IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 12, 2020, accepted March 25, 2020, date of publication March 30, 2020, date of current version April 14, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2984039

Session-Based Graph Convolutional ARMA Filter Recommendation Model

HUANWEN WANG, GUANGYI XIAO, NING HAN, AND HAO CHEN
College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China

Corresponding author: Hao Chen (chenhao@hnu.edu.cn)

This work was partially supported by the National Key R&D Program of China (2018YFB1402600) and the National Natural Science Foundation of China (61772190).

ABSTRACT With the rapid popularity of Internet shopping, session-based personalized recommendations have become an important means to help people discover their potentially interesting items in real-time. Most existing works only model a session as a sequence and use recurrent neural networks for recommendation. Despite their effectiveness, the results may not be sufficient to capture the potential relationships between items. In this work, we integrate a graph convolutional layer based on Auto-Regressive Moving Average (ARMA) filters into the Graph Neural Network (GNN). In particular, it is a new session-based recommendation framework Graph Convolution ARMA Filter (AUTOMATE), which can capture complex transformations between items through a sequence of sessions modeled as graph-structured data. Each session is then represented as the composition of the current interest and the global preference of that session using an attention network. The rationality and efficacy of the proposed AUTOMATE model are extensively evaluated on two public real-world datasets. Experimental results show that our model is significantly better than other state-of-the-art methods.

INDEX TERMS ARMA, attention mechanism, graph convolutional networks, sequential behavior, session-based recommendation.

I. INTRODUCTION

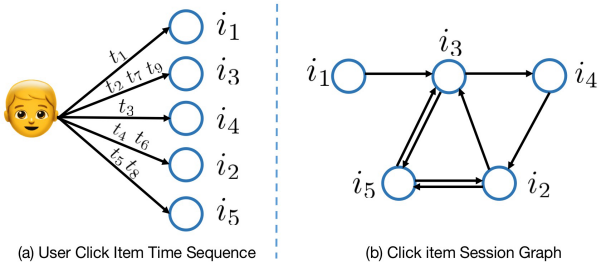
A. MOTIVATION

As a branch in the field of recommendation, the session-based recommender system has attracted attention from academia and industry in recent years. Some e-commerce recommendation systems and news media websites usually track the visitors' id, recommending content that may be of interest to users in real-time to improve the user's experience. Most existing session-based recommendation methods use item-to-item similarity, co-occurrence, transition probabilities or Recursive Neural Networks(RNN) for the recommendation. Although effective, learning to predict from an anonymous session is still a challenging problem due to the inherent uncertainty of user behavior and limited information provided by the browser sessions.

The basic purpose of session-based recommendation is to improve the accuracy of recommendations and improve the user experience. Although many methods have been proposed so far, session-based recommendation remains in its infancy due to the following challenges: (1) The Markov Chain (MC) [1] assumes that there is a strong dependency between consecutive items operated by the users. However, this may not be the case in the real-world transactional data because a user may just randomly pick up some items he/she likes into the cart. (2) RNN have made significant progress in session-based recommendation tasks [2]–[5]. However, most of existing RNN-based models do not expose the global knowledge of frequent click patterns or consider the variability of user interest drift with time [6]. (3) The self-attention as a special attention mechanism, has been widely used to model sequence data, and has achieved significant results in many applications, e.g., machine translation [7] and sequential recommendation [8]. However, this operation disperses the distribution of attention, which results in lacking local dependencies over adjacent items and limiting its capacity for learning contextualized representations of items [9].

In general, despite their effectiveness, we argue that the graph-structured data model has more expressive abilities to handle repeated or cycled purchases, whereas traditional sequential methods have difficulty in coping with, these methods always model single way transitions between con-

The associate editor coordinating the review of this manuscript and approving it for publication was Choon Ki Ahn.



(a) User Click Item Time Sequence (b) Click item Session Graph

FIGURE 1. An illustration of the user click sequence and corresponding graph.

secutive items and neglect the transitions among the contexts. Thus, complex transitions among distant items are often overlooked. More specifically, most existing methods use only descriptive features (e.g., IDs and attributes) to build embedding functions, and abandon the connection and temporal modeling between items. As a result, when embedding is not enough to capture Collaborative Filtering (CF), these methods must rely on interactive capabilities to make up for the suboptimal embedding. In this work, first of all, we consider that the user clicks on the item is time-series, and introduces the ARMA filter method to extract time-series information in the session, which can effectively evaluate the time decay effect of the user's historical preferences on the current session preferences. However, the traditional ARMA filters are not located in the node space, which makes their implementation inefficient. To solve this scalability problem, we use a recursive formula to represent the ARMA layer, which leads to a fast and distributed implementation that exploits efficient sparse operations on tensors. Secondly, when there are not enough click actions in the session and there are few loops, it is difficult to predict the user's next action. Therefore, we take item as the center and model the separated session sequence as graph-structured data to better capture the complex relationships among items, between sessions of different users, and sessions of the same user in different time periods. In addition, the obtained ARMAConv filters are not learned in the Fourier space caused by a given Laplacian, but are located in the node space and have nothing to do with the underlying graph structure. This allows our AUTOMATE to process graphs with unknown topologies in inductive inference tasks, thereby better-predicting items that users might like.

Running example. Figure1 illustrates the concept of the graph structure formed by the user's click sequence. Figure 1(a) shows the user-item interaction, the avatar represents an anonymous user, and items are represented by circles, which correspond to $i_1, i_2, \ldots, i_5$ respectively. The user's click sequence within a session is marked on the interactive line, which is represented by $t_1, t_2, \ldots, t_7$ respectively. Figure 1(b) shows the graph structure between items within the session (in a single session). This graph structure is a directed graph composed of the chronological order of items clicked by the user. This sequence connection contains rich semantics and can carry graph signals $i_1 \rightarrow i_3 \rightarrow i_4 \rightarrow i_2 \rightarrow i_5$ indicates that there is a certain relationship between these items. The longer path $i_1 \rightarrow i_3 \rightarrow i_4 \rightarrow i_2 \rightarrow i_5 \rightarrow i_2 \rightarrow i_3 \rightarrow i_5 \rightarrow i_3$ suggests that the similarity between i_5 and i_4, as both item have interacted with i_3. Moreover, from the holistic view of session graph, item i_3 is more likely to be of interest to user than item i_5 and i_2, since there are three paths to i_3, while only two paths to i_5 and i_2. Of course, this is just a graph structure composed of a single session sequence for a single user. Correspondingly, in our model, we take item as the center and model all the separated session sequences as graph-structured data, which is better captures complex relationships between items, between sessions of different users, and sessions of the same user at different time periods.

B. SUMMARY OF CONTRIBUTIONS

In this study, we address the limitations of existing graph convolutional layers in modeling the desired filter response and propose a novel graph neural networks framework, namely Gr**a**ph Convol**uti**on AR**MA** Fil**te**r (AUTOMATE), to explore rich transitions among items and generate accurate latent vectors of items, as shown in Figure 2. It can be seen as illustrating the workflow of the proposed AUTOMATE method. Specifically, all historical session sequences are constructed as a directed graph, where each session sequence can be treated as a subgraph. Then we process each session graph in turn and updates an item's embedding by aggregating the embedding of the interacted items. After that, we use the linear weighted sum of the user's global interests and his/her local interests in that session as the embedding vector. Finally, we predict the likelihood of the next click for each session. This work makes the following main contributions:

- The use of separate session sequence modeling graph structured data for recommendation is a promising and challenging exploration. As far as we know, this is the first time that a graph convolution neural network and an ARMA method to solve serialized data are considered. Combined with solving the session recommendation problem.
- We propose a novel end-to-end model AUTOMATE to obtain better session representations for predicting products that users may like.
- Extensive experiments on real datasets have proved the rationality and effectiveness of our method, and at the same time, we publish our project and datasets at https://github.com/HuanwenW/AUTOMATE.

The rest of the paper is organized as follows. In Section II, we review related research. In Section III, we give a formal definition of the session-based recommendation problem and describe the proposed AUTOMATE model in detail. In Section IV, we give comparative experimental results in different dimensions. In Section V, we summarize the paper and mention further research directions.

II. RELATED WORK

Session-based recommendation is a branch of the recommender system, which is a typical application of

recommender systems based on implicit feedback. Data only provides positive observations (e.g., click sequences), without any explicit preference information. In this section, we briefly review the most related work on session-based recommendation from the following three aspects, i.e., conventional recommendation methods, ARMA based methods, and neural network based methods. Here we highlight the differences with our AUTOMATE.

A. CONVENTIONAL RECOMMENDATION METHODS

One way to attract general interest among users is through CF approach based on the user's entire purchase/click history. For example, MF [10], [11] projects the ID of each user and item as an embedding vector, and conducts an inner product between them to predict an interaction. Along with this line, Yang *et al.* [12] modeled mutual influence between users, and mapped users into two low-dimensional spaces: truster space and trustee space, by factorizing social trust networks. Koren [13] learned temporal representations by factorizing the (user, item, time) tensor. Xiong *et al.* [14] developed a similar model named time SVD++. Another method is a Markov Chain (MC) sequential method based on strong independence related hypotheses. which predict users' next behavior based on the user's previous behavior. For example, Zimdars *et al.* [15] proposed a sequential recommender based on Markov chains and investigated how to extract sequential patterns to learn the next state using probabilistic decision-tree models. Shani *et al.* [16] study different sequential patterns for recommendation and found that contiguous sequential patterns are more suitable for sequential prediction task than general sequential patterns. Rendle *et al.* [1] first proposed a hybrid model of matrix factorization and first-order Markov chains, which can be used for the next shopping basket recommendation. The third way is called neighborhood methods. For example, Item-KNN [17] based heuristic-based session K nearest neighbor (KNN) scheme, which tries to make recommendations based on item similarities calculated from the co-occurrences of items in sessions. This cluster-based approach outperforms the above conventional recommendations in most test configurations and the datasets at that time.

B. ARMA METHODS

Compared to general approximation filter category solutions [18], the Automatic Regression Moving Average Filter (ARMA) provides more accurate filter design, in some cases, provides accurate results when modeling the required response program solution. And recently many ARMA-based hybrid models have been proposed [19]–[21]. The basic operations in graph signal processing include processing signals indexed on graphs either by filtering them, extracting specific parts from them or by changing their representation domain using some kind of transformation or dictionary that is more suitable for representing the information contained in them [22]. Zheng *et al.* [23] proposed a spectral convolution operation to discover all possible connectivity between users and items in the spectral domain. Moreover, the graph convolutional layer model based on an Auto-Regressive Moving Average (ARMA) filter [19] is significantly better than previous polynomial-based filters.

C. NEURAL NETWORK BASED METHODS

The session-based recommendation method has only achieved breakthrough development in recent years [24]–[26]. In summary, The approaches based on RNN [2], [3], [27] and the method based on Graph Convolutional Network (GCN) [26], [28], [29] are the two most mainstream branches, and the purpose of both is to explore the user preference timing relationships in a session.

RNN has been applied successfully in numerous applications, such as the most successful model for modeling sentences the sequential [30], the click prediction [31], location prediction [32], and next basket recommendation [33]. Hidasi *et al.* [2] were one of the first people to explore GRU as a special form of RNN to predict the next action in a session. Subsequently, Li *et al.* [3] proposes a hybrid encoder with an attention mechanism that can model the user's sequential behavior and capture the user's main purpose in the current session. Recently, Liu *et al.* [27] proposed a model that captures the user's overall interest from the long-term memory of the session context, while taking into account the user's current interest in the recently clicked short-term memory, and achieved a good breakthrough.

The emergence of GCN brings new hope for learning the representation of graph-structured data, which is being widely used in different tasks [29], [34], [37], [38]. GCN [35] is a scalable approach that chooses the convolutional architecture via a localized approximation of spectral graph convolutions, which is an efficient variant and can operate on graphs directly as well. After this method, the Graph Attention Network [36] was proposed, which uses the masked self-attention layers to address the shortcomings of prior methods based on graph convolutions or their approximations. Wu *et al.* [28] apply GCN on aggregating separated session sequences into graph-structured data, applying GNN to generate potential vectors of items, and then representing each conversation through a traditional attention network. Subsequently, Xu *et al.* [26] proposed a graph contextualized self-attention model, which utilizes both graph neural network and self-attention mechanism, for session-based recommendation.

In this paper, we combine the ARMA filter with time series characteristics and convolutional neural network in the research of session recommendation. This allows our AUTOMATE to process graphs with unknown topologies in inductive inference tasks, thereby better-predicting items that users might like. The difference from the previous research work is mainly that we construct the separated session sequence into graph-structure data that can better capture the complex relationship between items, sessions of different users, and sessions of the same user at different time periods. In addition, considering that the user clicks on the item is not isolated but

time-series, we introduce the ARMA method to extract the time-series information in the graph model.

III. THE PROPOSED METHOD

In this section, we describe the proposed AUTOMATE model. We first formulate the problem of session-based recommendation. Then we describe the architecture of our model in detail.

A. SESSION-BASED RECOMMENDATION

A session is a set of items (e.g., products, music or movies) that are collected or consumed in one event or a collection of actions or events that happened in a period of time. Given partially known session information (e.g., part of a session or recent historical sessions), session-based recommendation aims to predict the unknown part of a session or the future sessions based on modeling the complex relations embedded within a session or between sessions. Here we give a formulation of the session-based recommendation problem as below.

In general, in session-based recommendation, let $I = \{i_1, i_2, i_3, \ldots, i_{|I|}\}$ denote a set of all unique items. A collection of items that are interacted by a certain user during a certain period (e.g., one hours) or in a certain event (e.g., one shopping visit) constitute a session $s = \{i_1, i_2, i_3, \ldots, i_{|s|}\}$. All the sessions together in one dataset denoted by $S = \{s_1, s_2, s_3, \ldots, s_{|S|}\}$. Generally, The goal of the session-based recommendation is predict the next click i_t by taking the prior session information as the context and condition, which is called a session context C in this work. Taking session s_n as the current session for making recommendations (recommending unknown item i_t in s_n), an intra-session context C^I is the set of items that are already known in s_n, namely $C^I = \{i | i \in s_n, i \neq i_t\}$. To be exact, our model generates a ranking list over all candidate items that may occur in that session. $\hat{y} = \{y_1, y_2, y_3, \ldots, y_{|I|}\}$ denotes the output probability for all items, where $\hat{y}_k (1 \leq k \leq |I|)$ corresponds to the recommendation score of item i_t. Since a recommender typically makes more than one recommendation for the user, thus we will choose the items with top-K values in $\hat{y}$ to be the candidate items for recommendation.

B. AUTOMATE

We now present our AUTOMATE as illustrated in Figure 2. There are five components in the framework: (1) The input layer consists of directed session graphs composed of all session sequences. (2) All session graphs use shared item as a link to form a more informative graph, the embedding layer projects item attribute features to vector representations. (3) The ARMA graph convolutional layer can naturally deal with time-varying topologies and graph signals, which are used to obtain the vector of all nodes involved in the session graph. (4) The hybrid embedded session layer uses the attention network to combine the global preferences and current interests of each session. And (5) the prediction layer will predict the probability of each item that will appear to be the next-click one for each session and make recommendations.

1) SESSION GRAPHS STRUCTURE

Given a session $s = \{i_1, i_2, i_3, \ldots, i_{|s|}\}$, we treat each item i_k as a node and (i_{k-1}, i_k) as an edge which represents a user clicks item i_k after i_{k-1} in the session s. Therefore, each session sequence can be modeled as a directed graph $G = (N, E)$, where N is the set of nodes, e.g., $N(G) = \{i_1, i_2, i_3, \ldots, i_{|I|}\}$, and E is the set of edges, e.g., $E(G) = \{< i_1, i_3 >, < i_3, i_4 >, \ldots, < i_{k-1}, i_k >\}$. We use the traditional method, the adjacency matrix A to represent the relationship between nodes. For example, considering a session $s = \{i_1, i_3, i_4, i_2, i_5, i_2, i_3, i_5, i_3\}$, the corresponding graph are shown in Figure 1.

2) LEARNING ITEM EMBEDDINGDS ON SESSION

We present how to obtain latent feature vectors of nodes via graph neural network. Following embedding-based recommenders [39], we embed each item in a unified embedding space, the node vector $\mathbf{e}_i \in \mathbf{R}^d$ indicates the latent vector of item i learned via graph neural networks, where d is the dimensionality. As a result, we establish the following embedding table:

$$\mathbf{N_E} = [\mathbf{e}_{i_1}, \mathbf{e}_{i_2}, \mathbf{e}_{i_3}, \ldots, \mathbf{e}_{i_{|I|}}], \tag{1}$$

which can be viewed as the latent features of items to characterize their intrinsic properties. At the same time, we use the adjacency matrix to store the edge relationships. In order to deal with the occurrence of duplicates in the sequence, we assign a normalized weight to each edge, that is, calculate the number of occurrences of the edge divided by the degree of the starting node of the edge.

3) ARMA GRAPH CONVOLUTION LAYER

Recursive implementation of ARMA filters based on neural networks has been successfully applied to ARMAConv [19]. A more efficient implementation can be obtained by applying only a few recursive updates and compensating by adding nonlinear and trainable parameters. The ARMNConv filters are not learned in the Fourier space caused by a given Laplacian, but are located in the node space and have nothing to do with the underlying graph structure. This allows our model to handle graphs with unseen topologies in inductive inference tasks. Specifically, we implemented a recursive update using the Graph Convolution Skip (GCS) layer as follows:

$$\bar{\mathbf{e}}_i^{(t+1)} = \delta\left(\hat{\mathbf{L}}\bar{\mathbf{e}}_i^{(t)}\mathbf{W}_0^{(t)} + \mathbf{e}_i^{(0)}\mathbf{V}^{(t)}\right). \tag{2}$$

where $\mathbf{W}_0^{(t)}$, $\mathbf{V}^{(t)}$ are trainable parameters, and $\mathbf{e}_i^{(0)}$ are the initial node features. The $\mathbf{L} = I - L = \hat{D}^{-1/2}AD^{-1/2}$ denotes the modified Laplacian, $\mathbf{L} = \mathbf{I} - \mathbf{D^{-1/2}AD^{-1/2}}$ ($\mathbf{D}$ is the degree matrix and $\mathbf{A}$ is the adjacency matrix.) This modification is a reasonable simplification, which can compensate for small deviations introduced by the trainable parameters $\mathbf{W}_0^{(t)}$ and $\mathbf{V}^{(t)}$. And $\delta(\cdot)$ represents the logistic sigmoid function.

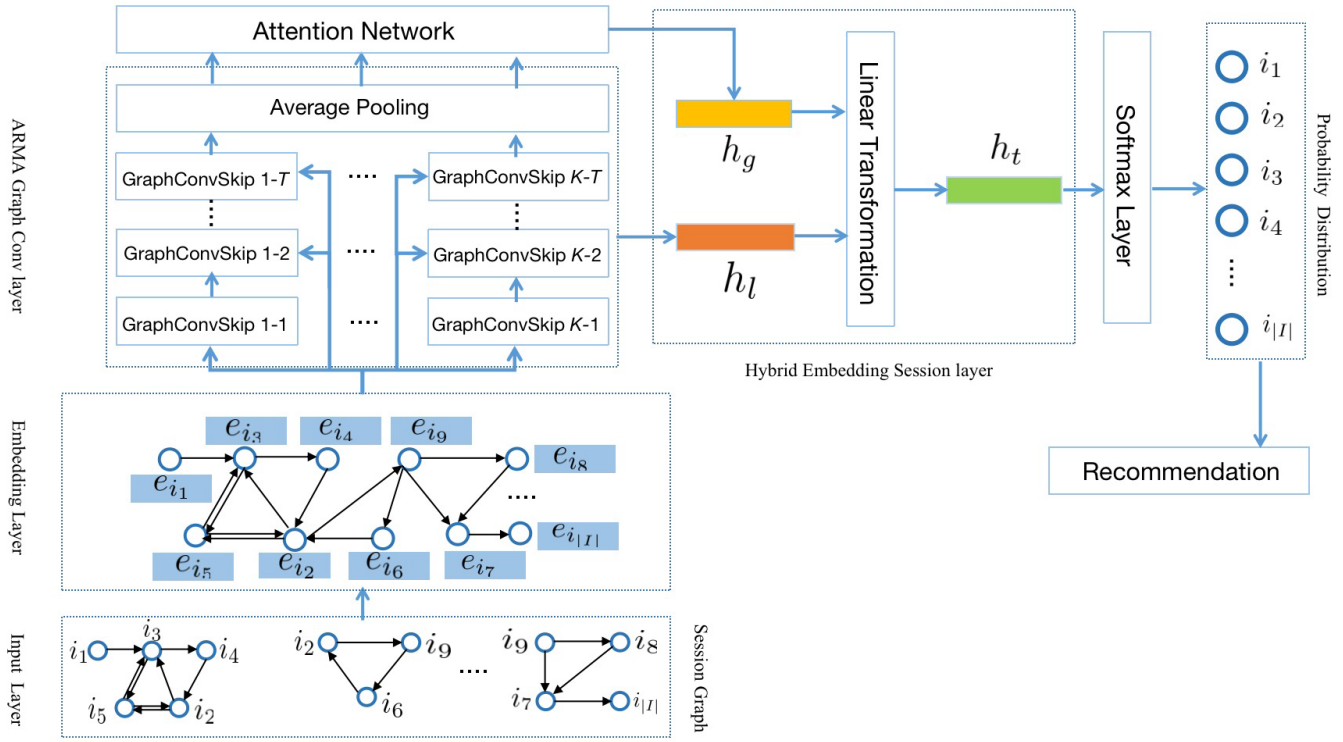


FIGURE 2. Schematic overview of AUTOMATE, which model all session sequences as session graphs. Where in the arrowed lines present the flow of information.

Each GCS layer extracts local substructure information by aggregating node information in local neighborhoods and, through the skip connection, by combining them with the original node features. The computational complexity of GCS layers is linear in the umber of edges, and can be efficiently implemented as a sparse product of $\mathbf{L}$ and $\mathbf{e}_i$.

Construct K parallel stacks, each with T GCS layers, and define the output of the ARMA convolutional layer as:

$$\bar{\mathbf{x}}_i = \frac{1}{K}\sum_{k=1}^{K}(\bar{\mathbf{e}}_i)_k^{(T)}, \tag{3}$$

where $(\bar{\mathbf{e}}_i)_k^{(T)}$ is the last output of the k-th stack. T is the number of GCS layers in a stack. We apply dropout to the skip connection of each GCS layer not only for regularization, but also to encourage diversity in the filters learned in each one of the K parallel stacks. This parallel design strategy gives ARMA filters a strong regularization capability, helps prevent overfitting, and greatly reduces spatial complexity in terms of trainable parameters.

4) CONSTRUCTING HYBRID SESSION EMBEDDING

For the task of session-based recommendation, the long-term preference has the summarization of the whole sequential behavior, while the currently short-term preference can adaptively select the important items in the current session to capture the user's main purpose. We think that the representation of the whole sequential behavior may provide useful information for capturing the user's main purpose in the current session. Therefore, to better predict the users' next click item, we plan to combine long-term preference and current interests of the session, and use this combined embedding as the session representation.

To represent each session as an embedding vector $s \in \mathbf{R}^d$, we first consider the local embedding $\mathbf{h_l}$ of session s. For a session $s = \{i_1, i_2, i_3, \ldots, i_{|s|}\}$. The local embedding of $\mathbf{h_l}$ can be simply defined as the last clicked item vector. We can represent the local session preference:

$$\mathbf{h_l} = \mathbf{x}_{i_{|s|}}. \tag{4}$$

Then, we take the all node vectors of $\mathbf{x}_{i_{|s|}}$ as the global embedding $\mathbf{h_g}$, we consider information in these embedding may have different levels of priority, we use an attention composite function that is responsible for calculating the attention based user's interests in general $\mathbf{h_g}$. The attention computation is defined as:

$$\alpha_i = \mathbf{W_1}\,\sigma(\mathbf{W_2}\,\mathbf{x}_{i_{|s|}} + \mathbf{W_3}\,\mathbf{x}_i + \mathbf{b}), \tag{5}$$

where $\mathbf{x}_{i_{|s|}} \in \mathbf{R}^d$ denotes the last-click, $\mathbf{x}_i \in \mathbf{R}^d$ denotes the i-th item node vector, $\mathbf{W_1} \in \mathbf{R}^d$ is a weighting vector, $\mathbf{W_2}, \mathbf{W_3} \in \mathbf{R}^{d\times d}$ are weighting matrices of item embedding vectors, $\mathbf{b} \in \mathbf{R}^d$ is a bias vector, and $\sigma(i)$ denotes the sigmoid function. α_i represents the attention coefficient of item $\mathbf{x}_i$ within the current session prefix. After obtaining the attention coefficients vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{|S|})$ with respect to the current session prefix s_n, the attention based user's interests

in general $\mathbf{h_g}$ with regard to the current session prefix s_n can be calculated as follows:

$$\mathbf{h_g} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i. \tag{6}$$

As shown in Figure 2, The summarization $\mathbf{h_g}$ is incorporated into $\mathbf{h_t}$ to provide a sequential behavior representation for AUTOMATE. By this hybrid embedding scheme, both the user's sequential behavior and main purpose in the current session can be modeled into a unified representation $\mathbf{h_t}$, which is the concatenation of vectors $\mathbf{h_g}$ and $\mathbf{h_l}$:

$$\mathbf{h_t} = \mathbf{W_3}[\mathbf{h_g}, \mathbf{h_l}], \tag{7}$$

where matrix $\mathbf{W_3} \in \mathbf{R}^{d \times 2d}$ compresses two combined embedding vectors into the latent space $\mathbf{R}^d$.

5) MODEL PREDICTION

After we compute the score $\hat{\mathbf{s}}$ for each candidate item $i \in I$ given session embedding $\mathbf{h_t}$ as follows:

$$\hat{\mathbf{s}} = \mathbf{h_t}^T \mathbf{x}_i. \tag{8}$$

Then, we obtain the predicted output vector of the model through the softmax layer:

$$\hat{\mathbf{y}} = softmax(\hat{\mathbf{s}}). \tag{9}$$

where $\hat{\mathbf{y}}$ denotes the recommendation probability of item i to be the next click in session s.

C. MODEL OPTIMIZATION

Similar to the previous work [3], we choose the cross entropy loss, which has been widely used to optimize recommendation models, such as DGRec [38] and GC-SAN [26]. It can measure the similarity between the true value and the predicted value. Cross entropy as a loss function also has the advantage of using the sigmoid function to avoid the problem of decreasing the learning rate of the mean square error loss function during gradient descent. The optimization function of our model is as follows:

$$loss = \sum_{i=1}^{n} \mathbf{y}_i log(\hat{\mathbf{y}}_i) + (1 - \mathbf{y}_i) log(1 - \hat{\mathbf{y}}_i) + \lambda ||\theta||^2. \tag{10}$$

where $\mathbf{y}$ denotes the one-hot encoding vector of the ground truth item, θ is the set of all learnable parameters. In order to prevent overfitting, we conduct L_2 regularization parameterized by λ on θ. In terms of parameter size, the majority of our parameter cost comes from the item embedding.

IV. EXPERIMENTS AND ANALYSIS

In this part, we first describe the datasets, compared methods, and evaluation metrics used in the experiments. Then, we compare the proposed AUTOMATE with other comparative methods. Finally, we make detailed analysis of AUTOMATE under different ways of connecting the network. We aim to answer the following research questions:

TABLE 1. Statistics of the datasets.

Dataset	items	clicks	train	test	avg.len
YOOCHOOSE1/64	16766	557248	369895	55898	6.16
YOOCHOOSE1/4	29618	8326407	5917746	55898	5.71
DIGINETICA	43097	982961	719470	60858	5.12

- **RQ1:** How does AUTOMATE perform as compared with state-of-the-art methods?
- **RQ2:** How does the different session embedding method (e.g., only local session embedding, global embedding with the attention mechanism) affect AUTOMATE?
- **RQ3:** How much impact does ARMAConv have on the model?
- **RQ4:** How much impact does the splitting of the data have on the experiment?

A. DATASET DESCRIPTION

We evaluate the effectiveness of AUTOMATE on two standard transaction datasets: YOOCHOOSE and DIGINETICA, which are publicly accessible and vary in terms of domain, size, and sparsity. We summarize the statistics of two datasets in Table 1.

YOOCHOOSE [1]: It is obtained from the RecSys Challenge 2015, which contains a stream of user clicks on an e-commerce website within 6 months. After filtering out sessions of length 1 and items that appear less than 5 times, there remains 7981580 sessions and 37483 items.

DIGINETICA [2]: It comes from CIKM Cup 2016, where only the transactional data is used in this study. After filtering out sessions of length 1 and items that appear less than 5 times. Finally the dataset contains 204771 sessions and 43097 items.

We first conducted some preprocesses over two datasets. Following the NARM [3], For YOOCHOOSE, we used the sessions of subsequent day for testing and filtered out clicks from the test set where the clicked items did not appear in the training set. For DIGINETICA, the only difference is that we use the sessions of subsequent week for testing. For an input session $s = \{i_1, i_2, i_3, \ldots, i_{|s|}\}$, we generated the sequences and corresponding labels $([i_1], V(i_2)), ([i_1, i_2], V(i_3)), \ldots, ([i_1, i_2, \ldots, i_{|s|-1}], V(i_{|s|}))$ for training on both YOOCHOOSE and DIGINETICA. The corresponding label $V(i_{|s|})$ is the last click in the current session. We also use the most recent fractions 1/64 and 1/4 of the training sequences of YOOCHOOSE. Note that some items that in the test set would not appear in the training set since we trained the model only on more recent fractions.

For each dataset, we randomly select 80% of historical interactions of each user to constitute the training set, and treat the remaining as the test set. From the training set,

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

we randomly select 10% of interactions as validation set to tune hyper-parameters.

B. EVALUATION METRICS

In our experiments, we use the following two widely-used evaluation protocols [3], [40]: Recall@K and MRR@K. By default, we set $K = 20$. We report the average metrics for all users in the test set.

- **Recall@20**: Recall@K represents the proportion of test cases which has the correctly recommended items in a top K position in a ranking list. In this paper, Recall@20 is used for all the tests, defined as:

$$Recall@20 = \frac{n_{hit}}{N}, \tag{11}$$

  where N denotes the number of test data in the Session-based Recommender systems, n_{hit} denotes the number of cases which have the desired items in top K ranking lists, a *hit* occurs when t appears in the top K position of the ranking list of $|I|$.
- **MRR@20**: The correct ranking of search results values in search results to evaluate the performance of the search system. The reciprocal rank is set to zero if the rank is above 20.

$$MRR@20 = \frac{1}{|I|}\sum_{t=1}^{|I|}\frac{1}{rank(i_t)}. \tag{12}$$

  where $rank(i_t)$ is for the t-th item. The MRR is a normalized score of range [0, 1], an increase in its value reflects that the majority will appear higher in the ranking order of the recommendation list, which indicates a better performance of the corresponding recommender system.

These two indicators describe several key features that should be included in the recommendation system from different perspectives. The larger the values of the above two indicators, the better the model.

C. BASELINE

The following models, including the state-of-art and closely related work, are used as baselines to evaluate the performance of the proposed AUTOMATE model:

- POP: The model recommends top-N rank items based on popularity in training data. It is a simple and strong baseline in certain domains.
- S-POP: This baseline recommends the top-N frequent items in the training set and in the current session respectively.
- Item-KNN [17]: A traditional item-to-item model, which recommends items similar to the existing items based on calculating the cosine similarity between candidate item A and existing item B.
- FPMC [1]: A state-of-the-art hybrid model combing matrix factorization and first-order Markov chain for next-basket recommendation.
- BPR-MF [39]: This is matrix factorization optimized by the Bayesian personalized ranking (BPR) loss, which exploits the user-item direct interactions only as the target value of interaction function.
- GRU4Rec [2]: The model uses recurrent neural networks for session-based recommendations, it utilizes a session-parallel mini-batch training process and also employs ranking-based loss functions during the training.
- NARM [3]: The model adds an attention mechanism to capture the user's main purpose and sequential behavior to the RNN-based.
- STAMP [27]: The model capable of capturing users' general interests from the long-term memory of a session context, whilst taking into account users' current interests from the short-term memory of the last-clicks.
- SR-GNN [28]: This model aggregates separated session sequences into graph structure data and applies GNN to generate latent vectors of items and then represent each session through traditional attention network.
- GC-SAN [26]: This is a graph contextualized self-attention model, which utilizes both graph neural network and self-attention mechanism, for session-based recommendation.
- GACOforRec [29]: The algorithm uses the GCNs as the model basis, abstracting the actual role of user preferences in the entire application scenario, and applying ConvLSTM and ON-LSTM to expand the algorithm's ability. So that it can handle users' long-term and stable preferences, and preserve the hierarchy of potential preferences.

D. PARAMETER SETTINGS

We implement our AUTOMATE model in Pytorch and will release our code, data, and parameter settings upon acceptance. All experiments are completed in the following environment: Python3.6, Pytorch1.0, GeForce RTX2080Ti GPUs. The embedding size is fixed to 100 for all models. In the setting of the model parameters, all parameters are initialized using a Gaussian distribution with a mean of 0 and a standard deviation of 0.1. The mini-batch Adam optimizer is exerted to optimize these parameters, the learning rate is tuned amongst {0.0001, 0.0005, 0.001, 0.005}, the coefficient of L2 normalization is searched in $\{10^{-5}, 10^{-4}, \ldots, 10^{1}, 10^{2}\}$, and learning rate decay rate in $\{0.0, 0.1, \ldots, 0.8\}$. For ARMAConv, we tune the layer T and stack K in $\{1, 2, \ldots, 9\}$. Moreover, early stopping strategy is performed, i.e., premature stopping if Recall@20 on the test data does not increase for 10 successive epochs.

E. PERFORMANCE COMPARISONS (RQ1)

To demonstrate the recommendation performance of our model AUTOMATE, we start by comparing the performance of all the methods. The performance of different algorithms is summarized in Table 2. Please note that, we have insufficient available memory to initialize FPMC, so the performance on YOOCHOOSE 1/4 is not reported. We have the following observations:

TABLE 2. Comparison of different obfuscations in terms of their transformation capabilities.

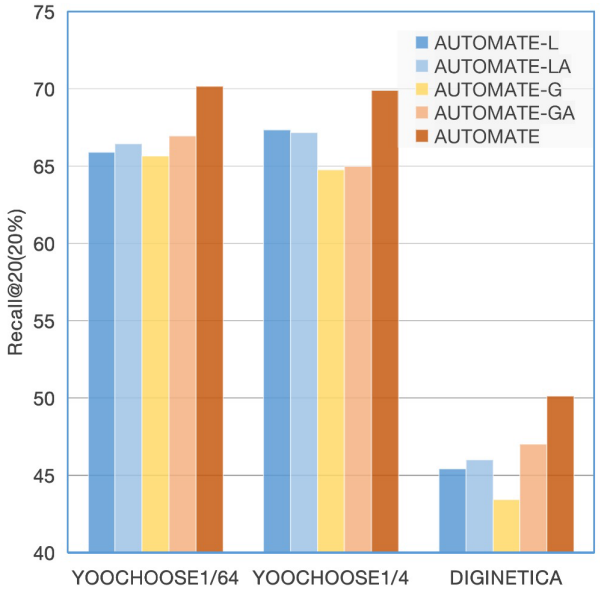
Model Class	Methods	YOOCHOOSE1/64		YOOCHOOSE1/4		DIGINETICA	
		Recall@20	MRR@20	Recall@20	MRR@20	Recall@20	MRR@20
Standard baseline	POP	6.71	1.65	1.33	0.30	0.89	0.20
	S-POP	30.44	18.35	27.08	17.75	21.06	13.68
Traditional	Item-KNN	56.60	21.81	52.31	21.70	35.75	11.57
	FPMC.	45.62	15.01	-	-	26.53	6.95
	BPR-MF	31.31	12.08	3.40	1.57	5.24	1.89
Neural Networks	GRU4Rec	60.64	22.89	59.53	22.60	29.45	8.33
	NARM	68.32	28.63	69.73	29.23	49.70	16.17
	STAMP	68.74	29.67	69.81	29.69	45.64	14.32
	SR-GNN	70.03	30.08	68.72	29.88	49.52	16.58
	GC-SAN	69.64	29.85	69.17	**29.92**	49.12	15.44
	GACOforRec	68.79	29.38	67.66	28.13	47.83	14.28
Ours Model	AUTOMATE	**70.15**	**30.72**	**69.89**	29.16	**50.11**	**16.72**

- POP achieves poor performance on three datasets. This indicates that the non-personalized Popularity-based methods is problematic. SPOP consistently outperforms POP across all cases, indicating the importance of session context information.
- The performance of traditional methods such as Item-KNN, MPMC and BPR-FM are not competitive, as they only outperform the naive POP and S-POP model. Item-KNN and MPMC achieve better performance than BPR-MF, Item-KNN utilizes the similarity between items in the session and FPMC is based on first-order Markov Chain. The results show that making recommendations solely based on co-occurrence popularity of the items (POP), or simply taking transitions over successive items could be very problematic in making accurate recommendations.
- We can see that five RNN-based methods consistently outperform the traditional baselines on all three data sets, which demonstrates that RNN-based models are good at dealing with sequence information in sessions. GRU4Rec leverages the recurrent structure with GRU as a special form of RNN to capture the user's general preference and improves the performances. STAMP improves the short-term memory by utilizing the last-clicked item, STAMP performs better than GRU4Rec, which indicates the effectiveness of short-term behavior for predicting the next item problem. NARM not only models the sequential behavior using RNN with GRU units but also uses attention mechanism to capture main purpose, which indicates the importance of main purpose information in recommendations. Further more, GACOforRec model combines ConvLSTM and ON-LSTM methods, focusing on short-term preferences and discrete preferences of volatility. We can see that the experimental results on the YOOCHOOSE dataset are basically the same as SR-GNN, which can further explain that the user's historical preferences will affect the user's next click.
- SR-GNN is the closest approach to our proposed approach and achieves the best performance among the baselines. This model aggregates separated session sequences into graph structure data, then through traditional attention network represents each session, thereby capturing more complex and implicit connections between user clicks.
- GC-SAN upgrades the general attention mechanism to a self-attention mechanism based on the SR-GNN model to model long-range dependencies regardless of the distance. And we can see that GC-SAN performs better on a larger YOOCHOOSE1/4 of the dataset. It further shows that the prediction results of the model are more advantageous on larger data sets.
- AUTOMATE produces the best performance on almost all data sets. In our model, by stacking two ARMAConv to the graph neural network layer, not only can the separated session sequences be aggregated into graph structure data, but also the graph neural network can be used to generate potential item vectors. This enables our AUTOMATE to process graphs with unknown topologies in inductive inference tasks to better predict items that users might like. And this is also the first time that the traditional ARMA method for serialized data is combined with a graph convolutional neural network. As we can see, our method achieves the best performance among all the methods on both YOOCHOOSE1/64 and DIGINETICA datasets in terms of Recall@20 and MRR@20. These results demonstrate the efficacy and validity of AUTOMATE for session-based recommendation.

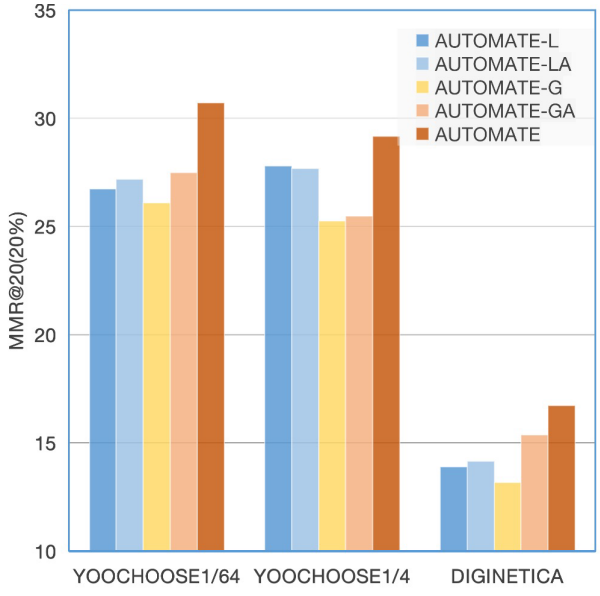
F. STUDY OF AUTOMATE (RQ2)

The proposed AUTOMATE method is flexible in constructing embedding strategy between session in the graph. The results of methods with four different embedding strategies are given in Figure 3. (1) local embedding only (AUTOMATE-L), (2) local embedding with the attention mechanism(AUTOMATE-LA), (3) global embedding only (AUTOMATE-G), and (4) global embedding with the attention mechanism (AUTOMATE-GA).

As can be seen from Figure 3, the local and global hybrid embedding method AUTOMATE achieves the best results on all three data sets, which proves the importance of combining current session interests with the long-term preferences.



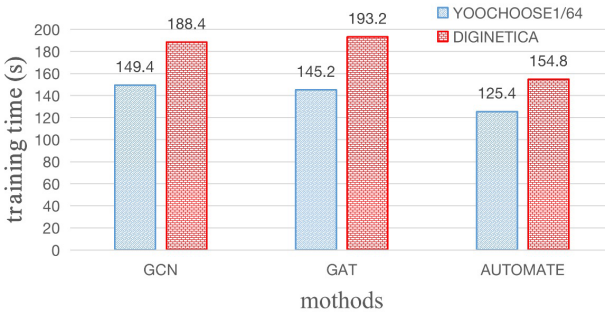
(a) Recall@20



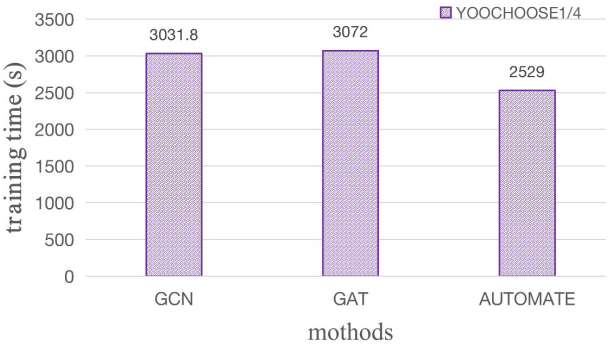
(b) MMR@20

FIGURE 3. The performance of different session represention.

In addition, as we have seen, (1) AUTOMATE-L performance is slightly better than AUTOMATE-LA on the three data sets, which indicates that the attention mechanism has little effect on the local processing of shorter sessions. (2) The performance of AUTOMATE-GA is better than AUTOMATE-G, which indicates that the session may contain some noisy behavior and cannot be handled separately. Combining the above two points, we can know that the attention mechanism helps to extract important data from the session data. Behavior to build long-term preferences is not sensitive to the construction of short-term preferences. Note that AUTOMATE-L performance, which is only represented by a partial session, still outperforms AUTOMATE-G and



(a) Yoochoose1/64 and Diginetica



(b) Yoochoose1/4

FIGURE 4. Efficiency of three methods on YOOCHOOSE and DIGINETICA datasets.

TABLE 3. Impact of ARMAConv.

DataSet	Models	Recall@20	MRR@20
YOOCHOOSE1/64	GCN	67.68	27.68
	GAT	64.88	25.75
	AUTOMATE	70.15	30.72
YOOCHOOSE1/4	GCN	68.88	27.81
	GAT	65.71	26.58
	AUTOMATE	69.89	29.16
DIGINETICA	GCN	48.17	16.33
	GAT	44.98	15.16
	AUTOMATE	50.11	16.72

AUTOMATE-GA, indicating that current interest is critical to the user's next click, and combining current interests with long-term preferences helps Improve session-based recommendations.

G. IMPACT OF ARMAConv (RQ3)

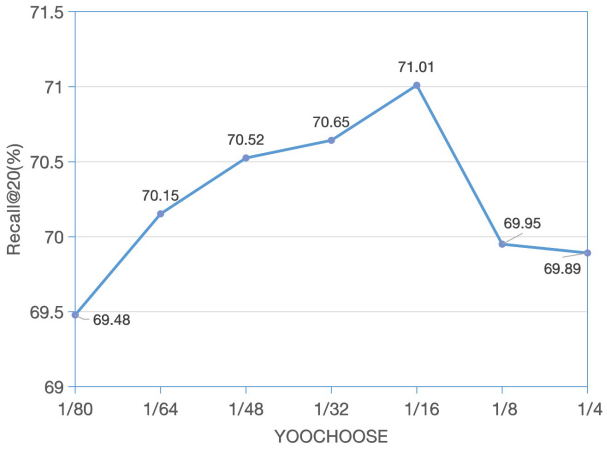
Although we can implicitly infer from Table1 that the effect of adding traditional ARMAConv is significantly improved, we still want to further verify the role of ARMAConv in AUTOMATE. We tried the following changes: (1) We removed the ARMAConv module from AUTOMATE and replaced it with the Graph Convolutional Network (GCN). (2) We removed the ARMAConv module from AUTOMATE and replaced it with the Graph Attention Networks (GAT). Table 3 shows a comparison of using and not using ARMAConv.

From Tables 2 and 3, we can see that without ARMAConv, GCN still performs better than all traditional models in the three data sets. In the neural network-based models, except for the three hybrid models mentioned later, almost all models are surpassed. We can also see that the graph convolutional network with attention mechanism (GAT) does not perform as well as GCN in Recall @ 20 or MMR @ 20, which is inconsistent with the original work of GCN and GAT. After consulting the data and experiments, we think that there are two reasons: (1) GCN is a first-order local approximation of spectral convolution. It is a multi-layer graph convolutional neural network. Each convolution layer processes first-order neighborhood information, and stacking several convolution layers can achieve multi-order neighborhood information transfer. GAT is a spatial domain convolution. Each vertex in the graph performs an attention operation relative to its neighbors, making it completely independent of the structure of the graph. However, the structure characteristics of the graph are also lost, and the effect may be very poor. (2) In addition to the visible nodes in the user's click behavior, there may be potential hidden variables. The processing method of GCN can improve the generalization ability of the model and make the results better. In addition, from Table 3, we can see that on the YOOCHOOSE data set, as the amount of data increases, the GAT effect improves significantly. It is also proved once again that the attention mechanism helps to extract important behavioral data from session data to establish long-term preferences, but is not sensitive to the construction of short-term preferences.

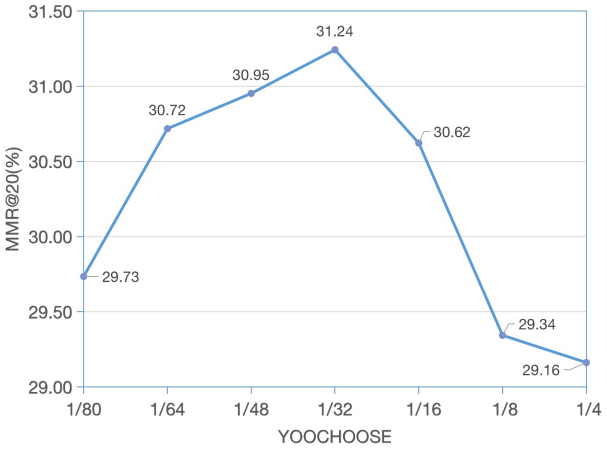
In addition, we use the training speed (training time of one epoch) as a comparison standard to explore whether our proposed method AUTOMATE has a performance advantage over the other two methods. The running time of YOOCHOOSE1/4 is very different from the other two data sets. We show the running time of three data sets separately in Figure 5 to make the results more intuitive. We can see that GCN and GAT run on the three data at the same time, and GCN consumes a little less time. This may be because GAT uses a deep architecture, which contains multiple levels of self-attention. AUTOMATE takes the least time, which may be because it applies parallel training on data at the ARMAConv layer, which can save time.

H. IMPACT OF YOOCHOOSE DATA VOLUME (RQ4)

For a fair comparison, we use the latest scores of the YOCHOCHOOSE training sequence 1/64 and 1/4 for the experiment according to NARM [3]. To further verify the impact of the size change of the data set on the experimental results, we will further the data. Split into YOCHOCHOOSE1/80, YOCHOCHOOSE1/48, YOCHOCHOOSE1/32, YOCHOCHOOSE1/16 for verification. Similarly, we will filter out sessions with a length of 1 in the data set and items with fewer than 5 occurrences. The test set includes follow-up workdays related to the training set, and we filter out clicks (items) that do not appear in the training set. Figure 4 shows the results of the different degrees of resolution of YOCHOCHOOS. As



(a) Recall@20



(b) MMR@20

FIGURE 5. The performance of different size datasets.

the amount of data increases, the effects on Recall@20 and MMR@20 are significantly improved, reaching a maximum on YOCHOCHOOSE1/16. However, it has declined since YOCHOCHOOSE1/4, which indicates that the recommendation results will increase as the data set increases, but not as large as possible. As shown in the recommendation model proposed in [5], we also need to consider the change of user behavior over time, that is, the longer the time, the less impact this behavior has on the user.

V. CONCLUSION

In this work, we devised a new session-based recommendation framework AUTOMATE, which can effectively capture the complex relationships between items, sessions of different users, and sessions of the same user at different time periods. The key of AUTOMATE is to use the ARMAConv layer, based on which we combine long-term preferences with the current interest in the session to obtain the graph transfer signal. Extensive experimental comparisons of different dimensions on two real-world datasets prove the rationality and effectiveness of injecting the items data of session sequence

structure into the learning process of graph neural networks. In future, We will focus on trying more hybrid graph convolutional network model methods for representation learning on large-scale graph data to achieve more effective and interpretable recommendations.

REFERENCES

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

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

[3] 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. (CIKM)*, 2017, pp. 1419–1428.

[4] L. Hu, L. Cao, S. Wang, G. Xu, J. Cao, and Z. Gu, "Diversifying personalized recommendation with user-session context," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 1858–1864.

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

[6] Z. Wang, C. Chen, K. Zhang, Y. Lei, and W. Li, "Variational recurrent model for session-based recommendation," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2018, pp. 1839–1842.

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

[8] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2018, pp. 197–206.

[9] P. Liu, S. Chang, X. Huang, J. Tang, and J. C. K. Cheung, "Contextualized non-local neural networks for sequence learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, Jul. 2019, pp. 6762–6769.

[10] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1257–1264.

[11] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[12] B. Yang, Y. Lei, J. Liu, and W. Li, "Social collaborative filtering by trust," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 8, pp. 1633–1647, Aug. 2017.

[13] Y. Koren, "Collaborative filtering with temporal dynamics," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 447–456.

[14] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell, "Temporal collaborative filtering with Bayesian probabilistic tensor factorization," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2010, pp. 211–222.

[15] A. Zimdars, D. M. Chickering, and C. Meek, "Using temporal data for making recommendations," in *Proc. 17th Conf. Uncertainty Artif. Intell.*, 2001, pp. 580–588.

[16] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, "Using sequential and non-sequential patterns in predictive Web usage mining tasks," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2002, pp. 669–672.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web (WWW)*, 2001, pp. 285–295.

[18] S. K. Narang, A. Gadde, and A. Ortega, "Signal processing techniques for interpolation in graph structured data," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 5445–5449.

[19] F. M. Bianchi, D. Grattarola, C. Alippi, and L. Livi, "Graph neural networks with convolutional ARMA filters," 2019, *arXiv:1901.01343*. [Online]. Available: http://arxiv.org/abs/1901.01343

[20] M. Fey and J. Eric Lenssen, "Fast graph representation learning with PyTorch geometric," 2019, *arXiv:1903.02428*. [Online]. Available: http://arxiv.org/abs/1903.02428

[21] R. Levie, E. Isufi, and G. Kutyniok, "On the transferability of spectral graph filters," 2019, *arXiv:1901.10524*. [Online]. Available: http://arxiv.org/abs/1901.10524

[22] N. Tremblay, P. Gonçalves, and P. Borgnat, "Design of graph filters and filterbanks," in *Proc. Cooperat. Graph Signal Process.*, 2018, pp. 299–324.

[23] L. Zheng, C.-T. Lu, F. Jiang, J. Zhang, and P. S. Yu, "Spectral collaborative filtering," in *Proc. 12th ACM Conf. Recommender Syst.*, 2018, pp. 311–319.

[24] M. Ludewig and D. Jannach, "Evaluation of session-based recommendation algorithms," *User Model. User-Adapted Interact.*, vol. 28, nos. 4–5, pp. 331–390, Dec. 2018.

[25] C. Wu and M. Yan, "Session-aware information embedding for E-commerce product recommendation," in *Proc. ACM Conf. Inf. Knowl. Manage. (CIKM)*, 2017, pp. 2379–2382.

[26] 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-based recommendation," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 3940–3946.

[27] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, "STAMP: Short-term attention/memory priority model for session-based recommendation," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 1831–1839.

[28] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, Jul. 2019, pp. 346–353.

[29] M. Zhang and Z. Yang, "GACOforRec: Session-based graph convolutional neural networks recommendation model," *IEEE Access*, vol. 7, pp. 114077–114085, 2019.

[30] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural network based language model," in *Proc. 11th Annu. Conf. Int. Speech Commun. Assoc.*, 2010, pp. 1–24.

[31] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, and T.-Y. Liu, "Sequential click prediction for sponsored search with recurrent neural networks," in *Proc. 28th AAAI Conf. Artif. Intell.*, Jun. 2014, pp. 1369–1375.

[32] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," in *Proc. 13th AAAI Conf. Artif. Intell.*, Jun. 2016, pp. 194–200.

[33] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, "A dynamic recurrent model for next basket recommendation," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2016, pp. 729–732.

[34] F. Scarselli, M. Gori, A. Chung Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

[35] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*. [Online]. Available: http://arxiv.org/abs/1609.02907

[36] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*. [Online]. Available: http://arxiv.org/abs/1710.10903

[37] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for Web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 974–983.

[38] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang, "Session-based social recommendation via dynamic graph attention networks," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, Jan. 2019, pp. 555–563.

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

[40] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.

HUANWEN WANG received the B.E. degree in computer science and technology from Henan Normal University, China, in 2016. She is currently pursuing the Ph.D. degree with the College of Computer Science and Electronic Engineering, Hunan University, China. Her research interests include deep learning, personalized recommendation, and interpretable recommendation.

GUANGYI XIAO received the B.E. degree in software engineering from Hunan University, China, in 2006, and the M.S. and Ph.D. degrees in software engineering from the University of Macau, in 2009 and 2015, respectively. He is currently an Associate Professor with the College of Computer Science and Electronic Engineering, Hunan University. His principal research interests include concept representation, semantic integration, interoperation, and collaboration systems, mainly applied to the fields of e-commerce, e-marketplace, e-banking, and virtual world.

NING HAN received the M.S. degree in computer science and technology from Hunan University, China, in 2018, where he is currently pursuing the Ph.D. degree with the College of Computer Science and Electronic Engineering. His research interests include information retrieval, recommender systems, and multimedia analytics.

HAO CHEN received the B.E. and M.S. degrees in software engineering from Hunan University, China, in 2000 and 2004, respectively, and the Ph.D. degree in computer science and technology from Central South University, China, in 2012. He is currently a Full Professor and a Ph.D. Supervisor with the College of Computer Science and Electronic Engineering, Hunan University. His research interests include web mining, personalized recommendation, and big data technology.