, we can calculate the representation of the current session combined with the collaborative information of the neighborhood sessions:

$$s_i = g_i s_i^{current} + (1 - g_i) s_i^{neighbor} \tag{13}$$

Then, we compute the recommendation probability $\widehat{y}$ by softmax function for each candidate item $v_i \in V$. The addition of tag information and neighbor information is beneficial to improve the score of candidate items that belong to same or similar tags as the items in the current session. The calculation of the recommendation probability is as follows:

$$\widehat{y} = softmax(s_i^{\top}[e_i; f_i]) \tag{14}$$

We use cross-entropy loss as our loss function, which can be written as follows:

$$L(\widehat{y}) = -\sum_{i=1}^{M} y_i log(\widehat{y}_i) + (1 - y_i) log(1 - \widehat{y}_i) + \lambda \parallel \theta \parallel^2 \tag{15}$$

where y is the one-hot encoding vector of the ground truth item, θ is the set of all learnable parameters.

Algorithm 1 elaborates the construction process of Cu-tag and Ne-tag in our proposed CoKnow.

Algorithm 1 CoKnow

Input: session sequence X_i, which contains item sequence S_i^m, tag sequence S_i^c; connection matrix A^m, A^c

Output: the probability list $\widehat{y} = \{\widehat{y}_1, \widehat{y}_2, \ldots, \widehat{y}_M\}$

Construct Cu-tag module:
for $S_i^m, S_i^c \in X_i$ **do**
    use (S_i^m, A^m) and (S_i^c, A^c) to construct item graph and tag graph
    obtain the representation of nodes e_i^t and f_i^t in the item graph and tag graph respectively using GNN based on Eq.(1)-(5)
    concatenate e_n and f_n to obtain the local embedding s_{local}
    obtain global embedding s_{global} based on attention mechanism(Eq. (6)-(8))
    combine s_{local} with s_{global} to obtain the current session embedding $s_i^{current}$
    store $s_i^{current}$ in the outer memory matrix M
return
Construct Ne-tag module:
for $m_i \in M$ **do**
    calculate the similarity between s_{global} and m_i
    obtain the neighborhood session embedding $s_i^{neighbor}$ based on Eq.(9)-(11)
return
combine $s_i^{current}$ with $s_i^{neighbor}$ to obtain the ultimate current session embedding s_i (Eq.(12)-(13))
compute the probability $\widehat{y}$ based on Eq.(14)

4 Experiments

4.1 Research Questions

In this section, we will answer the following research questions:

(RQ1) How is the performance of CoKnow compared with other state-of-the-art methods?

(RQ2) How does CoKnow perform on sessions with different lengths?

(RQ3) Does the existence of each module in CoKnow has some significance?

(RQ4) Will the prediction performance of CoKnow be influenced when the number of neighborhood sessions is different?

4.2 Datasets

We evaluate the proposed model on the following two real e-commerce datasets:

• **Cosmetics**[1]**:** This is a user behavior record of a medium cosmetics online store in October and November 2019. It is a public dataset published on kaggle competition platform. Each record contains attribute information, such as session_id, item_id, item_category and timestamp. It is worth mentioning that we are first to try to perform a session-based recommendation task on this dataset.

• **UserBehavior**[2]**:** A Taobao user behavior dataset provided by Alimama.

[1] https://www.kaggle.com/mkechinov/ecommerce-events-history-in-cosmetics-shop
[2] https://tianchi.aliyun.com/dataset/dataDetail?dataId=649userId=1

It contains a series of purchase behavior records of users during November 25 to December 03, 2017. Each record contains attribute information, such as userID, itemID, item_category and timestamp. The userID is directly used as session_id.

In this paper, we filter out all session sequences with a length shorter than

Table 1. Statistics of datasets used in the experiments.

Statistics	Cosmetics		UserBehavior
	2019-Oct	2019-Nov	
# of training sessions	33087	20127	2816
# of test sessions	6322	4765	706
# of items	22063	16175	14641
# of categories	103	98	1403
Average length	6.9938	9.1128	18.6063

2 or longer than 50 items[5]and items appearing less than 5 times. In addition, similar to[19], we use the data augmentation method to generate sequences and corresponding labels by splitting the input session. Then, a session sequence of length n is divided into $n-1$ sub-session sequences. For an input item sequence $S_i^m = [v_1, v_2, \ldots, v_n]$, we generate the sequences and corresponding labels $([v_1], v_2)$, $([v_1, v_2], v_3)$,. . . . . , $([v_1, v_2, \ldots, v_{n-1}], v_n)$ for training and testing on all datasets, where $[v_1, v_2, \ldots, v_{n-1}]$ is the generated sequence and v_n is the label of the sequence, i.e. the next interactive item. Similarly, the same processing is performed for the tag sequence. The data statistics are shown in Table 1.

4.3 Baselines

To evaluate the prediction performance, we compare CoKnow with the following representative baselines:

- **Pop:** Pop always recommends the most popular items in the training set. It is a simple method that still performs well in some scenarios.
- **Item-KNN[1]:** A baseline method based on the cosine similarity to recommend items that are most similar to the candidate item within the session.
- **FPMC[20]:** This is a method that combines Markov chain model and matrix factorization for the next-basket recommendation. Here, we ignore user latent representations to make it suitable for session-based recommendation.
- **GRU4Rec-topK[21]:** Based on deep learning, this method uses RNN to model the user's interaction sequence. And it improved GRU4Rec with a top-K based ranking loss.
- **NARM[4]:** This model combines the attention mechanism with RNN, while capturing sequential behavior characteristics and main purpose of users.
- **STAMP[22]:** STAMP not only considers the general interest from long-term historical behavior, but also considers the user's last click to mine the immediate interest.
- **NEXTITNET[23]:** It is a CNN-based generative model. In this model, a stack of dilated convolutional layers is applied to increase the receptive field

when modeling long-range sequences.

• **SR-GNN[6]:** A recently proposed algorithm for session-based recommendation applying graph neural network. It constructs the item sequence as a directed graph, and combines attention mechanism to obtain the embedding of each session sequence.

• **CSRM[9]:** This is the model most relevant to our model. It utilizes GRU to model the current session at the item level, and applies an outer memory module to model neighborhood sessions.

• **CaSe4SR[7]:** It is a deep learning model that utilizes graph neural network to model item sequence and category sequence respectively. The difference from the model in this paper is that it only models the current session and does not consider collaborative information.

4.4 Evaluation Metrics and Experimental Setup

Evaluation Metrics. We use the most commonly used metrics Recall@20 and MRR@20 for session-based recommendation to evaluate model performance. Recall@20 is the proportion of ground-truth items appearing in the top-20 positions of the recommendation list. It does not consider the rank of the item that user actually clicked. MRR@20 is the average of the inverse of the ground-truth item ranking. If the rank of an item is greater than 20, the value is set to 0. This indicator takes the position of the item in the recommendation list into account, and is usually important in some order-sensitive tasks.

Experimental Setup. During training, we randomly initialize all parameters with a Gaussian distribution with a mean of 0 and a standard deviation of 0.1. The dimension of the embedding vector is set to $d = 100$. The mini-batch Adam optimizer is exerted to optimize these parameters, where the initial learning rate is set to 0.001. In addition, the training batch size is set to 512 and the L2 penalty is 10^5. We vary the number of neighborhood sessions from $[128, 256, 512]$ to study the effects of Ne-tag.

4.5 Results and Analysis

Comparison with baseline methods(RQ1). To demonstrate the performance of the proposed model, we compare CoKnow with other state-of-art session-based recommendation methods and the results can be seen in Table 2. It shows that CoKnow consistently achieves the best performance in terms of both Recall@20 and MRR@20 on two datasets. Overall, the recommendation performance of all methods on UserBehavior is low. This may because the time interval in the sessions on this dataset is longer than other datasets, and as can be seen from Table 1, there are more item categories in this dataset, which means that the user's behavior is more complicated, making it harder to judge user's real intentions. Our model has been greatly improved compared to other models, which also confirms the ability of CoKnow to handle complex behavior sequences. In conventional methods, Item-KNN achieves some improvement over Pop which only considers the popularity of the items, which shows

Table 2. The performance comparison of CoKnow with other baseline methods over two datasets.

Method	Cosmetics				UserBehavior	
	2019-Oct		2019-Nov			
	Recall@20	MRR@20	Recall@20	MRR@20	Recall@20	MRR@20
Pop	0.0420	0.0104	0.0470	0.0109	0.0148	0.0038
Item-KNN	0.1519	0.0601	0.1657	0.0659	0.0428	0.0143
FPMC	0.2422	0.1877	0.2092	0.1623	0.0711	0.0478
GRU4Rec-topK	0.4007	0.2447	0.4048	0.2210	0.0855	0.0318
NARM	0.4103	0.2271	0.4149	0.2141	0.0806	0.0328
STAMP	0.3738	0.1967	0.4148	0.2294	0.0621	0.0240
NEXTITNET	0.3303	0.1741	0.3488	0.1639	0.0353	0.0164
SR-GNN	0.3325	0.1655	0.3377	0.1610	0.0481	0.0181
CSRM	0.4515	0.2530	0.4367	0.2283	0.0906	0.0431
CaSe4SR	0.4396	0.2556	0.4551	0.2585	0.1865	0.0880
CoKnow(ours)	**0.4558**	**0.2669**	**0.4669**	**0.2694**	**0.1989**	**0.1094**

that considering the similarity of items in the session can improve the accuracy rate. The biggest difference among FPMC and the above two models is that it models user's historical behavior records. Experimental results show the contribution of sequential information to recommendation performance. Furthermore, deep learning-based methods generally outperform the conventional algorithms. GRU4Rec-topK and NARM use recurrent units to capture the general interests of users and have achieved more prominent prediction results, which indicates the effectiveness of RNN in sequence modeling. STAMP also obtains better results by combining general interests with current interests(the last-clicked item), which indicates the importance of the last click. In comparison, the prediction performance of NEXTITNET based on CNN is still slightly worse, indicating that CNN is still not as capable of capturing sequential information as RNN on our dataset. CSRM receives higher recommendation performance than SR-GNN, reflecting the role of neighborhood sessions. Looking at these baseline methods, CaSe4SR is the best recommendation method, which fully shows the effectiveness of category information for session-based recommendation.

Analysis on sessions with different lengths(RQ2). In addition to verifying the prediction ability of our model on all sessions, we also group sessions in the datasets according to the length to verify the validity of our model. Specifically, we divide each dataset into three groups based on the average length of sessions in datasets in Table 1, that is, the length is less than or equal to 5, the length is less than or equal to 15 and the length is greater than 15. We named these three groups "Short", "Medium" and "Long" respectively. It is worth noting that because the number of sessions in the "Short" group of UserBehavior is too small, we have not conducted experiments on this group of UserBehavior here.

It is obvious that as the length of session increases, the performance on all two datasets in terms of Recall@20 and MRR@20 decreases to varying degrees. This may be because as the length of the session increases, the number of items that users click on out of curiosity or accident will increase, resulting in an increase in irrelevant items in the session sequence, which is not conducive to the extraction of user preferences. Compared with other groups, the prediction performance of

Table 3. The performance of different methods on sessions with different lengths.

Method		Cosmetics				UserBehavior	
		2019-Oct		2019-Nov			
		Recall@20	MRR@20	Recall@20	MRR@20	Recall@20	MRR@20
Short	GRU4Rec-topK	0.4898	0.3518	0.5066	0.2879	-	-
	NARM	0.6207	0.4826	0.5788	0.3832	-	-
	STAMP	0.5462	0.4149	0.3859	0.2400	-	-
	NEXTITNET	0.4653	0.3272	0.4673	0.2819	-	-
	SR-GNN	0.5396	0.3919	0.5110	0.3333	-	-
	CSRM	0.5962	0.4294	0.5685	0.3789	-	-
	CaSe4SR	**0.6356**	0.5049	0.5916	**0.4274**	-	-
	CoKnow(ours)	0.6223	**0.5161**	**0.6022**	0.4239	-	-
Medium	GRU4Rec-topK	0.3481	0.1491	0.3835	0.1753	0.1576	0.0733
	NARM	0.4012	0.2092	0.4342	0.2305	0.2342	0.1179
	STAMP	0.2889	0.1412	0.3422	0.1797	0.1171	0.0501
	NEXTITNET	0.2462	0.1175	0.2658	0.1229	0.0563	0.0363
	SR-GNN	0.3599	0.1813	0.3917	0.2017	0.1136	0.0455
	CSRM	0.3978	0.2091	0.4215	0.2246	0.2036	0.0815
	CaSe4SR	0.4130	0.2360	0.4438	0.2475	0.2827	0.1309
	CoKnow(ours)	**0.4192**	**0.2677**	**0.4441**	**0.2874**	**0.3092**	**0.1476**
Long	GRU4Rec-topK	0.3155	0.1412	0.3268	0.1584	0.0826	0.0293
	NARM	0.2425	0.0948	0.2802	0.1182	0.0467	0.0190
	STAMP	0.2694	0.1134	0.2939	0.1331	0.0479	0.0173
	NEXTITNET	0.2233	0.0864	0.2420	0.0966	0.0251	0.0125
	SR-GNN	0.2341	0.0946	0.2525	0.1033	0.0682	0.0255
	CSRM	0.3085	0.1358	0.3297	0.1545	0.0846	0.0364
	CaSe4SR	0.3503	0.1897	0.3707	0.2045	0.1742	0.0769
	CoKnow(ours)	**0.3666**	**0.2133**	**0.3788**	**0.2268**	**0.1815**	**0.0932**

all models on short sessions is much better than other groups. In "Short" group, NARM achieves the best performance among RNN-based baseline methods on all datasets, but as the length of the session increasing, its performance drops quickly. This may explain that RNN has difficulty coping with long sessions. SR-GNN and CSRM have also achieved good recommendations on the two datasets. This shows the role of graph neural networks and neighborhood information in capturing user preferences. The performance of CaSe4SR is more prominent.

According to the results, we can observe that our model performs best on different groups of all datasets, which fully proves that the information of tag and neighbors is an effective complement to the capture of user's intention in long and short sessions. Although the result on CaSe4SR in short sessions slightly exceeds CoKnow, this may be because the model cannot accurately capture user's preferences due to too little behavior, and the joining of neighborhood sessions has misled the current session.

Impact of different modules(RQ3). In order to illustrate more clearly the validity of each party of our model, we compare CoKnow with its three variants: (1) CoKnow$_{cu}$: CoKnow without the Ne-tag module and only graph neural networks are used to model the current session. It is actually CaSe4SR. (2) CoKnow$_{ne}$: CoKnow without the Cu-tag module which only uses the Ne-tag module to model neighborhood sessions. (3) CoKnow$_{item}$: CoKnow only constructs a model at item level, that is, it removes the tag information.

It can be seen from Table 4 that the prediction result of CoKnow combing

two modules is the best. CoKnow$_{cu}$ outperforms CoKnow$_{ne}$, which indicates that a session's own information is more important than collaborative information of the neighborhood sessions. Besides, the prediction effect of CoKnow$_{item}$ is slightly worse than CoKnow, which shows that the tag information can supplement the information of the session to a certain degree, so as to capture user's real intention more accurately. Especially on the more complicated UserBehavior, the prediction performance of our proposed model CoKnow has been greatly improved compared to CoKnow$_{item}$. This fully indicates that tag information can simplify the sequence with various behaviors, which is beneficial to the capture of user's intentions.

Influence of the number of neighbors k(RQ4). The number of neighborhood sessions also has a certain effect on the prediction performance of the model. Therefore, we compare the performance of the models when the number of neighbors takes different values. It can be seen from Table 5 that not all datasets are consistent with the concept that the prediction effect increases with the number of neighborhood sessions. On 2019-Nov, it is true that as the number of neighbors k increases, the values of the MRR@20 metrics are increasing. Other datasets don't exactly follow this pattern. For example, the value of Recall@20 on 2019-Oct is the best when $k = 128$. This may be because an increase in the number of neighbor sessions may bring more irrelevant information to interfere with the expression of the current session for some datasets. Considering the results of all datasets comprehensively, our model CoKnow can achieve the optimal results when $k = 128$. Therefore, in this paper, the number of neighbors is set to 128.

Table 4. Performance comparison of different variants of CoKnow.

Method	Cosmetics				UserBehavior	
	2019-Oct		2019-Nov			
	Recall@20	MRR@20	Recall@20	MRR@20	Recall@20	MRR@20
CoKnow$_{cu}$	0.4396	0.2556	0.4551	0.2585	0.1865	0.0880
CoKnow$_{ne}$	0.3385	0.1812	0.3252	0.1624	0.0656	0.0217
CoKnow$_{item}$	0.4543	0.2666	0.4655	0.2688	0.1209	0.0754
CoKnow(ours)	**0.4558**	**0.2669**	**0.4669**	**0.2694**	**0.1989**	**0.1094**

Table 5. Performance comparison of CoKnow with different number of neighborhood sessions k.

Method	Cosmetics				UserBehavior	
	2019-Oct		2019-Nov			
	Recall@20	MRR@20	Recall@20	MRR@20	Recall@20	MRR@20
$k = 128$	**0.4558**	0.2669	**0.4669**	0.2694	**0.1989**	**0.1094**
$k = 256$	0.4499	**0.2706**	0.4658	0.2714	0.1949	0.1085
$k = 512$	0.4506	0.2655	0.4665	**0.2722**	0.1940	0.1059

5 Conclusion

In this paper, we propose a method that utilizes tag information to discover proper neighbors to improve session-based recommendation. We establish Cu-tag module and Ne-tag module. The former uses graph neural networks to model the item sequence and tag sequence of the current session, and the latter utilizes memory network to store information of neighborhood sessions to supplement the current session. These two modules are combined via a fusion gating mechanism to achieve better recommendations. Extensive experiments verify that the information of tags and neighbors play a good role in the expression of current session.

The limitation of this work is that we can only consider a limited number of neighborhood sessions, which may miss some more similar neighborhood sessions and affect prediction performance. In future work, we will explore solutions to this problem.

6 Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 61872260), particularly supported by Science and Technology Innovation Project of Higher Education Institutions in Shanxi Province (No. 2020L0102).

References

1. B. Sarwar, G. Karypis, J. Konstan, J. Riedl.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web (2001)
2. Guy Shani, David Heckerman, Ronen I. Brafman.: An MDP-Based Recommender System. In: Journal of Machine Learning Research, pp. 1265-1295 (2005)
3. Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, Domonkos Tikk.: Session-based Recommendations with Recurrent Neural Networks. In: International Conference on Learning Representations 2016 (ICLR)(2016)
4. Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, Jun Ma.: Neural Attentive Session-based Recommendation. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 1419–1428 (2017)
5. Ren, Pengjie, Chen, Zhumin, Li, Jing, Ren, Zhaochun, Ma, Jun, de Rijke, Maarten.: RepeatNet: A repeat aware neural recommendation machine for session-based recommendation. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 4806–4813 (2019)
6. Wu, Shu, Tang, Yuyuan, Zhu, Yanqiao, Wang, Liang, Xie, Xing, Tan, Tieniu.: Session-based recommendation with graph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 346–353 (2019)
7. Lin Liu, L. Wang, Tao Lian.: CaSe4SR: Using category sequence graph to augment session-based recommendation. In: Knowl. Based Syst., pp. 106558 (2021)
8. Jason Weston, Sumit Chopra, Antoine Bordes.: Memory Network. In: International Conference on Learning Representations 2015 (ICLR) (2015)

9. Wang, Meirui, Ren, Pengjie, Mei, Lei, Chen, Zhumin, Ma, Jun, de Rijke, Maarten.: 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 (2019)
10. Rong Jin, J. Y. Chai, L. Si.: An automatic weighting scheme for collaborative filtering. In: SIGIR '04 (2004)
11. Y. Wu, C. DuBois, A. Zheng, M. Ester.: Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In: WSDM '16 (2016)
12. X. He, Lizi Liao, Hanwang Zhang, L. Nie, Xia Hu, Tat-Seng Chua.: Neural Collaborative Filtering. In: Proceedings of the 26th International Conference on World Wide Web (2017)
13. Qiao, J., Wang, L., Duan, L.: Sequence and graph structure co-awareness via gating mechanism and self-attention for session-based recommendation. In: International Journal of Machine Learning and Cybernetics, pp. 1-15 (2021)
14. Xu, Chengfeng, Zhao, Pengpeng, Liu, Yanchi, Sheng, Victor S, Xu, Jiajie, Zhuang, Fuzhen, et al.: Graph contextualized self-attention network for session-based recommendation. In: IJCAI, pp. 3940-3946 (2019)
15. Qiu, Ruihong, Li, Jingjing, Huang, Zi, Yin, Hongzhi.: 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 (2019)
16. Wang, Wen, Zhang, Wei, Liu, Shukai, Liu, Qi, Zhang, Bo, Lin, Leyu, et al.: Beyond clicks: Modeling multi-relational item graph for session-based target behavior prediction. In: Proceedings of The Web Conference 2020 (2020)
17. Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, et al.: Sequential Recommendation with User Memory Networks. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pp. 108–116 (2018)
18. Huang, Jin and Zhao, Wayne Xin and Dou, Hongjian and Wen, Ji-Rong and Chang, Edward Y.: Improving sequential recommendation with knowledge-enhanced memory networks. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, pp. 505–514 (2018)
19. Tan, Yong Kiam, Xu, Xinxing, Liu, Yong.: Improved recurrent neural networks for session-based recommendations. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, pp. 17–22 (2016)
20. Rendle, Steffen, Freudenthaler, Christoph, Schmidt-Thieme, Lars.: Factorizing personalized markov chains for next-basket recommendation. In: Proceedings of the 19th international conference on World wide web, pp. 811–820 (2010)
21. Hidasi, Balazs, Karatzoglou, Alexandros.: 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)
22. Liu, Qiao, Zeng, Yifu, Mokhosi, Refuoe, Zhang, Haibin.: 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 (2018)
23. Yuan, Fajie and Karatzoglou, Alexandros and Arapakis, Ioannis and Jose, Joemon M and He, Xiangnan.: A Simple Convolutional Generative Network for Next Item Recommendation. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, pp. 582-590 (2019)