Received July 12, 2019, accepted July 23, 2019, date of publication July 30, 2019, date of current version August 22, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2931945

Self-Attention Network for Session-Based Recommendation With Streaming Data Input

SHIMING SUN[1,2,3], YUANHE TANG[1,2], ZEMEI DAI[1,2], AND FU ZHOU[1,2]
[1]NARI Group Corporation (State Grid Electric Power Research Institute), Nanjing 211106, China
[2]NARI Technology Development Limited Company, Nanjing 211106, China
[3]State Key Laboratory of Smart Grid Protection and Control of NARI Group Corporation, Nanjing 211106, China

Corresponding author: Shiming Sun (sunshiming@sgepri.sgcc.com.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YF0902600, and in part by the State Grid Corporation of China through the Research on High Performance Computing Technology for Real-time Data and Network Analysis of Large Power Grid.

ABSTRACT In the current era of the rapid development of big data, it has become increasingly critical and practical to study recommender systems with streaming data input. However, the recommender system is often faced with the challenge that the history records of new users or anonymous users are not available. Specifically, session-based recommendation, which aims to predict a user's next actions, is a typical task to overcome the challenge. To capture a user's long-term preference in session-based recommendations, recurrent neural networks (RNN)-based models have been widely applied with impressive results, but the inherent sequential nature of RNNs prevents parallelism within training examples, which is critical in long sessions because memory constraints limit batching across instances. In this paper, we propose a novel method, i.e., self-attention network for session-based recommendation (SANSR), which is based on only attention mechanisms, dispensing with recurrence, and supports parallelism in the session. The proposed model attempts to find items that are relevant based on previous time steps in the ongoing session and to assign them different weights to predict the next item. The extensive experiments are conducted on two real-world datasets, and the experimental results show that our proposed model is superior to the state-of-the-art methods.

INDEX TERMS Session-based recommendation, self-attention network, streaming data.

I. INTRODUCTION

Given the vast amount of information in people's lives, how to obtain valid information from extensive data has become an urgent problem. Recommender systems can provide personalized recommendations for different users so that each user can obtain the information they desire from the limited, diverse information screened by the recommender system.

Traditional recommender systems typically utilize informative user profiles to model user preferences and recommend items that may be of interest to users. In other words, each user must have sufficient interaction records, and the user identity must be visible in each interaction event. Nevertheless, in many online systems, such as e-commerce websites and most news and media sites, the recommender system can rely on only the current session information to provide accurate recommendations because the identities of new users and unlogged users are unknown, which makes their history unavailable. To this end, session-based recommender systems have done a lot of research on how to make accurate recommendations for new users.

Previous session-based recommendations mainly adopt item-based heuristic methods to describe user preferences within the session by capturing the relationships between items. However, these methods do not take into account the entire sequence information of the session. Taking inspiration from the success of Recurrent Neural Networks (RNN) on sequence modeling, researchers have introduced this technique into session-based recommendation [1]–[3] to capture the sequential properties. RNNs have advantages over approaches based on the similarity between items, with impressive results based on the capacity to model the whole session of user interactions (views, clicks, etc.). Based on the RNN framework, the recommendation accuracy can be improved from the perspective of items and users. In the case of items, since an item itself is rich in information,

The associate editor coordinating the review of this manuscript and approving it for publication was Shirui Pan.

it is natural to add the attribute information (e.g., text and image) of the item to the RNN framework [4]. In terms of users, the recommendation accuracy is reflected mainly in the accuracy of the model in modeling user preferences. However, users' interests are diverse and drift over time. For example, a user may initially browse multiple brands of electronic cameras. After adding one electronic camera to their shopping cart, the user's interest may shift to other things that reflect the diversity of the user's interest. Therefore, attention-based RNN methods are proposed to model the sequential behavior of previous clicks and to depict users' diverse interests. Recently, NARM [2] applies an attention mechanism to model a user's sequential behavior and capture the user's main purpose in the current session. Similar to NARM, STAMP [5] captures a user's preference and current interests by combining simple MLP networks and an attention mechanism.

Although the methods above achieve satisfactory, state-of-the-art results, they have some limitations. First, the inherent sequentiality of RNN-based methods does not support parallelism, which may cause the training process to consume a substantial amount of time when the session sequence is long. Second, previous work reveals the importance of item interaction patterns, but these studies always model one-way transitions between consecutive items, neglecting the relations between other items in the session. Fortunately, the transformer, a model that is based solely on the attention mechanism called 'self-attention', thereby dispensing with recurrence and convolutions entirely, is proposed. The transformer not only enables parallelism and requires less time to train than RNN-based methods but also captures the interactions between items in the session, regardless of their distance.

The session-based recommendation can be applied in online services, where each session represents a sequential interaction. The function of self-attention to capture the global dependency of the entire sequence makes it suitable for the session-based recommendation which has sequential patterns. To this end, we argue that self-attention mechanisms can play a significant role in session-based recommendation. In this work, we first apply an embedding layer to generate input embedding matrix, and then utilize several self-attention blocks which is composed of multi-head self-attention and feedforward network to capture the interactions between items, even if the items are far from each other. Finally, we make use of the incremental training method and get the user's preference for different items in the prediction layer. Overall, the proposed method can improve the training efficiency through parallelization when facing a large amount of input data on the one hand and adaptively assign weights to previous items at each time step in the session on the other.

The rest of this paper is organized as follows. We define the problem in Section 2, and present our framework in Section 3. Then, we verify the effectiveness and efficiency of our method via experimental results in Section 4. Section 5 briefly introduces the related work. Finally, the conclusions and outlooks of this work are presented in Section 6.

II. PROBLEM STATEMENT

Session-based recommendation is the task of predicting a user's next click in current session on the basis of his/her sequential transaction data. Here, we present a formulation of the session-based recommendation problem.

Let $V = \{v_1, v_2, \ldots, v_{|V|}\}$ represent a set of unique items in all sessions and $S = \{s_1, s_2, \ldots, s_{|S|}\}$ be the session set, where $|V|$ and $|S|$ are the total numbers of unique items and sessions, respectively. For an anonymous session s_i, its sequential transaction, in time order, is represented by $s_i = \{v_{s_i,1}, v_{s_i,2}, \ldots, v_{s_i,n}\}$, where $v_{s_i,t} \in V$ denotes that an item is clicked on at time step t in this session. The goal of session-based recommendation is to predict the next click (i.e., $v_{s_i,n+1}$) for session s_i. Formally, our model aims to generate a ranked list of all candidate items that can occur in that session by predicting their click probability. The scores of all items are denoted by $\hat{y} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{|V|}\}$, where $\hat{y}_i$ corresponds to the recommendation score of item v_i. The prediction scores are ranked in descending order, and the items ranked in the top N are used for recommendation.

III. THE PROPOSED METHOD

In this section, we introduce how we build the proposed SANSR via an embedding layer, several self-attention blocks and a prediction layer. The architecture of the SANSR is shown in left part of Figure 1. First, we transform the sequential transaction of the session into a fixed-length sequence; then, we embed their elements into low-dimensional dense spaces. Next, self-attention blocks are used to capture the global dependencies between the input and the output, and the interactions between items. Then, we utilize a multiple self-attention network to learn high-level item transactions. Finally, we adopt a prediction layer to predict the next click of the session and utilize the pairwise objective function with BPR optimization criterion and the incremental training method to train the parameters of the model. In the following, we introduce each part of our model in detail.

A. EMBEDDING LAYER

For each session, we transform the sequential transaction $s_i = \{v_{s_i,1}, v_{s_i,2}, \ldots, v_{s_i,n}\}$ into a fixed-length corresponding sequence $x_i = \{x_{s_i,1}, x_{s_i,2}, \ldots, x_{s_i,l}\}$, where l represents the fixed length. If $n \geq l$, we consider the most recent l actions; otherwise, we fill the left side of the sequence with a constant zero vector as "padding" until the length is l. Since the self-attention mechanism does not contain recurrent or convolutional modules, it cannot perceive the order of the transactions in the session. Therefore, we need to take the position into account, and our model employs a fully connected layer to embed item IDs and relative position (i.e. one-hot representation) into two continuous low-dimensional spaces. Formally, let $\mathbf{M} \in \mathbb{R}^{|V|\times d}$ and $\mathbf{P} \in \mathbb{R}^{|V|\times d}$ represent a matrix consisting of the unique item

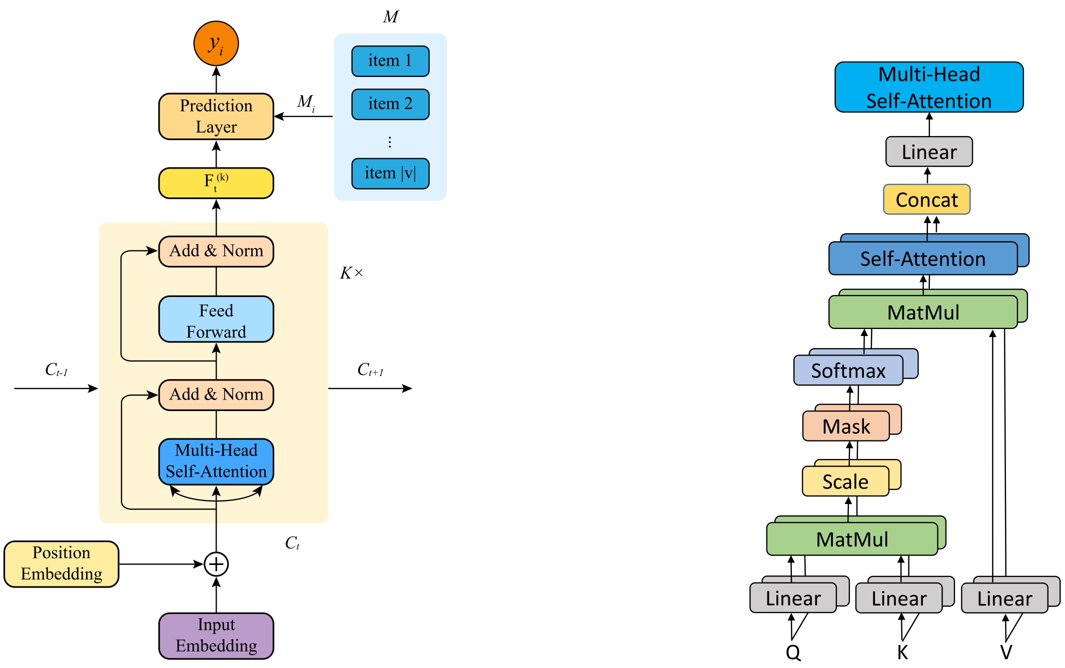


FIGURE 1. **(left)The architecture of SANSR. (right)The architecture of Multi-Head Self-Attention.**

embedding and corresponding position embedding respectively, where d is the dimensionality of the latent embedding spaces. We retrieve the input embedding matrix and corresponding position embedding matrix from $\mathbf{M}$ and $\mathbf{P}$ based on the input information and relative position information. Finally, the corresponding position embedding is embedded into the input embedding by element-by-element summation to generate the final input embedding matrix, which is represented by $\mathbf{E} \in \mathbb{R}^{l\times d}$.

B. SELF-ATTENTION BLOCKS

Since self-attention was developed, it has attracted extensive attention and has proved its amazing ability through application in many fields. Moreover, self-attention greatly improves training efficiency and reduces training time because it does not require recurrent and convolutional modules. Self-attention blocks are composed of multi-head self-attention and a pointwise feedforward network.

1) SELF-ATTENTION

The self-attention mechanism maps a query and a set of key-value pairs to an output. The output is computed as a weighted sum of values, where the weight assigned to each value is determined by the corresponding key and query. As shown in the right part of figure 1, the three inputs of multi-head self-attention, $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$, come from the same place, the final input embedding matrix $\mathbf{E}$. Then we compute the dot products of the query with all keys, divided each by $\sqrt{d}$ as the operation of the scale. Next, we mask the query and value to make the zero we pad in the embedding layer has a small weight so that it wont affect the self-attention too much. Besides, we apply a softmax function to obtain the weights on the values while dot products are used again between the matrix of weights and $\mathbf{V}$ to generate the self-attention. Since there are two heads in the architecture of multi-head self-attention, the two self-attentions are concatenated and once again projected to obtain the multi-head self-attention. The weight between a query and value is closely related to the correlation between the two. The more relevant the query is to the value, the larger the corresponding weight value will be. Therefore, the self-attention mechanism is able to characterize the global dependencies between the input and the output and to capture the interactions between items, regardless of their distance in the session. As the network becomes deeper, the increase in model capacity will lead to overfitting, and the training process will become unstable because of the vanishing gradient. To alleviate these problems, we apply layer normalization to stabilize and accelerate the network training, while dropout is used to mitigate overfitting. Finally, we normalize the input embedding matrix via layer normalization before projecting it linearly onto three matrices and apply the dropout operation on the output to obtain the final output as:

$$\begin{aligned}
\tilde{\mathbf{E}} &= \mathit{LayerNorm}(\mathbf{E}) = \alpha \odot \frac{\mathbf{E}-\mu}{\sqrt{\sigma^2+\epsilon}} + \beta \\
\mathbf{Q}, \mathbf{K}, \mathbf{V} &= \tilde{\mathbf{E}}\mathbf{W}^Q, \tilde{\mathbf{E}}\mathbf{W}^K, \tilde{\mathbf{E}}\mathbf{W}^V \\
\mathit{ATT}(\mathbf{Q},\mathbf{K},\mathbf{V}) &= \mathit{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V} \\
\mathbf{H} &= \mathit{Dropout}(\mathit{ATT}(\mathbf{Q},\mathbf{K},\mathbf{V}))
\end{aligned} \tag{1}$$

where $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d\times d}$ are projection matrices and $\sqrt{d}$ is a scale factor used to scale the dot products to avoid vanishing gradients. $\odot$ is an element-wise product, μ and σ are the mean and variance of $\mathbf{E}$, and α and β are learned scaling factors and bias terms.

2) MULTI-HEAD SELF-ATTENTION

Parallel execution of the self-attention mechanism is used to apply different linear projections to project the queries, keys and values to d dimensions. Next, the d-dimensional output values are generated, concatenated and once again projected to obtain the final values. This approach, which uses multiple projections to perform the self-attention function in parallel, is called multi-head self-attention, and it allows the model to focus on information from different representation subspaces. However, blindly increasing the number of heads in the self-attention network does not necessarily lead to improved results, which will be described in detail in the section about the influence of the hyper-parameters. The final values are computed as:

$$\begin{aligned}\mathbf{S} &= Concat(\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_h)\mathbf{W}^O \\ \mathbf{H}_i &= Droupout(ATT(\tilde{\mathbf{E}}\mathbf{W}_i^Q, \tilde{\mathbf{E}}\mathbf{W}_i^K, \tilde{\mathbf{E}}\mathbf{W}_i^V))\end{aligned} \tag{2}$$

where the $\mathbf{W}^O, \mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{R}^{d\times d}$ are the parameter matrices and h is the number of attention heads. Since the dimension of each head is reduced in multi-head self-attention, the sum of the calculation costs of all heads is similar to that of the single-head attention with full dimensionality.

3) FEEDFORWARD NETWORK

The feedforward network adopts a one-way multilayer structure, where neurons in the same layer are not related with each other and the information transmission between layers goes in only one direction. The network is a composite mapping of simple nonlinear processing units, so it can obtain complex nonlinear processing capacity to overcome the deficiency of self-attention, which is a linear model. To this end, we apply a pointwise two-layer feedforward network consisting of two linear transformations with a ReLU activation function. Although multilayer neural networks can learn meaningful features hierarchically, simply adding more layers does not necessarily result in better performance. The residual connections are able to propagate the embedding of last visited items into the final layer, making it easier for the model to take advantage of low-layer information. At the same time, the feedforward network, as a neural network, must make use of normalization and dropout, as well as self-attention.

$$\begin{aligned}\tilde{\mathbf{S}} &= LayerNorm(\mathbf{S}+\mathbf{E}) \\ \mathbf{F} &= ReLU(\tilde{\mathbf{S}}\mathbf{W_1}+\mathbf{b_1})\mathbf{W_2}+\mathbf{b_2} \\ \mathbf{F} &= Dropout(\mathbf{F})+\tilde{\mathbf{S}}\end{aligned} \tag{3}$$

where $\mathbf{W_1}, \mathbf{W_2}$ are weighted matrices and $\mathbf{b_1}, \mathbf{b_2}$ are bias vectors. For simplicity, we define the above whole self-attention block that contains multi-head self-attention and the feedforward network as:

$$\mathbf{F} = SAN(\mathbf{E}) \tag{4}$$

C. MULTI-SAN

After the self-attention block, $\mathbf{F}$ has aggregated the embeddings of all previous items with adaptive weights, and different layers can capture different types of features [6]. Therefore, higher-level item transitions can be learned through multiple levels of the self-attention network based on $\mathbf{F}$, and the k-th ($k > 1$) layer is defined as:

$$\mathbf{F}^{(k)} = SAN(\mathbf{F}^{(k-1)}) \tag{5}$$

where the first layer is denoted as $\mathbf{F}^{(1)} = \mathbf{F}$.

D. PREDICTION LAYER

1) PREDICTION

After the k-layer self-attention network adaptively extracts the information of previously consumed items, we predict the next item on the basis of $\mathbf{F}^{(k)}$. Specifically, we employ a single-item embedding $\mathbf{M}$ to reduce the model size and alleviate overfitting when we compute the score r_i for each candidate item $v_i \in V$:

$$r_i = \mathbf{F}^{(k)}\mathbf{M}_i^T \tag{6}$$

where r_i is the probability of item v_i being the next item. Hence, the higher the value of r_i is, the more likely item v_i will be the next item the user clicks on. We generate recommendations by ranking these scores.

Then, the softmax function is used to obtain the output vector of the model:

$$\hat{y} = softmax(r) \tag{7}$$

where $r \in \mathbb{R}^{|V|}$ denotes the recommendation scores of the user for how interested they are in all candidate items and $\hat{y} \in \mathbb{R}^{|V|}$ denotes the probabilities that the candidate items are the next to be clicked on in the session.

For each session, we opt for the pairwise objective function with the BPR optimization criterion [7], where a given user u prefers observed item v_i over unobserved item v_j in the current session. Then, we train our model by minimizing the following objective function:

$$\mathcal{L} = \arg\min_{\theta} \sum_{s_i\in S} \sum_{\substack{v_i\in s_i \\ v_j\notin s_i}} -\ln(\sigma(\hat{y}_i - \hat{y}_j)) + \frac{\lambda}{2}\|\theta\|^2 \tag{8}$$

where θ is the set of model parameters and $\sigma(\cdot)$ is the sigmoid function. We adopt a variant of stochastic gradient descent (SGD) called adaptive moment estimation (Adam) [8] for faster convergence.

2) INCREMENTAL TRAINING

Session-based recommender systems may confer a high volume of input at high speed, that is, the input rate is so high that the available computational resources of the system cannot process it immediately, which will lead to system overload. To alleviate the overload problem and the dynamic of input data, we adopted the incremental training method in the experiment. Whenever new data comes, it is not necessary to rebuild the model but to further train the new data based on the original model. There is a storage and recovery operation in the incremental training process, which stores the current model parameters and network structure, and restores the saved model parameters and network structure when new data is encountered. Let C_t denote the newly trained network at time t. It only needs to use the result of C_{t-1} and further training the new data. It is not necessary to retrain C_{t-1}. Next, the model parameters and network structure of C_t are stored, so that C_{t+1} can load the C_t without retraining, and so on. Through incremental training, the system only needs to save the model parameters and network structure without saving historical data, thus reducing the occupation of storage space. In addition, incremental training makes full use of historical training results in current sample training, significantly reducing the time for subsequent training.

IV. EXPERIMENTS

In this section, we first present our experimental setting; then, we conduct experiments to answer the following research questions:

RQ1: How does the performance of the proposed method compare to that of the state-of-the-art methods?

RQ2: Is the efficiency of the proposed method superior to that of the state-of-the-art methods?

RQ3: What is the influence of the components of the model, such as residual connections?

RQ4: How do the key hyper-parameters, for example, the number of attention heads and the number of self-attention blocks, affect model performance?

A. EXPERIMENTAL SETUP

1) DATASETS

We evaluate the proposed method on two real-world representative datasets, the Yoochoose,[1] which can be obtained from the RecSys Challenge 2015, composed of user clicks on an e-commerce website within 6 months, and Diginetica,[2] which is based on transactional data from CIKM Cup 2016. Following [2], [5], we eliminate all sessions of length 1 and items appearing fewer than 5 times in Diginetica dataset. To assess the performance of each method for a long session, we filter out sessions with a length of less than 5 and items with a frequency of less than 5 in Yoochoose. Furthermore, similar to [3], we generate sequences and corresponding labels $([v_{s_i,1}], v_{s_i,2})$, $([v_{s_i,1}, v_{s_i,2}], v_{s_i,3})$, ..., $([v_{s_i,1}, v_{s_i,2}, \ldots, v_{s_i,n-1}], v_{s_i,n})$ for training and testing on both datasets by splitting the input sessions $s_i = \{v_{s_i,1}, v_{s_i,2}, \ldots, v_{s_i,n}\}$. Specifically, we set the sessions of the subsequent day as the test set for Yoochoose and the sessions of the subsequent week as the test set for Diginetica. Finally, Yoochoose is left with 1,882,447 sessions and 30,830 items, while Diginetica is left with 202,633 sessions of 982,961 clicks on 43,097 items. To avoid heavy computation, we randomly sample 100 negative items and rank these items with the ground truth following previous work [9], [10]. The statistics of the datasets are summarized in Table 2.

2) BASELINES

We conduct comparisons with the following representative baselines to evaluate the performance of the proposed method.

- **Pop** is a simple baseline that always recommends the most popular items in the training set.
- **BPR-MF** [7] optimizes the matrix factorization model on implicit feedback data using a pairwise ranking loss without sequential information.
- **FPMC**[3] [11] is a classic hybrid model combining matrix factorization and first-order Markov chain for next-basket recommendation. If this method ignores the user latent representations when calculating the recommendation score, it is equivalent to session-based recommendation.
- **GRU4Rec**[4] [1] is an RNN-based deep learning model for session-based recommendation that aims to provide accurate recommendations by modeling the whole session. GRU4Rec utilizes a session-parallel mini-batch training process to model user action sequences.
- **NARM**[5] [2] employs RNNs with attention mechanisms to capture a user's main purpose and sequential behavior, which are treated as equally important complementary features.
- **STAMP**[6] [5] is a novel short-term memory priority model that captures a user's general interest based on the long-term memory of a session context and the user's current interest based on the short-term memory of the last clicks in a session.

3) EVALUTION METRICS

To evaluate the recommendation performance of all models, we adopt three common metrics, i.e., Hit-ratio(HR), Mean Reciprocal Rank (MRR) and Normalized Discounted Cumulative Gain (NDCG).

- **HR**: When the number of ground truth is one, the hit ratio is equivalent to recall, which is used to calculate the probability that the correct items are returned. HR@N is a proportion defined as the number of records that in the recommended list of top N divided by the total number of test records, that is, how many of the first N items are recommended to hit the user's actual preferences. This metric is an evaluation of unranked retrieval results.
- **MRR**: MRR is the average of the reciprocal ranks of the correctly recommended items. MRR@N is set to 0 when the rank exceeds N. This metric takes the recommendation ranking into account, and the higher the MRR value

[1]http://2015.recsyschallenge.com/challege.html
[2]http://cikm2016.cs.iupui.edu/cikm-cup

[3]http://github.com/khesui/FPMC
[4]http://github.com/hidasib/GRU4Rec
[5]https://github.com/lijingsdu/sessionRec_NARM
[6]https://github.com/uestcnlp/STAMP

TABLE 1. The performance of different methods on the two datasets. We generate the Top-10 and Top-20 items for recommendation. Boldface indicates the best results (the higher, the better), while the second best are underlined.

Datasets	Diginetica						Yoochoose					
Measures	HR@10	HR@20	MRR@10	MRR@20	NDCG@10	NDCG@20	HR@10	HR@20	MRR@10	MRR@20	NDCG@10	NDCG@20
Pop	0.0066	0.0111	0.0025	0.0028	0.0034	0.0046	0.0498	0.0811	0.0203	0.0224	0.0271	0.0349
BPR-MF	0.0980	0.1295	0.0448	0.0350	0.0571	0.0638	0.2321	0.2709	0.1267	0.1296	0.1147	0.1315
FPMC	0.1215	0.1723	0.0565	0.0602	0.0719	0.0849	-	-	-	-	-	-
GRU4Rec	0.2001	0.3165	0.0777	0.0863	0.1064	0.1364	0.4711	0.6043	0.2167	0.2268	0.2134	0.2473
NARM	0.5139	0.5989	0.2668	0.2734	0.3156	0.3476	0.5423	0.6612	0.2486	0.2563	0.3192	0.3486
STAMP	0.5150	0.6099	0.2674	0.2741	0.3263	0.3504	0.5440	0.6633	0.2503	0.2587	0.3197	0.3500
SANSR	**0.7837**	**0.8391**	**0.6514**	**0.6524**	**0.6832**	**0.6951**	**0.9516**	**0.9698**	**0.8217**	**0.8186**	**0.8538**	**0.8550**

∗On Yoochoose, we don't have enough memory to initialize FPMC

TABLE 2. Statistics of the experiment datasets.

Dataset	Yoochoose	Diginetica
#clicks	16,017,273	982,961
training sessions	1,877,867	186,670
test sessions	4,580	15,963
#items	30,830	43,097
avg.len	8.50	4.85

is, the closer the correct recommendation is to the top of the ranking list.

- **NDCG**: NDCG evaluates ranking performance by taking the positions of correct items into consideration. NDCG@10 considers only whether the ranking of the top N is correct.

In this experiment, we consider N values of 10 and 20.

B. PERFORMANCE COMPARISON

In this subsection, we compare our SANSR model with the state-of-the-art methods to demonstrate its performance. The results of all methods on two datasets are shown in Table 1, and we have the following observations.

First, the performance of the simplest popularity-based nonpersonalized recommendation methods (i.e., Pop) is the worst on bost datasets, while BPR-MF improves the performance by considering the personalized preferences of each user and optimizing the pairwise ranking loss function. This result confirms that personalization plays an important role in recommendation tasks. However, BPR-MF does not consider any session sequence information, which may be the reason FPMC has better performance.

Second, GRU4Rec, the first method to introduce RNN into session-based recommendation to capture the general characteristics of users, achieves impressive performance compared to traditional methods. This result illustrates the great potential of deep learning techniques in session-based recommendation. The methods that not only apply the RNN-based method but also leverage the attention mechanism (i.e., NARM and STAMP) perform better than GRU4Rec, which demonstrates that the attention mechanism can effectively improve the performance of the neural network model. STAMP performs better than NARM, indicating that short-term behavior is effective in improving the predictions.

Finally, our proposed SANSR consistently outperforms all other methods by a large margin under all measurements on both datasets. SANSR performs better than Pop and BPR because it contains sequential information. The reason why SANSR is superior to FPMC and GRU4Rec is that it is able to capture users dynamic preference by considering the whole interaction sequence. The SANSR gets better results than the most advanced method for the session-based recommendation (i.e., NARM and STAMP) because the RNN-based attention approach is to pass information between adjacent items while self-attention captures their interactions regardless of the distance between the items. In addition, the performance on the Yoochoose is better than that on Diginetica, possibly because the average length of Yoochoose is longer. The longer the session length is, the richer the historical data that can be obtained by the self-attention mechanism to more comprehensively establish the global dependence between input and output and obtain more accurate prediction results.

C. EFFICIENCY ANALYSIS

Because session-based recommendations are typically applied to online applications, it is important to evaluate the efficiency of the proposed model. In this subsection, we use training speed (time taken for one epoch of training) as a comparison criterion to explore whether our proposed method, SANSR, has performance advantages over other baseline methods. Since the first baseline Pop does not involve a training process, it is excluded from the comparison. The running time of the two datasets is quite different, we show the running time of the two datasets respectively in Figure 3 to make the results more intuitive. Above all, methods that do not use neural networks (i.e., BPR and FPMC) are usually time consuming, which may be because they cannot achieve incremental learning. Furthermore, GRU4Rec is more efficient than not only the methods that do not use neural networks, but also NARM because it utilizes a session-parallel mini-batch training process. Besides, STAMP takes the least time, probably because it applies simple MLP network while NARM employs the GRU network and SANSR utilizes a deep architecture which contains multiple levels of the self-attention network. The MLP has faster training speed. Finally, the training speed of SANSR is smaller than other baselines except STAMP, which may be because it supports

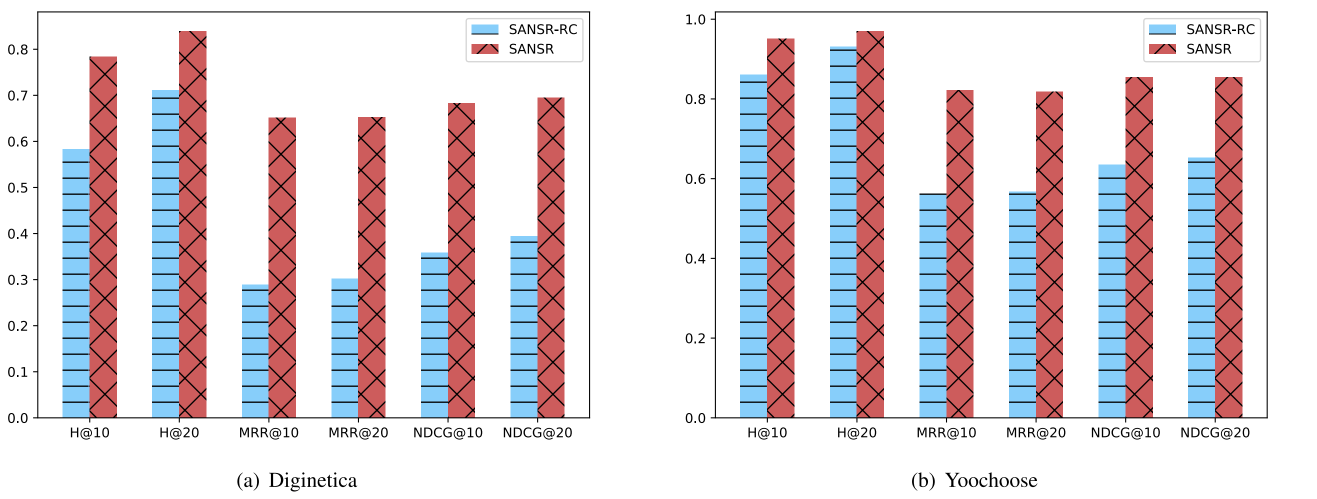


FIGURE 2. **Influence of Residual Connections on the Diginetica and Yoochoose datasets.**

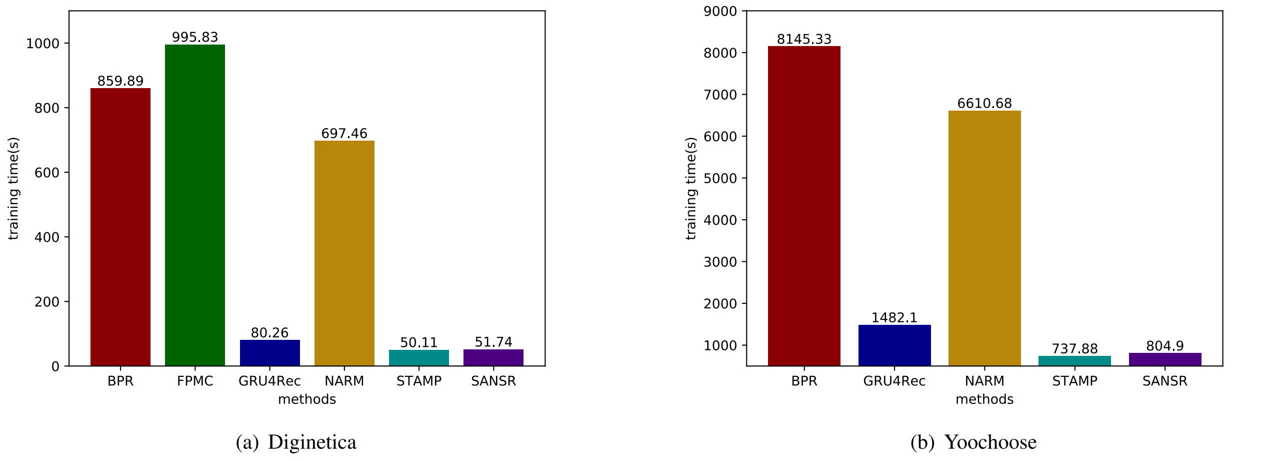


FIGURE 3. **Efficiency of all methods on Diginetica and Yoochoose datasets.**

parallelism without recurrent module in the session and the incremental training method can save the training time. On the basis of the previous performance comparisons, we can also find that although STAMP trains a little faster than SANSR, the performance of STAMP is worse. After considering the balance between efficiency and effectiveness, SANSR performs better.

D. INFLUENCE OF COMPONENT

We conduct experiments to analyze the contribution of residual connections in forming our model. SANSR-RC indicates that residual connections are excluded from the model. The residual connections are used in the input and output of the feedforward networks to enable the network to take advantage of the low-layer information. In Figure 2, we observe that the performance without residual connections is significantly worse in terms of all metrics. The reason may be that there is information in low layers that is helpful to the recommendation, but this information is difficult to propagate to the final layer without residual connections. Therefore, residual connections are useful not only to make the model learn meaningful features in layers but also to make full use of the low-layer features.

E. INFLUENCE OF HYPER-PARAMETERS

Different models have different hyper-parameters, but some hyper-parameters are common and play important roles in model performance. In this subsection, we optimize the performance of our model by studying the impacts of several hyper-parameters, such as the number of attention heads and self-attention blocks, and embedding size.

1) IMPACT OF THE NUMBER OF SELF-ATTENTION HEADS h

The model is capable of attending to information from different representation subspaces by means of multi-head self-attention. We consider the number of self-attention heads of $\{1, 2, 3, 4, 5\}$ to optimize the performance of our model. Since the dimension size must be an integer multiple of

TABLE 3. The Performance of SANSR with varying *h* and *k* in terms of NDCG@20 on Diginetica and Yoochoose datasets.

Dataset	N@20 \ k / h	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$
Diginetica	h =1	0.5622	0.6560	0.6428	0.5293	0.3086
	h =2	0.5493	0.6562	0.6479	0.5393	0.2936
	h =3	0.5724	0.6484	0.6428	0.6419	0.3021
	h =4	0.5462	0.6540	0.6402	0.5142	0.2959
	h =5	0.5479	**0.6574**	0.6433	0.4907	0.2989
Yoochoose	h =1	0.7895	0.8468	0.7578	0.5239	0.4947
	h =2	0.8159	0.8559	0.8223	0.5509	0.4854
	h =3	0.8093	0.8525	0.8137	0.5434	0.4114
	h =4	0.8184	0.8512	0.8098	0.6094	0.4078
	h =5	0.8045	**0.8576**	0.8288	0.6000	0.4129

the number of attention heads in multi-head self-attention, we take the dimension size as 120. From Table 3, we find that there is no specific rule for the influence of h on the model, which may be because the dimension size we choose is not suitable for decomposition into smaller subspaces. In other words, increasing the number self-attention heads does not necessarily improve the performance of the model. In this case, we need to choose an appropriate value of h for the model.

2) IMPACT OF THE NUMBER OF SELF-ATTENTION BLOCKS k

Although a self-attention block has the ability to aggregate all previous items' embeddings, it will be better if more complex item interactions can be learned through several blocks. The number of self-attention blocks is searched from $\{0, 1, 2, 3, 4\}$. The effect of the number of self-attention blocks on the performances in Table 3 shows a significant trend, that is, the performance increases with increasing number of blocks and then decreases after reaching a threshold. Specifically, the performance is poor when $k = 0$ because the model is based on only the last item. And we find that SANSR still performs better than the baselines when $k = 0$, probably because SANSR takes into account positional information, and the last item does play an important role in characterizing user preferences. Besides, after getting the input embedding and positional embedding with fully connected layer, the input embedding and positional embedding are multiplied by $\sqrt{d}$ as the operation of the scale, which may affect the results. The best performance is achieved when $k = 1$, and the performance remains good when $k = 2$, indicating that the hierarchical self-attention structure is helpful for learning complex item interactions. Finally, the performance starts to decline obviously when $k = 3$, which may be because the model becomes increasingly complex as the number of self-attention blocks increases. After comprehensive consideration of the influence of k and h, we set $h = 2, k = 1$ for both datasets.

3) IMPACT OF THE EMBEDDING SIZE d

In our model, the dimension size d is relevant to not only the item embedding size but also the dimension of the projection matrices in self-attention. We search embedding sizes from $\{32, 64, 128, 256, 512\}$, and the results are shown in Figure 4. We observe that large embedding dimensions result in better item embeddings and are effective for building high-level factor interactions through a self-attention network. In the experiments, we set the embedding size to 128 on both datasets to achieve an acceptable trade-off between computation cost and recommendation quality.

V. RELATED WORK

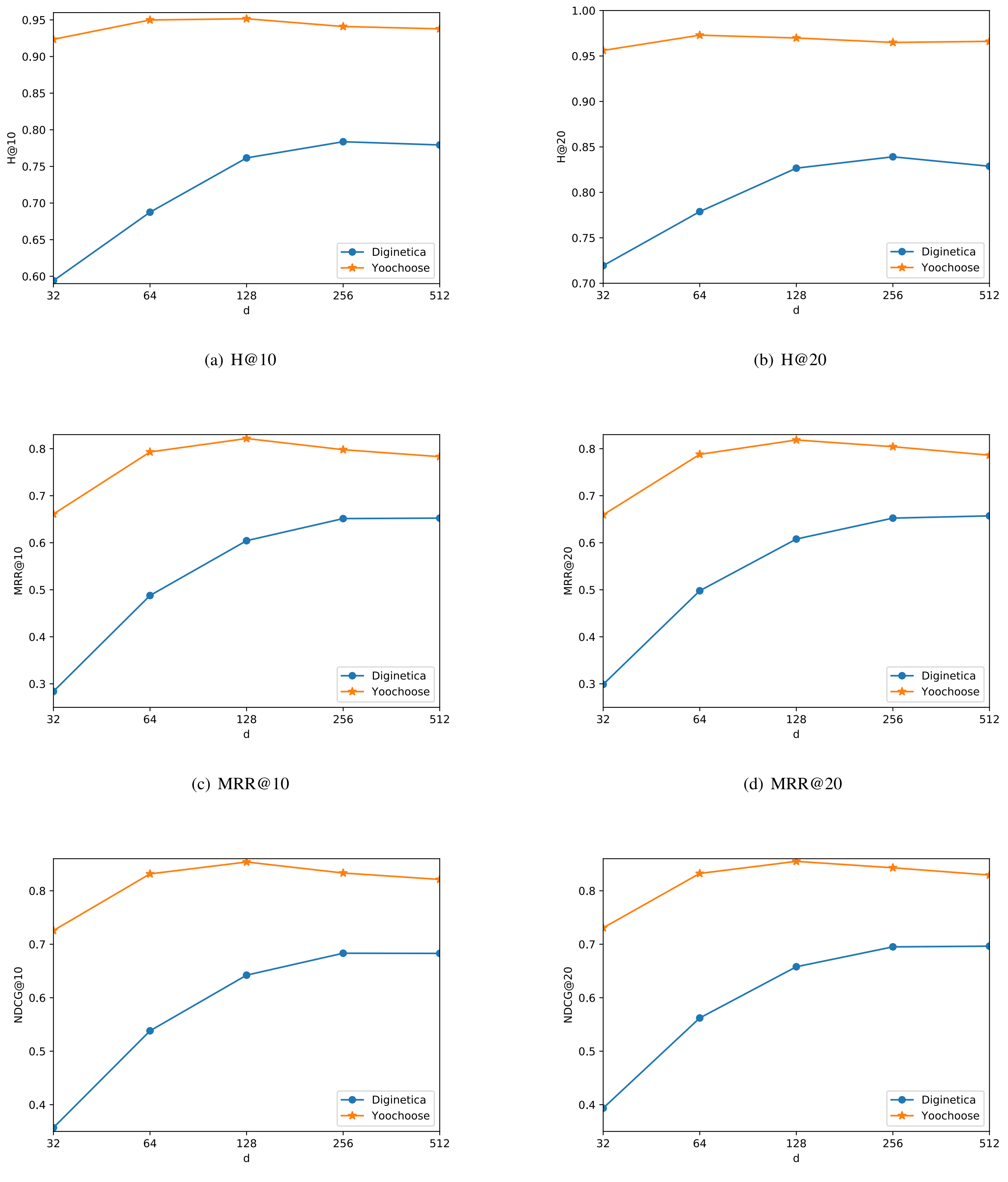
In this section, we briefly review closely related work from two perspectives: session-based recommendation and attention mechanism.

A. SESSION-BASED RECOMMENDATION

Session-based recommendation is a challenging task because, in many cases, users are anonymous and their preferences (e.g., ratings) are not visible. The methods to obtain the general interest of users are composed of mainly item-based collaborative filtering methods, neighborhood methods and Markov chain-based models. Item-based collaborative filtering [12] describes user preferences based on the user's entire history in the session. The neighborhood approach [13] makes recommendations based on item similarities calculated by the common co-occurrence of items in the session. Models based on Markov chains [14], [15] utilize sequential connections between user behavior to make predictions. However, these methods rarely consider the sequential interactions between items that are not adjacent in the session, thereby ignoring the information of the whole sequence of sessions.

Deep neural network methods overcome the traditional method's inability to characterize sequential information in session-based recommendation. Reference [1] proposes the application of RNN to session-based recommendation and regards the item clicks in each session as a sequence, which is depicted by GRU. Reference [16] proposes a hierarchical RNN model to transfer user interest between sessions for personalized session recommendation. The above solutions are primarily based on RNNs, especially LSTM and GRU, which have good performance when modeling sequential data and the potential to solve the vanishing gradient problem.

In addition to the dynamic and evolutionary characteristics, users' interests in a session are also diverse. Attention-based RNN methods can automatically assign different influences to previous items through the attention mechanisms. Reference [2] takes into account both the user's sequential behavior and the main purpose in the current session and computes recommendation scores via a bilinear matching scheme. Reference [5] emphasizes the current interest represented by the last click to capture the hybrid features of the current and general interests on the basis of previous clicks, thus explicitly introducing the importance of the last click into the recommender system.



(a) H@10 (b) H@20

(c) MRR@10 (d) MRR@20

(e) NDCG@10 (f) NDCG@20

FIGURE 4. **Influence of embedding size under all metrics.**

B. ATTENTION MECHANISM

Attention mechanisms are widely used in many fields, such as image/video caption [17], [18] and machine translation [19], because of their ability to select information that is critical to the current task target from a large amount of information. The attention technique utilizes different weights to adjust the effects of different parts on the target. For example, [20] designs a two-layer hierarchical attention network to take both user-item and item-item interactions into account for sequential recommendation.

Reference [21] employs an attention mechanism to capture users' varying preferences for items. However, these traditional attention mechanisms can capture only a fixed user preference representation and ignore the diversity of user preferences. Reference [22] learns deep adaptive user representations through a memory component and a novel attention mechanism to handle the diversity of users' interests. In all the above methods, the attention mechanisms are implemented to enhance the recommendation accuracy on the basis of a neural network and cannot exist independently from the neural network. Recently, [23] models the dependencies between words based entirely on self-attention, without any recurrence or convolution, and achieves state-of-the-art performance. However, self-attention has not been used in session-based recommendation. To this end, we propose the SANSR method, which can adaptively assign weights to previous items at each time step in a session, to model the relationship between the items in sessions.

VI. CONCLUSION

In this paper, we propose a novel model named self-attention network for session-based recommendation (SANSR) to provide users with accurate recommendations. Specifically, we use a multilayer self-attention network to learn the interactions of high-level item transitions. In each layer of the attention network, multi-head self-attention is applied to represent the interactions between items, regardless of distance, and to capture the relevance between the previous time steps in the current session and the items. To solve the information overload problem caused by streaming data input in session-based recommendation, incremental learning is adopted to update the model, and the latest model is saved in limited memory. The proposed method not only dispenses with the recurrent and convolutional modules to simplify the network but also supports parallelism when representing users' short-term and long-term preferences. The experimental results demonstrate that SANSR achieves good performances in terms of HR, MRR, and NDCG compared with several start-of-the-art methods on two real-world datasets.

REFERENCES

[1] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," 2015, *arXiv:1511.06939*. [Online]. Available: https://arxiv.org/abs/1511.06939

[2] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," in *Proc. ACM Conf. Inf. Knowl. Manage.*, Nov. 2017, pp. 1419–1428.

[3] Y. K. Tan, X. Xu, and Y. Liu, "Improved recurrent neural networks for session-based recommendations," in *Proc. 1st Workshop Deep Learn. Recommender Syst.*, Sep. 2016, pp. 17–22.

[4] B. Hidasi, M. Quadrana, A. Karatzoglou, and D. Tikk, "Parallel recurrent neural network architectures for feature-rich session-based recommendations," in *Proc. 10th ACM Conf. Recommender Syst.*, Sep. 2016, pp. 241–248.

[5] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, "STAMP: Short-term attention/memory priority model for session-based recommendation," in *Proc. SIGKDD*, Aug. 2018, pp. 1831–1839.

[6] A. Anastasopoulos and D. Chiang, "Tied multitask learning for neural speech translation," 2018, *arXiv:1802.06655*. [Online]. Available: https://arxiv.org/abs/1802.06655

[7] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. 25th Conf. Uncertainty Artif. Intell.*, Jun. 2009, pp. 452–461.

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980

[9] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. Int. Conf. World Wide Web*, Apr. 2017, pp. 173–182.

[10] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. SIGKDD*, Aug. 2008, pp. 426–434.

[11] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized Markov chains for next-basket recommendation," in *Proc. 19th Int. Conf. World Wide Web*, Apr. 2010, pp. 811–820.

[12] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Comput.*, vol. 7, no. 1, pp. 76–80, Jan./Feb. 2003.

[13] B. Sarwar, G. Karypis, G. Karypis, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web*, vol. 1, pp. 285–295, May 2001.

[14] W. Gu, S. Dong, and Z. Zeng, "Increasing recommended effectiveness with Markov chains and purchase intervals," *Neural Comput. Appl.*, vol. 25, no. 5, pp. 1153–1162, Oct. 2014.

[15] G. Shani, D. Heckerman, and R. I. Brafman, "An MDP-based recommender system," *J. Mach. Learn. Res.*, vol. 6, pp. 1265–1295, Sep. 2005.

[16] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, "Personalizing session-based recommendations with hierarchical recurrent neural networks," in *Proc. 11th ACM Conf. Recommender Syst.*, Aug. 2017, pp. 130–137.

[17] K. Cho, A. Courville, and Y. Bengio, "Describing multimedia content using attention-based encoder-decoder networks," *IEEE Trans. Multimedia*, vol. 17, no. 11, pp. 1875–1886, Nov. 2015.

[18] J. Chen, F. Zhuang, X. Hong, X. Ao, X. Xie, and Q. He, "Attention-driven factor model for explainable personalized recommendation," in *Proc. SIGIR*, Jul. 2018, pp. 909–912.

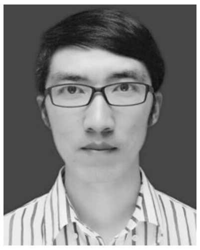
[19] O. Caglayan, L. Barrault, and F. Bougares, "Multimodal attention for neural machine translation," 2016, *arXiv:1609.03976*. [Online]. Available: https://arxiv.org/abs/1609.03976

[20] H. Ying, F. Zhuang, F. Zhang, Y. Liu, G. Xu, X. Xie, H. Xiong, and J. Wu, "Sequential recommender system based on hierarchical attention network," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 3926–3932.

[21] T. Bai, J.-Y. Nie, W. X. Zhao, Y. Zhu, P. Du, and J.-R. Wen, "An attribute-aware neural attentive model for next basket recommendation," in *Proc. SIGIR*, Jul. 2018, pp. 1201–1204.

[22] L. Zheng, C.-T. Lu, L. He, S. Xie, V. Noroozi, H. Huang, and P. S. Yu, "MARS: Memory attention-aware recommender system," 2018, *arXiv:1805.07037*. [Online]. Available: https://arxiv.org/abs/1805.07037

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.



SHIMING SUN received the B. Eng. degree in computer technology from the Nanjing Institute of Technology, Nanjing, China, in 2003, the M.S. degree in computer software engineering from Hohai University, Nanjing, in 2006. He is currently a Senior Engineer with the Department of Basic Research and Development, NARI Group Corporation (State Grid Electric Power Research Institute), Nanjing. His research interests include electric power automation systems, recommender systems, and big data analysis.

YUANHE TANG received the M.S. degree from North China Electric Power University, China, in 2011. He is working in smart grid field. His research interests include recommender systems and software architecture design on power dispatching automation systems.

ZEMEI DAI received the bachelor's and M.S. degrees from Xi'an Jiaotong University, Xi'an, China, in 1995 and 1998, respectively. She is with the State Key Laboratory of Smart Grid Protection and Control of NARI GROUP Corporation. Her research interests include the application technology of power systems, automation systems, recommender systems, and the power Internet of Things.

FU ZHOU received the M.S. degree in agricultural electrification and automation from Jiangsu University, in 2010. He is currently an Engineer with the NARI Research Institute. His research interests include distribution automation systems, big data applications, recommender systems, graph theory, and wireless sensor networks.