IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 22, 2021, accepted March 21, 2021, date of publication March 30, 2021, date of current version April 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3069846

Hierarchical Context-Aware Recurrent Network for Session-Based Recommendation

YOUFANG LENG AND LI YU
School of Information, Renmin University of China, Beijing 100872, China
Corresponding author: Li Yu (buaayuli@ruc.edu.cn)

ABSTRACT Recently, session-based recommendations are becoming popular to explore the temporal characteristics of customers' interactive behaviors. The user's behavior in the session not only contains the item sequence but also rich context information such as how the user navigates to the item, the operation type, and the dwell time on the item, which may impact users' next actions. How to incorporate these contextual features effectively for session-based recommendations remains a great challenge. In this paper, we divide them into two hierarchies: session-context at the macro-level and event-context at the micro-level, and then propose a **Hi**erarchical **C**ontext-**a**ware **R**ecurrent Network (HiCAR) by incorporating both users' micro-interaction with the item and the two-hierarchy contexts, wherein a Session Context Learning module with the n-way hybrid strategy is adopted to model multi-feature interactions in the session-context. Moreover, an Event Context Learning module consists of TIME-LSTM with a time gate is designed to model the sequential behavior with event-context. By experimenting on two real-world datasets, we find that our HiCAR model outperforms state-of-the-art baselines on both datasets, which demonstrates its advantages in modeling users' sequential behaviors and contexts simultaneously.

INDEX TERMS Session-based recommendation, context-aware recommendation, sequential behavior learning, recurrent neural network.

I. INTRODUCTION

Session-based Recommender systems (SRS) are an important component of modern commercial online systems, usually used for improving user experiences by making suggestions based on user behavior in browser sessions, and the recommender's task is to predict users' next actions based on the sequence of the actions in the current session [1].

Recent studies have highlighted the importance of using recurrent neural networks (RNN) in a wide variety of recommender systems, among which the application in session-based recommendations has led to significant progress [1]–[5]. For example, Hidasi *et al.* [1] apply recurrent neural networks with Gated Recurrent Units (GRU) to model the item sequence in a session. Tan *et al.* [2] propose a data augmentation technique to improve the performance of the RNNs for session-based recommendation. Li *et al.* [3] propose an RNN based encoder-decoder model (NARM), which takes the last hidden state from the RNN as the sequential behavior, and uses the hidden states of previous clicks for attention computation to capture the main purpose (general interests) in a given session. Liu *et al.* [4] proposed to use the attention mechanism to take into account users' current interests from the short-term memory of the last-clicks and achieves state-of-the-art performance.

The above RNN-based methods show promising improvements over traditional recommendation approaches, but they only consider modeling the session as a chronological sequence of items, without explicitly considering the purchase context that affects the user's next behavior. In recent years, some works, such as [6]–[10], have integrated contextual information to SRS to enhance the recommendation performance. But they mainly deem all contexts as global features and integrate them into the sequential models, without subdividing the context information.

Therefore, in this paper, we argue that the contexts can be further divided into two hierarchies: **session context** and **event context**, as shown in Figure 1, and modeling them separately can lead to more accurate recommendations. (1) Firstly, the session-level context refers to the macro circumstance where a session happens, which mainly contains contextual factors such as time, address, weather, season, etc. Session-context should be considered in an SRS because different contexts can bring uncertainty and dynamics to

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

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

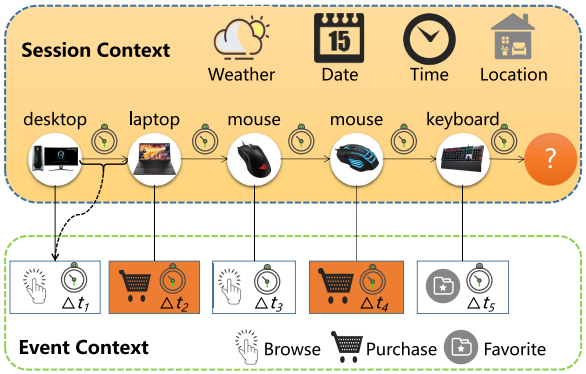

FIGURE 1. An illustration of the two-hierarchy contexts. First, the environment surrounding an entire session is session-context, such as weather, location, time, etc. Second, a session contains a sequence of events, each event also has its unique context, called event-context, such as the page where the event occurred, event type, dwell time, etc.

the session evolution and thus make a huge difference in a user's real-time preferences. For example, if a user buys a pumpkin at a normal time and it is better to recommend other vegetables such as potatoes. However, when the user buys a pumpkin on Halloween, we need to consider the unique context and it is more appropriate to recommend Halloween costumes for him. (2) Then, the event-level context refers to the micro circumstance of each of the user's click behavior on the item, such as how the user locates the item, what activities the user conducts on the item (e.g., reading the comments, carting, browse, purchase and collect) and how long the user stays with the item, etc. These micro event-contexts should also be considered because different event-contexts can bring different subsequent behaviors. For example, (a) if a user *browsed* a DELL game notebook, the recommender system can recommend another brand such as an HP game notebook for comparison. However, if the user has just *purchased* a DELL game notebook, then suggesting a game mouse or mechanical keyboard to him is more suitable. (b) Longer dwell time on an item suggests more interest in the item than shorter dwell time. (c) Deleting an item from a collection and then adding another item means that the user's interest has shifted. Existing session-based recommendation methods often ignore the hierarchical context information.

To tackle the above problem, we propose a novel neural networks framework, namely **Hi**erarchical **C**ontext-**a**ware **R**ecurrent Network (**HiCAR**), which incorporates both the session-context and event-context for a more accurate session-based recommendation. Specifically, our model includes three main components for learning the two-hierarchy contexts. (1) The first one is Session-context Learning (**SCL**), which captures high-order non-linear interactions between each context factor by the n-way hybrid pooling strategy. (2) The second one is Event-context Learning (**ECL**), which equips Time-LSTM with a fine-tuning time gate to perceive the behavior sequence pattern under a particular event-context. (3) Finally, the Combination and Recommendation (**CR**) component combines both of them to a unified representation and computes recommendation scores for each candidate item. The top k of the ranking list is the final recommendation list. The main contributions of this work are summarized as follows:

(1) Compared to existing SRS that only model item sequences, we identified some key contextual factors surrounding item sequences to enhance session-based recommendations. These contexts have two hierarchies: the macro-level context of the entire session, i.e. Session-context, and the micro-level context of each action in the session, which is Event-context.

(2) We propose a hybrid HiCAR model, which contains two main elaborately designed components, namely SCL and ECL, to seamlessly incorporate session context and event context into SRS.

(3) We conduct extensive experiments on two real-world datasets. The results show that HiCAR outperforms other state-of-the-art methods from various aspects. Furthermore, we designed ablation experiments to verify the validity of our n-way hybrid based SCL component and TIME-LSTM based ECL component.

In the rest of the paper, we first introduce some related works in Section II, then detail our proposed HiCAR model in Section III. The experiments and results will be presented in Section IV. Finally, we conclude this paper and discuss the future work in Section V.

II. RELATED WORK

Session-based recommendation has received a lot of attention in recent years and has been widely researched from both research and business perspectives. In this section, we conducted a literature review on SRS from three aspects: *traditional methods*, *deep learning*-based approaches and *context-aware enhanced* methods.

A. TRADITIONAL METHODS

Typically, traditional recommendation methods can be divided into two types: general methods and sequential methods.

1) GENERAL METHODS

such as collaborative filtering (CF) are based on a user-item matrix extracted from the interaction history between users and items. For example, the Matrix Factorization (MF) approach [11] factorized the user-item matrix to estimate the users' latent vector. Another approach is neighborhood methods [12], [13] which try to make recommendations based on item similarities calculated from the co-occurrences of items in sessions. Though these methods have proven to be effective and are widely employed, they broke down the basic transaction unit (e.g., a session or basket) into multiple records (e.g., user-item interaction pairs) and missed the sequential features which contain users' preference shift through time.

2) SEQUENTIAL METHOD

To address the above issues, sequential methods based on Markov Chains (MC) are proposed which utilizes users'

history sequential behaviors to predict their next action [14]–[16]. For example, Zimdars *et al.* [14] proposed a sequential recommender based on Markov chains and investigate how to extract sequential patterns to learn the next state using probabilistic decision-tree models. Shani *et al.* [15] present a Markov Decision Processes (MDP) aiming to provide recommendations in a session-based manner, the simplest MDP boil down to first-order Markov chains where the next recommendation can be simply computed through the transition probabilities between items. Factorizing Personalized Markov Chains (FPMC) [16] is a hybrid model that combines the power of MF and MC to model both general interests and sequential behavior between every two adjacent baskets for the next basket recommendation. Nevertheless, all the MC-based methods have the same deficiency that these recommenders can only model local sequential behaviors between every two adjacent actions and some of which may be irrelevant.

B. DEEP LEARNING BASED SRS

Deep learning methods, especially Recurrent Neural Network, have proven to be effective in modeling sequential data recently [17]. Inspired by recent advances in the natural language processing area [18], some deep learning-based methods have been developed and some of which represent state-of-the-art in session-based recommendation researches [1]–[5], [19]. Hidasi *et al.* [1] employ GRU to model session data, which learns deep representation directly from previous clicks in the given session and provides recommendations of the next action. Tan *et al.* [2] proposed a data augmentation technique to further improve the performance of GRU4Rec for the session-based recommendation. Li *et al.* [3] proposed an RNN based encoder-decoder model (NARM), which takes the last hidden state of RNN as the representation of the sequential behavior, and uses the attention computation to capture the main purpose (general interests) in a given session. Liu *et al.* [4] proposed to adopt the attention mechanism to take into account users' current interests from the short-term memory of the last-clicks.

In recent years, the success of Graph Neural Networks (GNN) in modeling graph data has ignited its research on SRS. Song *et al.* [20] introduced graph attention networks into the social recommendation for modeling both dynamic user interests and context-dependent social influences. Wu *et al.* [21] proposed an SR-GNN model which models the transition relationship among items in the ongoing session by a gated graph neural networks (GGNN) to generate accurate session representations. Xu *et al.* [22] further extend SR-GNN with a self-attention mechanism to obtain contextualized non-local representations for producing recommendations. Qiu *et al.* [23] proposed a weighted graph attention network (WGAT) based approach for generating item representations, which are then aggregated by a Readout function as the user preference. Pan *et al.* [24] apply a star graph neural network (SGNN) and a highway network (HN) to model the complex transition relationship between items in a session.

Although the previous attempts mentioned above mainly focused on how to explore sequential behavior patterns from sessions, in which the RNN based methods aim to learn the sequential relationship between items, and the GNN based methods establish the state transition relationship based on the item sequences, they did not model the hierarchical contextual information in an ongoing session that affects the user's next choice.

C. CONTEXT-AWARE ENHANCED SRS

Recently, various approaches [6]–[10] have been proposed to enhance SRS by integrating contextual information. Manotumruksa *et al.* [6] proposed a contextual attention recurrent architecture (CARA) that leverages contextual information associated with the sequences to capture the users' dynamic preferences. Souza *et al.* [7] combine the user context features and Recurrent Neural Networks (RNN) for session-based news recommendation. In order to exploit the impact of various contextual information, a stacked RNN was proposed by [8], in which one layer is to capture the input context, and the other layer is used to capture the temporal context. Wu *et al.* [9] incorporated graph knowledge into SRS as external knowledge and generate the session's external context. Cui *et al.* [10] proposed a Hierarchical Contextual Attention-based (HCA) network for the sequential recommendation, in which the context is first summarized from several adjacent items by attention mechanism, and then is feed to the RNN to predict the next item. Wang *et al.* [25] exploit global-level item-transitions over all of the sessions to learn the global-level contextual information for SRS.

The context-aware enhanced methods listed above mainly focus on the global context of the session and do not further subdivide the context information. Although some methods, such as HCA [10], has a two-hierarchical context structure, its context refers to the summary of several adjacent items, which is essentially different from our HiCAR model which contains session-context and event context.

III. METHODS

In this section, we formulate the task of session-based recommendation and then introduce the proposed HiCAR model in detail.

A. NOTATIONS AND PROBLEM FORMULATION

Let $I = \{i_1, i_2, \ldots, i_{|I|}\}$ be the set of items, where $|I|$ is the number of items. All the sessions in a dataset form the set $S = \{s_1, s_2, \ldots, s_{|S|}\}$. For a given session s, a typical session-based recommender is to predict the following item i_{t+1} based on the prefix of the session $s = \{i_1, i_2, \ldots, i_t\}$. In our model, we extend the typical methods by adding two layers of context-aware components. Firstly, in the session layer, the environment in which each session occurs, i.e., session-context, is denoted as $C^s =< w, t, l >$, which indicates that the session occurred in location l at time t on weather w.

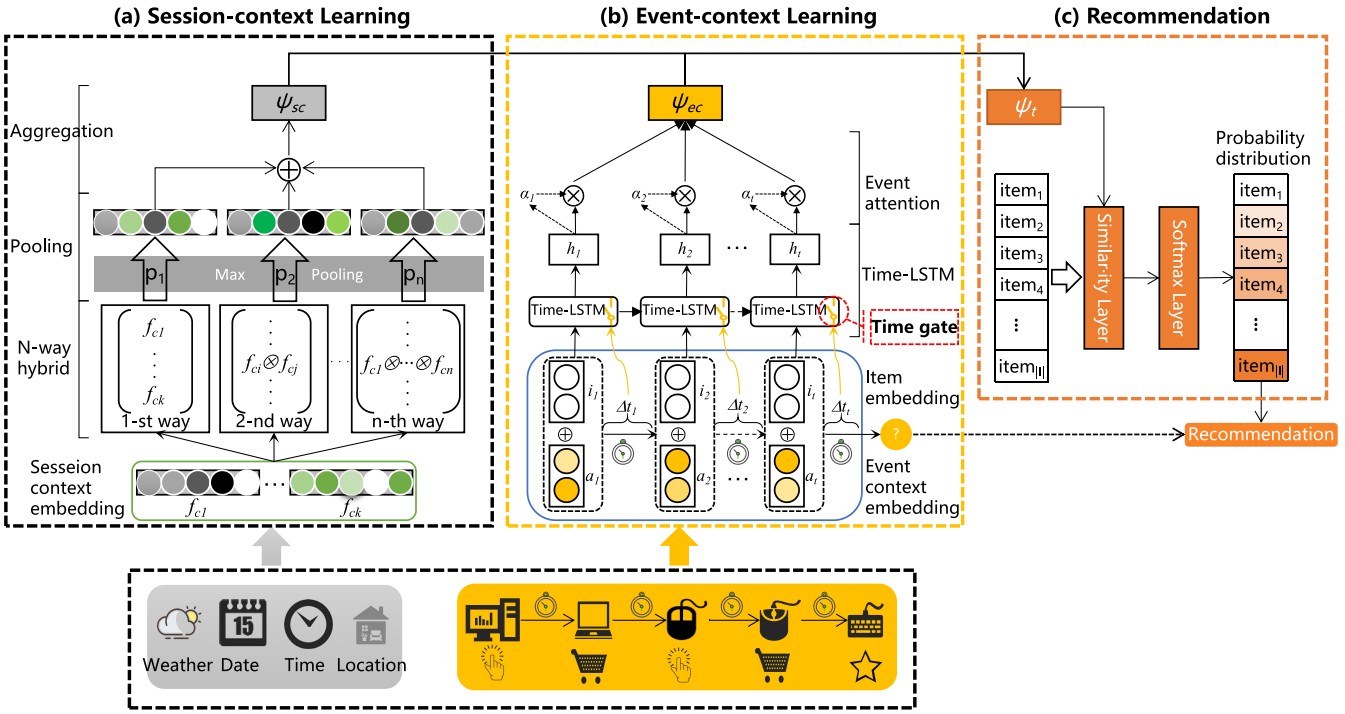

FIGURE 2. The HiCAR model architecture. HiCAR contains three main components: (a) *Session-Context Learning* (SCL), (b) *Sequential Behavior Learning under Event-Context* (ECL) and (c) *Combination and Recommendation* (CR).

Secondly, at the event layer, we add event contexts upon item sequence $s = \{i_1, i_2, \ldots, i_t\}$ and expand it to an event sequence $s = \{< a_1, i_1, d_1 >, \ldots, < a_t, i_t, d_t >\}$. Each event-context $C^E =< a_t, i_t, d_t >$ in the sequence is a triplet that denotes that at time step t the user performs activity a_t on item i_t with d_t dwell time. For example, a user spent 2 minutes browsing a gaming mouse.

Definition 1 (Session): A session refers to the interaction between the user and the system in a short period, such as an hour. Usually, a session has an expiration time, for example, 30 minutes. If the user does not interact with the system for over 30 minutes, the current session expires. When the user is active again, a new session is usually established by the system.

Definition 2 (Session Context): The session-context C^{s_n} is the global circumstances in which the session happens. One session corresponds to one session-context, and one session-context contains multiple session features, such as the location where the session occurs, the weather, whether it is a holiday, etc. Note that we only give a general definition of session-context here, the specific features used in the experiment are determined based on different data sets. In this paper, we have extracted features such as holidays, weeks, and hours to form the session-context, as detailed in the datasets section IV-A.

Definition 3 (Event Context): Taking s_n as the current session for making recommendations, it contains a sequence of interactions between the user and items, noted as $\{e_1, e_2, \ldots, e_t\}$. Each event e_t has its own environment $C^{e_t} =< a_t, i_t, d_t >$, which denotes that at time step t the user performs the activity a_t on the item i_t with d_t dwell time. The environment surrounding each event is called event-context.

With the notations and definitions, the problem we want to study is formally stated as: given the historical sessions, i.e., sequences of events $s = \{< a_1, i_1, d_1 >, \ldots, < a_t, i_t, d_t >\}$ with session-context C^S, we aim to build a model that can recommend the next item i_{t+1} the user most likely to visit at next time.

In this paper, we address this task with a novel recommendation framework, i.e., Hierarchical Context-aware Recurrent Network (HiCAR). HiCAR contains three main components: (a) *Session-Context Learning* (**SCL**), (b) *Sequential Behavior Learning under Event-Context* (**ECL**) and (c) *Combination and Recommendation* (CR). Specifically, ***SCL*** capture high-order non-linear interactions between each context feature by an n-way hybrid pooling strategy, and ***ECL*** capture the sequential behavior pattern under specific event-context. Note that in this paper Event-context Learning (ECL) refers to modeling both behavior sequence and its context rather than only the latter. Finally, ***CR*** produces recommendation by combining sequential behavior (under event-context) and session-context together. The overall architecture of the HiCAR model is shown in Figure 2.

B. SESSION-CONTEXT LEARNING

Different contextual factors have complex coupling relationships between them [26], [27]. Much work relies on particular features or manually crafting combination features by domain

experts. In order to obtain the high-order non-linear interaction between contextual factors, we apply the n-way hybrid strategy in the Session-context Learning (**SCL**) component, which is shown in Figure 2-(a).

1) SESSION-CONTEXT EMBEDDING LAYER

The first layer of SCL is the embedding layer. Suppose we have a session s with context $C^s =< c_1, c_2, \ldots, c_k >$, where $|C^s| = k$ means there are k kinds of session contexts. Each context factor c_i can be represented as a one-hot vector and the length of the vector is the number of values of that factor, the corresponding index of the one-hot vector has a value of 1, and the others are 0. First, we use an embedding layer to get the dense latent vector of each session context factor.

$$f_c = W^c \cdot c \tag{1}$$

where f_c means the embedding of context factor c. $W^c \in \mathbb{R}^{d\times|c|}$ is the embedding matrix, d is the dimension of each session context embedding. Through the embedding layer, all session context C^s can be mapped into a dense continuous space C^s_{emb}, as follows:

$$C^s_{emb} =< f_{c_1}, f_{c_2}, \ldots, f_{c_k} > \tag{2}$$

2) N-WAY HYBRID

An n-way hybrid layer, which stacks different way of hybrid strategy to explore meaningful feature interactions. For session contexts set C^s, the 1-st way is individual feature hybrid (H_1), while the 2-nd way is pairwise feature hybrid (H_2) by element-wise product, detailed as follows:

$$H_1(C^s) = \begin{bmatrix} f_{c_1} \\ \vdots \\ f_{c_k} \end{bmatrix}^{k\times d} \tag{3}$$

$$H_2(C^s) = \begin{bmatrix} \vdots \\ f_{c_i} \otimes f_{c_j} \\ \vdots \end{bmatrix}^{(k-1)k/2\times d} \tag{4}$$

where $1 \le i \le j \le k$, and $\otimes$ means element-wise product. Furthermore, the n-th way feature hybrid shares the same way:

$$H_n(C^s) = \begin{bmatrix} \vdots \\ f_{c_1} \otimes \ldots \otimes f_{c_n} \\ \vdots \end{bmatrix}^{\binom{k}{n}\times d} \tag{5}$$

3) MAX POOLING AND AGGREGATION

Each way is followed by a max-pooling operation to capture the complex inherent structures of the context factors. The aggregation layer sums up each way of the pooling output to form a final representation as follows:

$$Hybrid^n(C^s) = \sum_{i=1}^{n} p_i^{max}(H_i(C^s)) \tag{6}$$

The hybrid operation aggregates multi-order interaction between multiple features, which maps them to another contiguous latent space. We denote it as $\Psi_{sc} = Hybrid^n(C^s)$, which represents the macro environment the user is currently surrounding by. Du *et al.* [27] empirically proved that the 2-way hybrid leads to good performance with low computational cost. So we employ 2-way hybrid in this study:

$$\begin{aligned} \Psi_{sc} &= Hybrid^2(C^s) \\ &= \sum_{i=1}^{2} p_i^{max}(H_i(C^s)) \\ &= p_1^{max}(H_1(C^s)) + p_2^{max}(H_2(C^s)) \end{aligned} \tag{7}$$

which aggregates the 1-st way hybrid (H_1) and 2-nd way hybrid (H_2) into the final session-context representation Ψ_{sc}.

C. EVENT-CONTEXT LEARNING

SCL helps account for high-order feature interactions between the global contexts, but contributes less to the discovery of sequential behavior patterns. Therefore, ECL is designed to capture current purchase intention from the sequential behaviors in the session, which is an improved Recurrent Neural Network to simultaneously model the behavior sequence and its corresponding event-context.

As shown in Figure 2-(b), ECL is a three-layer network from bottom to top: (a) firstly, an embedding fusion layer embeds each event and its corresponding context into a dense vector; (b) then, a Time-LSTM layer with a time gate is elaborately designed to scan the sequence of events with different time intervals; (c) finally, the attention layer dynamically assigns different weights to different events and combine them into a hybrid representation.

1) EMBEDDING FUSION LAYER

Traditional RNN-based models recurrently scan through a series of elements, where each element is a single notation, e.g., an item; however, accurate user purchase intent cannot be obtained from a pure sequence of items, and the action type on an item can be used as side information to enhance the recommendation effect. For example, browsing the comments of an item and adding it to the favorite list usually means that the user is interested in it, while removing an item from the shopping cart means the opposite. However, traditional RNN based models fail to capture such microscopic differences.

To address this issue, an embedding fusion layer is designed before the RNN to extend the input sequence so that it can scan both the items and their contexts. Specifically, the event sequence is defined as $s = \{< a_1, i_1, d_1 >, \ldots, < a_t, i_t, d_t >\}$, where each sequence element $e =< a, i, d >$ is a triplet means that the user performs activity a_t on item i_t with d_t dwell time. The dwell time denotes how long the user stay on the current item, which is the delta time between two adjacent items:

$$d_j = t_{j+1} - t_j(seconds) \tag{8}$$

For the event-context $e =< a, i, d >$, i and a are one-hot vectors of items and actions with vocabulary sizes $|I|$, $|A|$, respectively. We employ an embedding layer to project them into a low-dimensional dense space, which is formally defined as:

$$f_i = W^I \cdot i \tag{9}$$
$$f_a = W^A \cdot a \tag{10}$$
$$x_t = f_{i_t} \oplus f_{a_t} \tag{11}$$

where $\oplus$ means concatenate operation, f_i means the embedding of item i, $W^I \in \mathbb{R}^{d_I \times |I|}$ is the item embedding matrix, d_I is the dimension of each item embedding. Similarly, f_a means the embedding of action type a, $W^A \in \mathbb{R}^{d_A \times |A|}$ is the embedding matrix, d_A is the dimension of each action type embedding.

In this way, for the item-action embedding pair $< f_{i_t}, f_{a_t} >$ at the $t-$th time step, the concatenating operation is used to fuse them to a combination embedding x_t, and the combination is used as the input of RNN. Next, we detail how to fuse and model the third element of the triplet, i.e. dwell time.

2) TIME-LSTM LAYER

In traditional tasks such as language modeling, RNN can only capture the sequence order without the time intervals between two adjacent elements. However, in the session-based recommendations, time intervals between users' actions (i.e. dwell time) are of significance in capturing the relations of users' actions. For example, if a user is interested in a phone, he may spend more time browsing through its images, parameter details, and other users' comments, while spending less time usually indicates that he is not interested in it, or clicked on it by mistake.

To address this issue, TIME-LSTM is proposed to model the action sequences with different time intervals to enhance the recommended effect of SRS.

Next, we first describe how to use TIME-LSTM to process each event with event-context, and then elaborate on the internal structure of the TIME-LSTM unit in the next subsection. At time step t, we have event $e_t =< f_{i_t}, f_{a_t}, d_t >$, where f_{i_t} and f_{a_t} are the item embedding and action type embedding respectively, and d_t is dwell time. The hidden state h_t of TIME-LSTM is:

$$h_t = \textit{Time-LSTM}(x_t, d_t, h_{t-1}) \tag{12}$$

where the input x_t of TIME-LSTM is the concatenation of item embedding and action type embedding. A time-gate is added in the TIME-LSTM unit to model the dwell time. So at time t, the hidden state h_t of TIME-LSTM is determined by x_t, the dwell time d_t and the previous hidden state h_{t-1} simultaneously.

3) TIME-LSTM UNIT

The proposed TIME-LSTM is shown in figure 3, which equips the standard LSTM [28] a time gate to fine-tune the input gate and forget gate simultaneously. The update equations of the LSTM [28] unit are as follows:

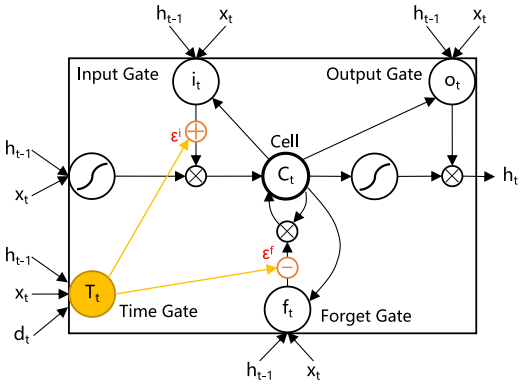

FIGURE 3. The proposed TIME-LSTM Unit. We add a time gate to the LSTM unit and fine-tunes the input gate and forget gate through the time gate.

$$i_t = \sigma(W_i x_t + U_i h_{t-1}) \tag{13}$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1}) \tag{14}$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1}) \tag{15}$$
$$\hat{c}_t = tanh(W_c x_t + U_c h_{t-1}) \tag{16}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t \tag{17}$$
$$h_t = o_t \odot tanh(c_t) \tag{18}$$

where i_t, f_t and o_t are the input gate, forget gate and output gate respectively, and the final hidden state is h_t.

In LSTM, the input gate is a controller to determine how much of the current input information can be updated into the cell state, and the forget gate is a controller to determine how much of the previous information needs to be forgotten [28]. From Eq.(17), it can be seen that the larger the value of the input gate, the more the current input value is updated to the cell state, while the smaller the value of the forget gate, the less the historical state is updated to the cell state. Based on this observation, a time gate $\mathbf{T_t}$ is designed to model the dwell time of each action in the session and fine-tune the input gate and forget gates, detail as:

$$\mathbf{T_t} = \sigma(W_t x_t + U_t h_{t-1} + \mathbf{V_t d_t}) \tag{19}$$

where σ is the sigmoid function, and d_t is dwell time and V_t is its weight matrix. Then the Eq.(17) is rewritten as follows, the changes of the formula are marked with underlines:

$$c_t = \underline{(f_t - \varepsilon_t^f)} \odot c_{t-1} + \underline{(i_t + \varepsilon_t^i)} \odot \hat{c}_t \tag{20}$$

where $\odot$ is element-wise product, and ε_t^f and ε_t^i are the fine-tuning factors of the forget gate and input gate, respectively, defined as follows:

$$\varepsilon_t^f = f_t \odot \mathbf{T_t} \tag{21}$$
$$\varepsilon_t^i = (1 - i_t) \odot \mathbf{T_t} \tag{22}$$

where the forget fine-tuning factor $\varepsilon_t^f \in (0, f_t)$ is used to adjust the forget gate f_t, which can guarantee $\underline{f_t - \varepsilon_t^f} \in (0, 1)$.

In the same way, the input fine-tuning factor $\varepsilon_t^i \in (0, 1 - i_t)$ can make sure that $i_t + \varepsilon_t^i \in (0, 1)$.

In this way, the time gate $T_t \in (0, 1)$ can control the fine-tuning degree of the input gate and the forget gate simultaneously. When it has a positive effect on the input gate, it adversely affects the forget gate accordingly. An intuitive understanding is that when a user is interested in an item and has a longer dwell time on it, the time gate will increase the impact of the current action and correspondingly reduce the impact of the previous actions. We will conduct experiments to verify the effectiveness of TIME-LSTM in the experimental section IV-F.

4) EVENT ATTENTION LAYER

The behaviors in the sequence have varied effects on subsequent behaviors [3], [4]. Therefore, we introduce an event-level attention layer based on the attention mechanism [29] to measure the importance of each event. This yields:

$$\alpha_j = \frac{exp(e_{jt})}{\sum_{i=1}^{t} exp(e_{it})} \tag{23}$$

$$e_{jt} = h_j^T h_t \tag{24}$$

where the weight α determines which part of the input sequence should be emphasized or ignored when making predictions. The function e_{jt} specifically computes the energy [29] between the final hidden state h_t and the representation of the previous event annotations h_j.

The event-context b_t, is then computed as a weighted sum of these event annotations:

$$b_t = \sum_{i=1}^{t} \alpha_i h_i \tag{25}$$

After the attention layer, we get the representation of sequence behavior under particular event-context, denoted as $\Psi_{ec}^B = b_t$.

D. COMBINATION AND RECOMMENDATION

So far, we have modeled two context representations: session-context Ψ_{sc} and event-context Ψ_{ec}^B. Since a user's interest depends on both the event-context and the session-context, the final representation is obtained by combining them together by a fully-connected layer:

$$\Psi_t = W_o[\Psi_{sc} \oplus \Psi_{ec}^B] + b_o \tag{26}$$

where W_o is a linear transformation matrix and b_o is the bias, and Ψ_t is the final representation of the current session s at time step t.

As shown in Figure 2-(c), we employ the dot-product similarity function to compute similarity scores between the final representations Ψ_t and candidate items. Then the similarity scores are normalized by a softmax layer to obtain probability $\hat{y}$ of the items that the user will click next time.

$$y = softmax(Emb^T \Psi_t) \tag{27}$$

where Emb is the embedding matrix of all items with the shape of $|I| \times |D|$, $|I|$ is the total number of items and $|D|$ is the dimension size, which shares parameters with the item embedding matrix W^I in Equation 9.

E. OBJECTIVE FUNCTION AND MODEL OPTIMIZATION

The HiCAR model contains three modules: SCL, ECL, CR, and the parameters of each module are $P_S = \{W_{c1}, W_{c2}, \ldots, W_{ck}\}$, $P_E = \{W_i, U_i, W_f, U_f, W_o, U_o, W_c, U_c, W_t, U_t\}$ and $P_{CR} = \{W_o, b_o\}$, respectively. Floating-point operations (FLOPs) is a common indicator that measures the computation cost, which is usually adopted in deep learning-based models [30]. The FLOPs of the three modules are:

$$FLOPs^{SCL} = \sum_{i=1}^{k}(2 \cdot n_i - 1) \cdot D_s + 2 \cdot k \cdot D_s + k \cdot (k-1) \cdot D_s^2 \tag{28}$$

$$FLOPs^{ECL} = L \cdot (16 \cdot (D_e + D_i) \cdot H + 2 \cdot H^2) \tag{29}$$

$$FLOPs^{CR} = |I| \cdot D_i \cdot (D_s + D_e + 1) \tag{30}$$

The total FLOPs are obtained by summing the three, where the computational cost of SCL depends on the number of session contexts k, the vocab size of each session context n_i, and the embedding size of session context D_s. Meanwhile, the cost of ECL depends on the max sequence length L, the embedding size of event context, and the hidden size H of GRU. For the CR module, its cost mainly depends on the item vocab size $|I|$, the item embedding size D_i, in addition to the embedding size of session and event contexts. When the values of the model hyper-parameters are assigned to them, we obtain the total FLOPs of 230M.

In the HiCAR model, SCL and ECL are independent before they are combined, which leads to the possibility that they can be calculated in parallel. Therefore, we implemented a parallel strategy on the SCL and ECL modules to improve computational efficiency. The actual FLOPs after parallelization is $max(FLOPs^{SCL}, FLOPs^{ECL}) + FLOPs^{CR}$, its value is approximate 128M, which is almost equivalent to the basic GRU4Rec model and will significantly improve model training and prediction time. We will verify it through experiments in Section IV-G.

To learn the parameters of the model $\{P_S, P_E, P_{CR}\}$, we jointly train the model by mini-batch gradient descent on cross-entropy loss:

$$Loss = -\sum_{i=1}^{|I|} y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i) \tag{31}$$

where $\hat{y}$ is the prediction probability distribution and y is the ground truth distribution. At last, a Back-Propagation Through Time (BPTT) method is adopted to train HiCAR.

IV. EXPERIMENTS

In this section, we conduct experiments to validate the effectiveness of HiCAR on sequential behavior based on hierar-

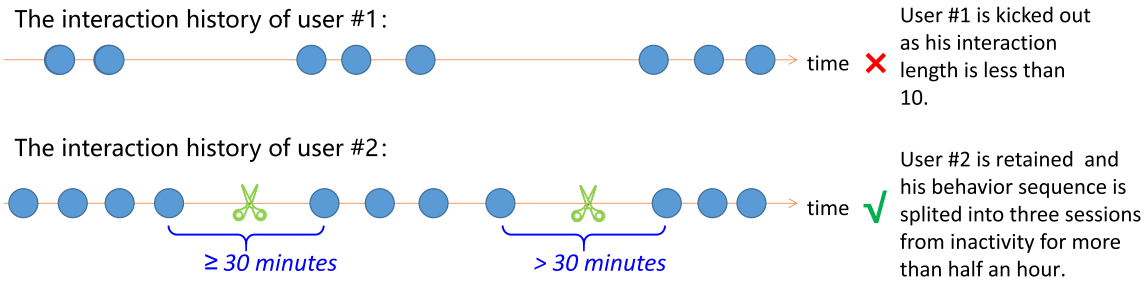

FIGURE 4. The process of obtaining sessions in the Jingdong dataset.

chical contexts. Particularly, we aim to answer the following research questions (RQs):

- *RQ1:* How is the performance of HiCAR compared with existing session-based recommender models?
- *RQ2:* How is the effectiveness of the SCL and ECL component of the HiCAR model in modeling session-context and event-context, respectively?
- *RQ3:* How does the time gate in TIME-LSTM affect the performance of HiCAR in different session lengths?
- *RQ4:* Since session-based recommendations require real-time recommendations for sessions, so how does the HiCAR model perform in terms of efficiency?

A. DATASETS AND SETTINGS

Two real-world datasets, i.e., the Yoochoose and Jingdong dataset, are adopted to evaluate the HiCAR model.

1) YOOCHOOSE

Yoochoose[1] is a public dataset released by RecSys'15 Challenge. It consists of six months of click-streams gathered from an e-commerce website. This dataset contains click events and purchase events. We merge them together and extract session-context and event-context to implement our HiCAR model according to the contextual information in the dataset. First, we extract two types of event-context: the action (click and purchase) and the time interval between every two adjacent events in a session, i.e. dwell time. Second, we introduce the calendar of the local time zone as side information to extract four kinds of session-context features:

- *Ten-days:* three ten-days time periods in a month.
- *Weekday:* seven days in a week.
- *Hour:* twenty-four hours of a day.
- *Holiday:* holiday or not.

We use the sessions of the subsequent day for testing. Because the Yoochoose training set is quite large and training on the recent fractions yields better results than on the entire fractions, we use the recent fractions 1/64 and 1/4 of training sequences following the previous work [2].

[1]http://2015.recsyschallenge.com/challenge.html

2) JINGDONG

JD[2] is provided by a Chinese e-commerce company Jingdong, which is one of the top two largest B2C online retailers in China. Specifically, it provides 27,087,895 interactions of 105,180 customers on 28,710 items within 75 days based on the real log data of users-commodities behaviors. There is no explicit session id in the JD dataset, so we implement our session-based recommendation through the following data preprocessing. We kick out the users if the number of their interacted items is less than ten. For each of the remaining users, we divided his behavior sequence into multiple sessions, and the time interval between two consecutive items in a session is less than half an hour. That is to say, if the time interval between two consecutive items is over half an hour, they will be divided into two different sessions, as shown in Figure 4. Customer behaviors in this dataset include browse, cart, delete-to-cart, purchase, collect, and click with the corresponding values of 1, 2, 3, 4, 5, and 6, respectively. The Session-context feature extracted on the JD dataset is the same as on the Yoochoose dataset. We use the sessions of the last 7 days as the test set. Each session is assigned to either the training or the test set, we do not split the data mid-session. The sessions of the last 30 days in the training set are used as validation sets to adjust model parameters.

For both Yoochoose and JD datasets, following the conventions of session-based recommendation [1], [3], [4], we filter out sessions of length 1 and items that appear less than 5 times in the dataset, and filter out items in the test set that didn't appear in the training set. Same as [2]–[4], we use the data augmentation techniques: for an input session $s = [e_1, e_2, \ldots, e_n]$, we generate the sequences and their corresponding labels $([e_1], i_2)$, $([e_1, e_2], i_3)$, $\ldots$, $([e_1, e_2, \ldots, e_{n-1}], i_n)$ for training and testing on three datasets, the label i_n is the target item in the last event of the current session. The statistics of the datasets after preprocessing are shown in table 1.

3) PARAMETER SETTINGS

We implemented our model based on Pytorch.[3] The hyper-parameters are optimized via an extensive grid search

[2]https://jdata.jd.com/html/detail.html?id=1
[3]http://www.pytorch.org

TABLE 1. Statistics of datasets.

Statistics	Yooc 1/64	Yooc 1/4	JD
train sessions	369,859	5,911,746	3,360,287
test sessions	55,898	55,898	345,911
items	16766	29618	24,744
behaviors	557,248	8,326,407	37,061,992
behavior types	2	2	6
avg.length	6.16	5.71	9.05

on all the data sets, and the best models are selected by early stopping based on the Recall@20 on the validation set. According to the averaged performance, the optimal parameters are set as follows: the item embedding dimension is set to 100. The TIME-LSTM cell size is set to 200. The embedding dimensions of the features in the event-context and session-context are set to 20 and 40 respectively. For training, optimization is done using mini-batch gradient descent with the learning rate sets to 0.001, and the mini-batch size is fixed at 128. The number of epochs is set to 30.

B. BASELINE METHODS

We compare the proposed HiCAR model with the following baselines.

POP: The POP predictor is a naive model that always recommends the most popular items in the training set. Despite its simplicity, it is often a strong baseline in certain domains.

Item-KNN [12], [13]: An item-to-item model which recommends items similar to the existing items based on the cosine similarity between the candidate item and the existing items within the session.

FPMC [16]: A state-of-the-art hybrid model that combines matrix factorization and first-order Markov chains for the next-item recommendation.

GRU4Rec [1]: The first work adopting RNN for session-based recommendations, it utilizes a session-parallel mini-batch training process and also employs ranking-based loss functions during the training.

GRU4Rec+ [2]: An improved model of GRU4Rec which adopts two techniques to improve the performance of GRU4Rec, including a data augmentation method and a method to account for shifts in the input data distribution.

NARM [3]: An RNN-based model that employs the attention mechanism to capture the main purpose from the hidden states and combines it with the sequential behavior as a final representation to generate recommendations.

STAMP [4]: An RNN-based state-of-the-art model that proposed to use attention mechanism to take into account users' current interests from the short-term memory of the last-clicks.

SGNN-HN [24]: It applies a star graph neural network (SGNN) and a highway network (HN) to model the complex transition relationship between items in a session.

HCA [10]: It utilizes a Hierarchical Contextual Attention-based (HCA) network for the sequential recommendation, in which the context is summarized from several adjacent items by attention mechanism.

STAR [8]: Two-layer RNNs are stacked to capture the input context and the temporal context.

C. EVALUATION METRICS

As recommender systems can only recommend a few items at each time, the actual item a user might pickup should be amongst the first few ones of the list. Therefore, we adopt the following two widely used metrics (i.e. Recall and NDCG) for evaluation of the performance of the SRS models.

Recall@K: The primary evaluation metric is the proportion of cases having the desired item amongst the top-K items in all test cases. Defined as:

$$Recall@K = \frac{n_{hit}}{N} \tag{32}$$

where N denotes the number of test data t, n_{hit} denotes the number of cases which have the desired items in top K ranking lists recommended by the system.

NDCG@K: The second metric used in our experiments is *Normalized Discounted Cumulative Gain*, which measures how well a method can rank the true item higher in the recommendation list.

$$DCG = \sum_{i=1}^{K} \frac{rel_i}{log_2^(i+1)} \tag{33}$$

$$NDCG = \frac{DCG}{IDCG} \tag{34}$$

where $rel_i = 1$ if the $i-$th item in the recommendation list is accepted by the user, and $rel_i = 0$ otherwise. IDCG is the maximum possible discounted cumulative gain (DCG) with the top N relevant items. We average the NDCG values of all sessions in the test set as the final result.

D. COMPARISON AGAINST BASELINES (RQ1)

The results on all the benchmark datasets are illustrated in Tables 2, 3 and 4 at top-5, top-10 and top-20 respectively, in which the best result of each column is highlighted in boldface. The following observations can be made from the results:

(1) Traditional methods such as Item-KNN and FPMC are not competitive, they only outperform the naive POP model. This is because they break down the sessions into multiple records (e.g., session-item interaction pairs), thus losing the sequence feature of behaviors in the session. These results provide evidence that traditional user-based methods are not suitable for the session-based recommendation. (2) It can be found that all RNN-based methods (i.e. GRU4Rec, GRU4Rec+, NARM, STAMP) significantly outperform the traditional baselines, which indicates that RNN-based methods are good at modeling sequential data and discovering the potential sequential patterns from the sessions and making more accurate recommendations. (3) Among all RNN-based baselines, NARM and STAMP occupy the top two. This is reasonable because the attention mechanism is

TABLE 2. Top-5 performance comparison of HiCAR with baseline methods over the benchmark datasets.

Methods	Yoochoose 1/64		Yoochoose 1/4		JD	
	Recall@5(%)	NDCG@5(%)	Recall@5(%)	NDCG@5(%)	Recall@5(%)	NDCG@5(%)
POP	5.34	1.52	1.09	0.22	10.09	3.48
Item-KNN	33.24	19.53	35.57	19.73	36.29	19.31
FPMC	31.43	14.07	-	-	-	-
GRU4Rec	39.53	21.24	39.89	20.71	44.32	39.79
GRU4Rec+	42.06	26.82	42.85	28.01	49.19	45.28
NARM	44.34	27.52	44.34	27.38	49.13	49.80
STAMP	44.69	28.08	45.39	28.57	51.82	50.63
SGNN-HN	45.72	28.14	46.57	29.14	52.92	51.31
HCA	45.65	29.24	47.29	29.17	53.06	50.91
STAR	46.28	28.47	47.85	29.44	53.35	52.38
HiCAR	**47.19**	**29.68**	**48.84**	**30.88**	**54.39**	**53.82**

TABLE 3. Top-10 performance comparison of HiCAR with baseline methods over the benchmark datasets.

Methods	Yoochoose 1/64		Yoochoose 1/4		JD	
	Recall@10(%)	NDCG@10(%)	Recall@10(%)	NDCG@10(%)	Recall@10(%)	NDCG@10(%)
POP	5.63	0.29	1.17	1.65	10.91	3.63
Item-KNN	44.38	21.13	44.54	21.27	48.27	21.91
FPMC	37.64	15.12	-	-	-	-
GRU4Rec	51.54	22.32	48.82	22.55	59.87	41.18
GRU4Rec+	55.62	28.51	54.36	27.61	64.58	48.33
NARM	56.50	27.13	56.83	26.83	65.39	48.73
STAMP	57.07	29.21	57.62	28.52	69.21	49.61
SGNN-HN	58.69	29.66	57.78	28.92	71.38	53.15
HCA	58.29	29.77	59.23	29.93	72.85	56.76
STAR	58.71	29.83	60.57	32.47	73.72	55.46
HiCAR	**60.89**	**32.76**	**62.27**	**33.56**	**75.92**	**58.06**

TABLE 4. Top-20 performance comparison of HiCAR with baseline methods over the benchmark datasets.

Methods	Yoochoose 1/64		Yoochoose 1/4		JD	
	Recall@20(%)	NDCG@20(%)	Recall@20(%)	NDCG@20(%)	Recall@20(%)	NDCG@20(%)
POP	6.71	1.68	1.33	0.34	12.46	3.43
Item-KNN	51.60	22.46	52.31	19.31	58.54	19.80
FPMC	45.62	15.61	-	-	-	-
GRU4Rec	60.64	23.81	59.53	24.41	70.34	38.62
GRU4Rec+	67.84	30.74	69.11	30.97	79.34	45.37
NARM	68.32	29.62	69.73	28.06	80.35	52.84
STAMP	68.74	30.56	70.44	31.80	81.26	54.51
SGNN-HN	70.06	31.25	70.85	31.93	82.71	55.43
HCA	70.45	33.67	72.63	32.46	83.02	55.97
STAR	72.32	34.03	72.43	32.58	83.12	55.00
HiCAR	**73.26**	**35.34**	**74.66**	**36.79**	**84.24**	**59.61**

adopted to give different weights to different behaviors in the sequence, thus allowing them to obtain important behaviors from the session and improve the prediction effectiveness. Despite this, our model still outperforms STAMP on RECALL@20 and NDCG@20 by about 4% in the three datasets. This highlights the significance of exploiting the sequential contextual information and demonstrates the effectiveness of ECL in our model. (4) The GCN-based approach, which replaces the sequence of items by complex transition relationships between items, is slightly better than the RNN-based approaches. For example, the improvement of SGNN-HN over the best RNN-based baseline (i.e. STAMP) in terms of Recall@20 and NDCG@20 on Yoochoose 1/64 are 1.32% and 0.69%, respectively, and 0.41% and 0.13% on Yoochose 1/4. This shows that although the GCN model that models the global relationship between items is better than the model employing only item sequences. However, it does not model the session-context and event-context information, therefore, the improvement is not as obvious as our proposed HiCAR model. (5) Context-aware enhanced models, such as HCA, STAR, and HiCAR, have a significant performance improvement over baseline methods, which indicates that contextual information can influence users' preference and lead to different choices, and effective modeling of the contexts can capture the micro difference and improve recommendation performance. (6) By taking both the event-context and session-context into consideration, the HiCAR model can outperform all the baselines in terms

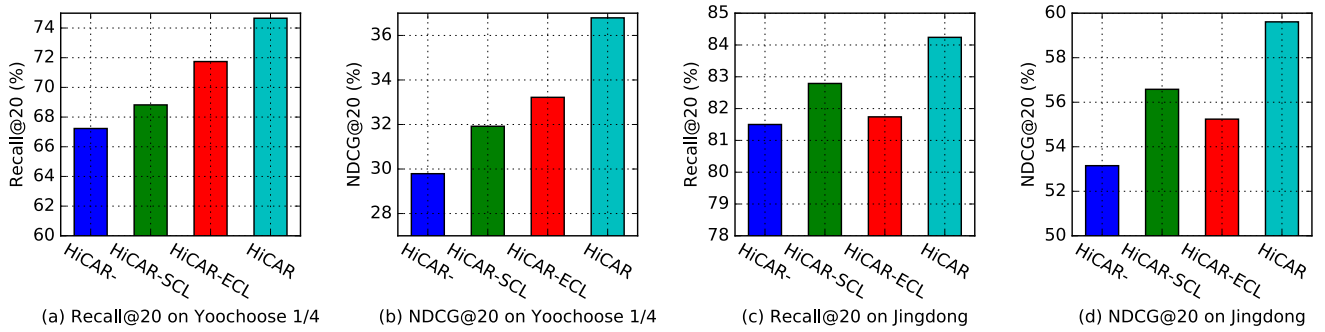

FIGURE 5. The comparison of the results between four variants of our HiCAR model.

of Recall@20 and NDCG@20 over three datasets. Take Yoochoose 1/4 for example, when compared with the best baseline (i.e., STAR), the performance improvements by HiCAR are around 2.23% and 4.21% in terms of Recall@20 and NDCG@20, respectively.

Based on the above analysis, it can be concluded that the proposed HiCAR model outperforms the baselines and can effectively capture session-context and event-context when performing SRS. We will carry out an ablation experiment in the next subsection to verify the effectiveness of each module in our HiCAR model.

E. ABLATION STUDY (RQ2)

We believe that both session-context and event-context will affect the next behavior in the session. In this section, we design three variants of the proposed HiCAR model to quantitatively compare the effect of each of them, separately.

- *HiCAR-:* it removes both session-context and event-context, leaving only the item sequence and the dwell time, and the attention mechanism is reserved in this variant. This variation serves as a baseline in all variants.
- *HiCAR-SCL:* This variation removes the entire SCL component, leaving only the ECL component to get the sequence behavior with the event context, in which the time gate is also retained.
- *HiCAR-ECL:* which removed the ECL component from the HiCAR model. In this variant, ECL degenerates into a standard LSTM that only models item sequences.
- *HiCAR:* The complete HiCAR model including both SCL and ECL to capture the session-context and event-context simultaneously.

The results are shown in Figure 5. The indicators on Yoochoose 1/4 and 1/64 have similar conclusions, so we only show the results on Yoochoose 1/64 and Jingdong. Through the experimental results we have the following observations:

First, We notice that the performances of HiCAR-SCL and HiCAR-ECL are both improved toward HiCAR- model. For example, on the Yoochoose 1/4 data, the Recall@20 of HiCAR-SCL and HiCAR-ECL increased by 1.6% and 4.5% compared to HiCAR-, respectively. This demonstrates the significant contribution of the two main components, i.e., SCL and ECL. Second, it is interesting that HiCAR-ECL outperforms HiCAR-SCL on the Yoochoose dataset, while HiCAR-SCL outperforms HiCAR-ECL significantly on the Jingdong dataset, which indicates that the contextual information in event-contexts and session-contexts has different effects on predicting future behaviors. That is to say, on the JD dataset, ECL has a greater impact than SCL. After analyzing the event context of the two, we found that on the Yoochoose dataset, there are only two types of events (click and purchase), while on the Jingdong dataset, there are six types of events (browse, cart, delete-to-cart, purchase, collect and click). One possible reason is that ECL can capture more accurate purchase intention from the subdivided behavior types on the Jingdong dataset, thereby enhancing the performance of recommendation. Finally, by incorporating both event-contexts and session-contexts, HiCAR outperforms all the variants significantly. This result further demonstrates the effectiveness of our HiCAR for modeling session-context and event-context simultaneously.

F. THE EFFECT OF TIME-GATE (RQ3)

To assess the effectiveness of TIME-LSTM, we design a group of comparative experiments from two aspects: (1) Firstly, we posit that a user's purchase intention of a session is hidden in his behaviors, which may be reflected by the dwell time on different items. So, we removed the time gate from the HiCAR model. In order to better distinguish them, the variant without the time gate is named as HiCAR$^{\ominus}$ and the complete model as HiCAR. (2) Furthermore, the longer the user's behavior sequence in a session, the more information can be used to discover the user's purchase intention. So we partition all sessions into three groups according to the sequence lengths: less than 5, between 5 and 10, and more than 10. We separately test the performance of the two variants on the three kinds of groups, the results are shown in Figure 6.

Through the experimental results, we have the following observations: First, we pay attention to the overall comparison between HiCAR and HiCAR$^{\ominus}$, the two indicators of HiCAR on both datasets are better than HiCAR$^{\ominus}$ without a time gate, which confirms that the TIME-LSTM can

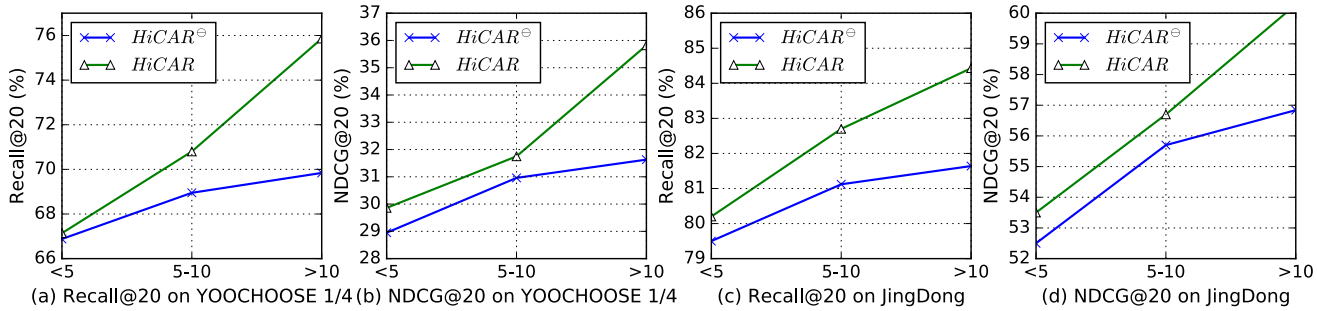

FIGURE 6. The effectiveness of time-gate under different sequence lengths. HiCAR$^{\ominus}$ denote the variate of HiCAR without the time gate.

effectively capture the difference of dwell time and improve the recommendation performance. Second, it is interesting that the performance improvement increases as the length of the interaction grow. Take the Recall@20 on Yoochoose 1/4 dataset in Figure 6-(a) for example, when the length of the session is less than 5, the recall@20 of HiCAR is 67.14%, which is only 0.26% higher than that of HiCAR$^{\ominus}$ (i.e. 66.88%). When the session length is between 5 and 10, the gap increases to 1.85% (68.95% vs 70.80%). When the session length is greater than 10, the gap even increases to 6.02% (69.84% vs 75.86%). This clearly shows that the longer the user's behavior sequence, the more his preference information can be reflected, which can be better captured by HiCAR. This further verified the validity of the TIME-LSTM in capturing the micro event-context information in the session and improves the performance of session-based recommendations.

G. RECOMMENDATION EFFICIENCY (RQ4)

We also compare the efficiency of HiCAR and other state-of-the-art deep learning based models. To be fair, all the methods run on the same Linux server with 64G RAM and 11G GPU memory. Table 5 reports the training time of each epoch and the prediction time of each batch with the size of 128.

TABLE 5. Running time on the Jingdong dataset.

Methods	Training Time (s)	Prediction Time (s)
GRU4Rec	91.0	0.142
GRU4Rec+	103.0	0.176
NARM	187.6	0.764
STAMP	154.0	0.582
HCA	228.4	1.171
STAR	271.0	1.237
HiCAR	126.5	0.153

We observe that GRU4Rec and GRU4Rec+ outperform others. This is because they do not consider the context information and modeling the context features is very costly. We also find that NARM and STAMP are inferior to GRU4Rec. We argue that this is because NARM uses complex global and local GRU units, and STAMP adds MLP cell A and MLP cell B after GRU, which brings more computational cost. When compared with context-aware based models (i.e. HCA and STAR), HiCAR saves about 52 percent training time. This is because the context information in HCA and STAR is first learned and then feed into the RNN, which is actually serial, while the ECL and SCL modules in HiCAR are implemented in parallel and then combined at the CR layer. All the above results imply that HiCAR may be more suitable for practical application since computational efficiency is crucial in real-world session-based recommender systems, which always comprise large amounts of sessions and items.

V. CONCLUSION AND FUTURE WORK

In this paper, we present a hierarchical context-aware recurrent neural network, HiCAR, to tackle the session-based recommendation task. HiCAR contains two main components to capture the impact of the hierarchical context surrounding user behavior. First, an SCL with the n-way hybrid strategy is adopted to model multi-feature interactions in the session-context. Then, an ECL component consists of TIME-LSTM with an elaborately designed time gate is employed to model the event-context within the behavior sequence. Finally, HiCAR combines the two levels of context for the next item recommendation. The experimental results on two real-world datasets demonstrated the effectiveness of the HiCAR model in session-based recommendations.

As to future work, more session-context and event-context, such as location and prices, may be integrated into the HiCAR model to enhance the performance of the session-based recommendation furthermore. Meanwhile, we will migrate our model to some other sequence recommendation tasks such as next-basket recommendation, in which the TIME-LSTM with time gate will be reconstructed to capture the time interval among the basket sequences.

REFERENCES

[1] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–23.

[2] 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.

[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.*, Nov. 2017, pp. 1419–1428.

[4] 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.
[5] Z. Li, H. Zhao, Q. Liu, Z. Huang, T. Mei, and E. Chen, "Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 1734–1743.
[6] J. Manotumruksa, C. Macdonald, and I. Ounis, "A contextual attention recurrent architecture for context-aware venue recommendation," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jun. 2018, pp. 555–564.
[7] G. D. S. P. Moreira, D. Jannach, and A. M. D. Cunha, "Contextual hybrid session-based news recommendation with recurrent neural networks," *IEEE Access*, vol. 7, pp. 169185–169203, 2019.
[8] L. Rakkappan and V. Rajan, "Context-aware sequential recommendations withStacked recurrent neural networks," in *Proc. World Wide Web Conf.*, May 2019, pp. 3172–3178.
[9] J. Wu, Z. Ou, and M. Song, "Session-based recommendation with context-aware attention network," in *Proc. 7th Int. Conf. Inf. Technol., IoT Smart City*, Dec. 2019, pp. 141–146.
[10] Q. Cui, S. Wu, Y. Huang, and L. Wang, "A hierarchical contextual attention-based network for sequential recommendation," *Neurocomputing*, vol. 358, pp. 141–149, Sep. 2019.
[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. Sarwar, G. Karypis, G. Karypis, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web*, vol. 1, May 2001, pp. 285–295.
[13] 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. 2003.
[14] A. Zimdars, D. M. Chickering, and C. Meek, "Using temporal data for making recommendations," in *Proc. 17th Conf. Uncertainty Artif. Intell.*, San Mateo, CA, USA: Morgan Kaufmann, 2001, pp. 580–588.
[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] 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.
[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.
[18] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[19] S. Wang, L. Hu, L. Cao, X. Huang, D. Lian, and W. Liu, "Attention-based transactional context embedding for next-item recommendation," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2018, pp. 2532–2539.
[20] 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.
[21] 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, 2019, pp. 346–353.
[22] 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.
[23] R. Qiu, J. Li, Z. Huang, and H. YIn, "Rethinking the item order in session-based recommendation with graph neural networks," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 579–588.
[24] Z. Pan, F. Cai, W. Chen, H. Chen, and M. de Rijke, "Star graph neural networks for session-based recommendation," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2020, pp. 1195–1204.
[25] Z. Wang, W. Wei, G. Cong, X.-L. Li, X.-L. Mao, and M. Qiu, "Global context enhanced graph neural networks for session-based recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2020, pp. 169–178.
[26] L. Cao, "Coupling learning of complex interactions," *Inf. Process. Manage.*, vol. 51, no. 2, pp. 167–186, Mar. 2015.
[27] Y. Du, H. Liu, Z. Wu, and X. Zhang, "Hierarchical hybrid feature model for Top-N context-aware recommendation," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2018, pp. 109–116.
[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[29] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–15.
[30] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: http://arxiv.org/abs/1704.04861

YOUFANG LENG received the master's degree in computer science and technology from the Renmin University of China, where he is currently pursuing the Ph.D. degree with the School of Information. His research interests include recommendation systems, deep learning, and natural language processing.

LI YU received the Ph.D. degree in systems engineering from the Beijing University of Aeronautics and Astronautics (BUAA), in 2004. He is currently an Associate Professor and the Vice Chair of the Department of Economic Information Management, Renmin University of China. He has published more than 40 papers in important academic journals and conferences. His research interests include big data analysis and application, recommendation systems, deep learning, Internet plus and innovation, data mining and business intelligence, social network marketing, financial data mining and analysis, e-commerce, Web2.0 and social computing, knowledge management, IT project management, and so on. He won awards for outstanding scientific research achievements and teaching achievements of the Renmin University of China.