User Behavior Retrieval for Click-Through Rate Prediction

Jiarui Qin, Weinan Zhang, Xin Wu, Jiarui Jin, Yuchen Fang, Yong Yu
Shanghai Jiao Tong University
{qinjr, wnzhang, wuxin, jerry, arthur_fyc, yyu}@apex.sjtu.edu.cn

ABSTRACT

Click-through rate (CTR) prediction plays a key role in modern online personalization services. In practice, it is necessary to capture user's drifting interests by modeling sequential user behaviors to build an accurate CTR prediction model. However, as the users accumulate more and more behavioral data on the platforms, it becomes non-trivial for the sequential models to make use of the whole behavior history of each user. First, directly feeding the long behavior sequence will make online inference time and system load infeasible. Second, there is much noise in such long histories to fail the sequential model learning. The current industrial solutions mainly truncate the sequences and just feed recent behaviors to the prediction model, which leads to a problem that sequential patterns such as periodicity or long-term dependency are not embedded in the recent several behaviors but in far back history. To tackle these issues, in this paper we consider it from the data perspective instead of just designing more sophisticated yet complicated models and propose **U**ser **B**ehavior **R**etrieval for **CTR** prediction (UBR4CTR) framework. In UBR4CTR, the most relevant and appropriate user behaviors will be firstly retrieved from the entire user history sequence using a *learnable* search method. These retrieved behaviors are then fed into a deep model to make the final prediction instead of simply using the most recent ones. It is highly feasible to deploy UBR4CTR into industrial model pipeline with low cost. Experiments on three real-world large-scale datasets demonstrate the superiority and efficacy of our proposed framework and models.

CCS CONCEPTS

• **Information systems** → **Information retrieval**; **Data mining**.

KEYWORDS

CTR Prediction; Information Retrieval; Sequential User Behavior Modeling

ACM Reference Format:
Jiarui Qin, Weinan Zhang, Xin Wu, Jiarui Jin, Yuchen Fang, Yong Yu. 2020. User Behavior Retrieval for Click-Through Rate Prediction. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401440

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401440

1 INTRODUCTION

Click-through rate (CTR) prediction plays a key role in today's online personalization platforms (e.g., e-commerce, online advertising, recommender systems), whose goal is to predict a user's clicking probability on a specific item under a particular context. With more than a decade of development for the online personalization platforms, the amount of user behaviors logged on the platforms grows rapidly. There are 23% of users have more than 1000 behaviors during six months on Taobao [23]. As there exist resourceful temporal patterns embedded in user behaviors, it becomes an essential problem for both industry and academia to build an effective and efficient model which could utilize user sequential behaviors to obtain accurate predictions of CTR.

In the deep learning era, there are many deep neural network (DNN) based CTR prediction models such as Wide&Deep [4], FNN [43], DeepCross [36], DeepFM [7], PNN [21, 22] and xDeepFM [17], most of which have been deployed on commercial personalization platforms. These models emphasize mining feature interactions, and are proposed to utilize the multi-categorical features of the data better. Nonetheless, such models ignore the sequential or temporal patterns of user behaviors.

As shown in [1, 8, 11, 16], the temporal dynamics of user behavior play a key role in predicting the user's future interests. These sequential patterns include concept drifting [37], long-term behavior dependency [16, 23], periodic patterns [26], etc. Thus, there are models proposed to capture the user's sequential patterns in both CTR prediction and sequential recommendation tasks. For CTR prediction, there are attention-based models like DIN [45] and DIEN [44], memory network-based models like HPMN [23]. More user behavior modeling methods are proposed for sequential recommendation, which is a quite similar task to CTR prediction. There are RNN-based models [12], CNN-based models [31], Transformer-based models [15] and memory network-based models [5, 35].

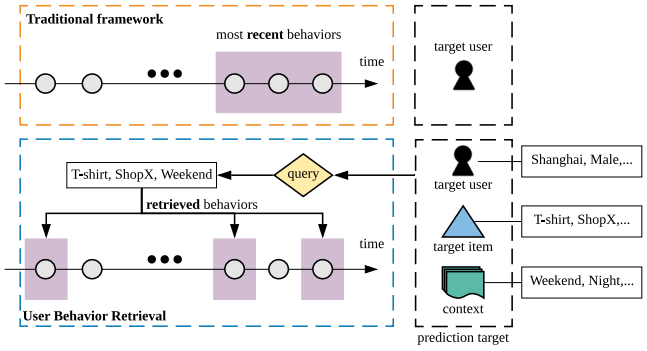

Figure 1: Comparison between traditional framework and UBR4CTR.

However, most of the sequential models above have a common problem in real-world practice. When the platforms have logged a large number of user behaviors, the common industrial solution is to truncate the whole behavior sequence and only uses the most recent N behaviors as the input to the prediction model [44, 45], as illustrated in the upper part of Figure 1. The strict requirements of online serving time plus bottleneck of system load and computational capacity put limits on the length of user sequence that could be used. As a result, in most of industrial cases, no more than 50 recent behaviors are used [44].

Traditional framework that uses the most recent N behaviors could cause negative issues. It is obvious that the effective sequential patterns may be not just embedded in the recent sequence of behaviors. It may be traced back into further history such as periodicity and long-term dependencies [23]. If we try to use a longer sequence, however, a lot of irrelevant behaviors and noises could be introduced. Let alone the time and space complexity the longer history brings.

In this paper, to tackle the above practical issues, we try to solve the problem from the data perspective instead of designing more sophisticated yet complicated model. Specifically, we target to design a framework to retrieve a limited number of historic behaviors that are most useful for each CTR *prediction target*. As shown in Figure 1, a prediction target consists of three parts, i.e.,the target user, the target item and the corresponding context. There are features of the prediction target, such as the user's location, gender, occupation, and item's category, brand, merchant, and context features such as time and scenario. Then we use a model to select a subset of these features, which builds a query to retrieve the relevant historical behaviors. All the user's behaviors are stored as the information items in a search engine, and we use the generated query to search from the historical records. The retrieved behaviors are used in CTR prediction.

For every different candidate target item for the same user, we will retrieve different behaviors for prediction because the generated queries are different. This is a significant change compared with traditional framework that uses exactly the same recent N behaviors for prediction on different items with the same user.

The resulted solution framework is called **U**ser **B**ehavior **R**etrieval for **CTR** (UBR4CTR). In UBR4CTR, the task is divided into two modules. The first is a *learnable* retrieval module which consists of a self-attentive network to select the features and form the query, and a search engine in which the user behaviors are stored in an inverted index manner. The other module is the prediction module in which an attention-based deep neural network is built to make the final prediction based on the retrieved user behaviors as well as the features of the prediction target.

The contributions of the paper can be summarized in three-fold:

- We reveal an important fact that it is important to retrieve more relevant user behaviors than just use the most recent behaviors in user response prediction. Instead of designing more sophisticated and complex models, we put more attention on retrieving user's behavioral data.
- We propose a new framework called UBR4CTR which manages to retrieve different behaviors for the same user when predicting her CTR to different items under different contexts. All the previous sequential models only use the most recent behaviors of a user. We propose a search engine based method and an effective training algorithm to learn to retrieve appropriate behavioral data.
- We conduct extensive experiments and compare our framework with several strong baselines using traditional framework over three real-world large-scale e-commerce datasets. The results verify the efficacy of UBR4CTR framework.

The rest of the paper is organized as follows. In Section 2, we will introduce the preliminaries and some notations used in this paper. Section 3 is about the detailed description of our proposed framework and models. The experimental settings and corresponding results are shown in Section 4. The deployment feasibility is discussed in Section 5. In Section 6 we discuss about some important related works. Finally, we conclude the paper and discuss the future work in Section 7.

2 PRELIMINARIES

In this section, we formulate the problem and introduce the notations. For CTR prediction task, there are M users in $\mathcal{U} = \{u_1, ..., u_M\}$ and N items in $\mathcal{V} = \{v_1, ..., v_N\}$. The user-item interactions are denoted as $\mathcal{Y} = \{y_{uv} | u \in \mathcal{U}, v \in \mathcal{V}\}$ and,

$$y_{uv} = \begin{cases} 1, & u \text{ has clicked } v; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Furthermore, each user-item interaction has a timestamp and context in which the interaction happened, thus the data is formulated as quadruples which is $\{u, v, c, ts\}$ indicating u clicked v at time ts in context c.

To model user's evolving interests, we organize the user behavioral history as $H_u = \{b_1^u, b_2^u, ..., b_T^u\}$ in which b_i^u stands for i-th behavior record of user u sorting by timestamp.

As click-through rate is essentially a matching probability between user, item and context, each behavior record b_i^u is consist of these three parts that $b_i^u = [u, v_i, c_i]$ where v_i is the i-th clicked item and c_i is the context at which the interaction happened.

At the feature level, each user u is represented by a bunch of features that $u = [f_1^u, ..., f_{K_u}^u]$ where f_p^u denotes the p-th feature of u. It is a common practice that all the features are multiple categorical features [24, 45]. Numerical features, if any, are discretized to categorical features. Similarly, $v = [f_1^v, ..., f_{K_v}^v]$ and $c = [f_1^c, ..., f_{K_c}^c]$. The goal of CTR prediction is to predict probability of target user u clicking target item v given historical behaviors of u under context c. It is formulated as

$$\hat{y}_{uv} = f(u, v, c | H_u; \boldsymbol{\theta}), \tag{2}$$

where f is the learned function with parameters $\boldsymbol{\theta}$. We conclude the notations and the corresponding descriptions in Table 1.

3 METHODOLOGY

In this section, we describe our proposed UBR4CTR (**U**ser **B**ehavior **R**etrieval for **CTR** prediction) framework in detail. We firstly give a big picture of the overall framework, then we give detailed descriptions on the user behavior retrieval module and the prediction module. Moreover, the training methods and some analysis about time complexity are given in the following sections.

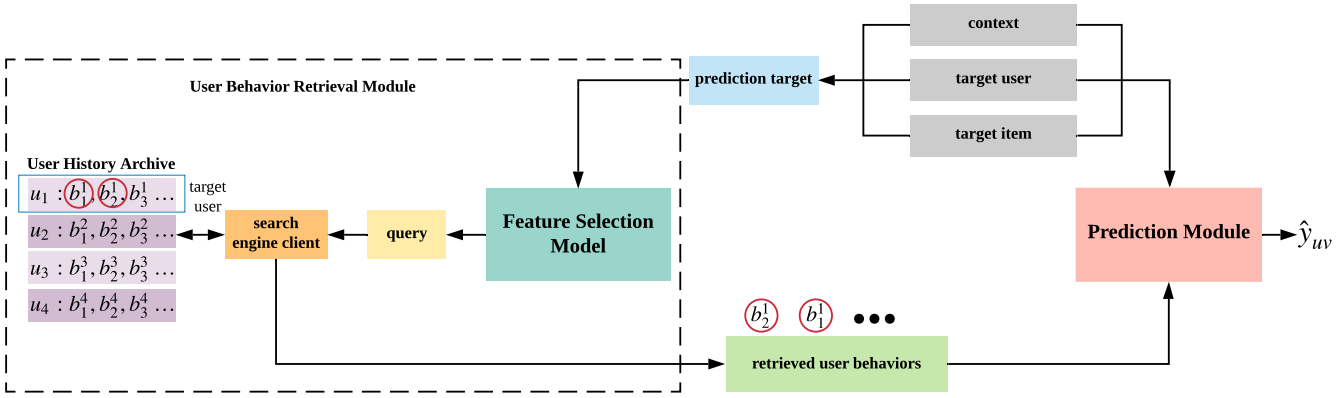

Figure 2: Overall framework of the proposed UBR4CTR framework.

Table 1: Notations and corresponding descriptions

Notation	Description.
u, v	The target user and the target item.
M, N	The number of users and items.
$y, \hat{y}$	The label and the predicted probability of the user-item interaction.
H_u	The behavior sequence of u.
f_p^u, f_q^v, f_r^c	One feature of user, item and context.
$\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{c}$	Embedding representation of target user u, target item v and the corresponding context c.
$\boldsymbol{f}_p^u, \boldsymbol{f}_q^v, \boldsymbol{f}_r^c$	Embedding representation of a feature.
$\boldsymbol{b}_i^u$	Embedding of i-th behavior of u.
B^u	The set of u's retrieved behaviors.
S	Number of retrieved behaviors.

3.1 Overall Framework

The overall framework of UBR4CTR is illustrated in Figure 2. The framework can be divided into two major modules: user behavior retrieval module and prediction module.

The user behavior retrieval module consists of a feature selection model, a search engine client and a user history archive. All the user historical behaviors are stored in the archive and they are organized in a feature-based inverted index manner which will be explained in details in Section 3.2.2. As shown in Figure 2, when we need to predict the click-through rate between the target user and target item in a certain context, all of the three parts of information are combined to form a prediction target. The prediction target essentially consists of a bunch of features of the target user, target item and the context. So the prediction target is then fed to the feature selection model which will select the appropriate features to form a query. Detailed design of the feature selection model is in Section 3.2.1. Then we use the query to search in the user history archive through a search engine client.

The search engine client retrieved a certain number of user behavior records and these records are then used by the prediction module. In the prediction module, we use an attention-based deep model to distinguish the influence of each behavior to the clicking probability and make the final prediction which will be discussed in Section 3.3.

The feature selection model and the prediction model are trained in turn. The goal of feature selection model is to select the most *useful* subset of features. The features of this subset will be combined to generate a query which is used to retrieve the most relevant user behaviors for the final prediction.

3.2 User Behavior Retrieval Module

In this section, we introduce the user behavior retrieval module which consists of a feature selection model and a behavior searching process.

3.2.1 Feature Selection Model. As shown in Figure 3, we regard all the features of target user u, target item v and the corresponding context c as the input of the feature selection model. Without loss of generality, we set f_1^u as the user id feature. User id is a special feature which we must select because we want to retrieve behaviors of user u herself. All the other features are concatenated as a whole. For simplicity, we denote all the features $[f_2^u, ..., f_{K_u}^u, f_1^v, ..., f_{K_v}^v, f_1^c, ..., f_{K_c}^c]$ as $[f_1, ..., f_{K_q}]$ correspondingly where $K_q = K_u + K_v + K_c - 1$.

To better model the relationships and interactive patterns between the features, we use self-attention mechanism [32]. Specifically, we define that

$$Q = K = V = \begin{pmatrix} \boldsymbol{f}_1 \\ \vdots \\ \boldsymbol{f}_{K_q} \end{pmatrix} \tag{3}$$

where $\boldsymbol{f}_i$ is the dense embedding representation of the i-th feature. And $K = Q = V \in R^{K_q \times d}$ where d is the dimension of the embeddings. The self-attention is defined as,

$$\mathrm{SA}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{4}$$

and multihead self-attention is

$$E = \mathrm{Multihead}(Q, K, V) = \mathrm{Concat}(head_1, ..., head_h)W^O, \tag{5}$$

where $head_i = \mathrm{SA}(QW_i^Q, KW_i^K, VW_i^V)$. The parameters matrixes are $W^O \in R^{hd \times d}$, $W_i^Q \in R^{d \times d}$, $W_i^K \in R^{d \times d}$, $W_i^V \in R^{d \times d}$.

The output of multihead self-attention E is then fed to a multi-layer perceptron with $ReLU(x) = max(0, x)$ activation function as

$$\hat{E} = \mathrm{MLP}(E), \tag{6}$$

where $\hat{E} \in R^{K_q \times 1}$. And the probability of selecting each corresponding feature is obtained by taking a sigmoid function as

$$P = \begin{pmatrix} p_1 \\ \vdots \\ p_{K_q} \end{pmatrix} = \sigma(\hat{E}), \tag{7}$$

where $\sigma(x) = \frac{1}{1+\exp^{-x}}$.

Then we sample the features according to these probabilities and thus get the selected subset of features. Note that the user id feature is always in the subset because we have to retrieve from the target user's own behaviors.

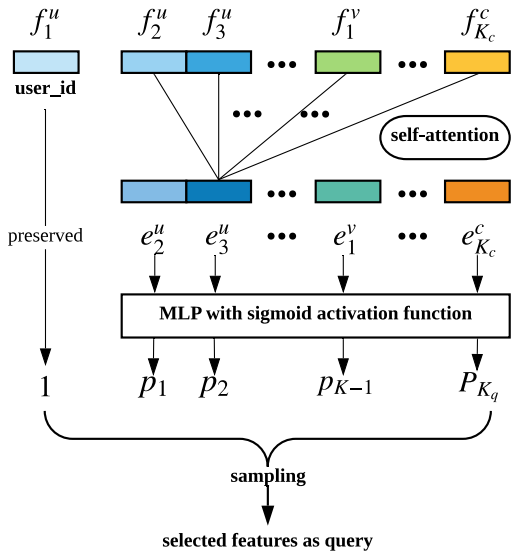

Figure 3: Illustration of feature selection model.

3.2.2 Behavior Searching. We use the typical search engine approach to store and retrieve the user behaviors. We regard each user behavior as a document and each feature as a term. Inverted index is used to track all the behaviors, which is shown in Figure 4. Each feature value is associated with a posting list which consists of all the behaviors that has this specific feature value. For instance, the "user_id_1" has the posting list of all the behaviors of the user whose id is 1 and "Nike" has the posting list which consists of the behavior records that brand ids are "Nike".

The logic of the query is essentially formulated as

$$f_1^u \ AND \ (f_1 \ OR \ f_2 \ OR ... OR \ f_n), \tag{8}$$

where f_1^u is user id. The selected query feature set is $q = \{f_1, f_2, ...f_n\}$. The posting list of f_1^u and the union set of f_1 to f_n's posting lists are intersected. The intersection is the candidate behavior set. Then we use BM25 [29] to score every behavior document in the candidate set and the top S are retrieved.

The similarity s between the query q and a behavior document D is calculated as,

$$s = \sum_{i=1}^{n} \text{IDF}(f_i) \cdot \frac{tf(f_i, D) \cdot (k_1 + 1)}{tf(f_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}, \tag{9}$$

where $tf(f_i, D)$ is feature f_i's term frequency in D which is 1 or 0. The length of a behavior document is defined as the number of the features in that behavior, so all the behavior documents have the same length. Thus the average length is each documents' length and $\frac{|D|}{\text{avgdl}} = 1$. k_1 and b are free parameters. We set $k_1 = 1.2$ and $b = 0.75$. IDF is defined as,

$$\text{IDF}(f_i) = \log \frac{\mathcal{N} - \mathcal{N}(f_i) + 0.5}{\mathcal{N}(f_i) + 0.5}, \tag{10}$$

where $\mathcal{N}$ is the total number of the behavior documents and $\mathcal{N}(f_i)$ is the number of documents that contains the feature f_i. The IDF term gives the common features less importance than rare features. It makes sense that rare features imply stronger signals on user's preference compared to the commonly seen features.

By ranking the candidate behaviors using BM25, we obtain the top S documents as the retrieved behaviors.

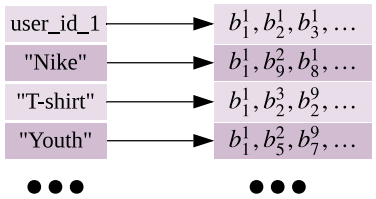

Figure 4: Illustration of feature based inverted index.

3.3 Prediction Module

For the prediction module, we use an attention-based deep neural network to model the importance of different user behaviors to the final prediction.

As shown in Figure 5, the comprehensive user representation $\boldsymbol{r^u}$ is calculated by weighted sum pooling as

$$\boldsymbol{r}^u = \sum_{i=1}^{S} \alpha_i \cdot \boldsymbol{b}_i^u, \boldsymbol{b}_i^u \in B^u \tag{11}$$

where $\boldsymbol{b}_i^u = [\boldsymbol{u}, \boldsymbol{v}_i, \boldsymbol{c}_i]$ and α_i represents the contribution of b_i^u to the comprehensive user representation. The attention weight α_i is calculated as

$$\alpha_i = \frac{\exp(w_i)}{\sum_{j=1}^{S} \exp(w_j)}, \tag{12}$$

where w_i is defined as,

$$w_i = \text{Att}(\boldsymbol{b}_i^u, \boldsymbol{t}), \tag{13}$$

and prediction target $\boldsymbol{t} = [\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{c}]$. Att is a multi-layer deep network with ReLU activation function.

Then the final prediction is calculated as

$$\hat{y}_{uv} = f_\phi\left(B^u, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{c}\right) = f_\phi\left(\boldsymbol{r^u}, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{c}\right) \tag{14}$$

where f is implemented as a three-layer perceptron, whose widths are 200, 80 and 1 respectively. ϕ is the parameters. The output layer uses sigmoid function and other layers use ReLU as activation.

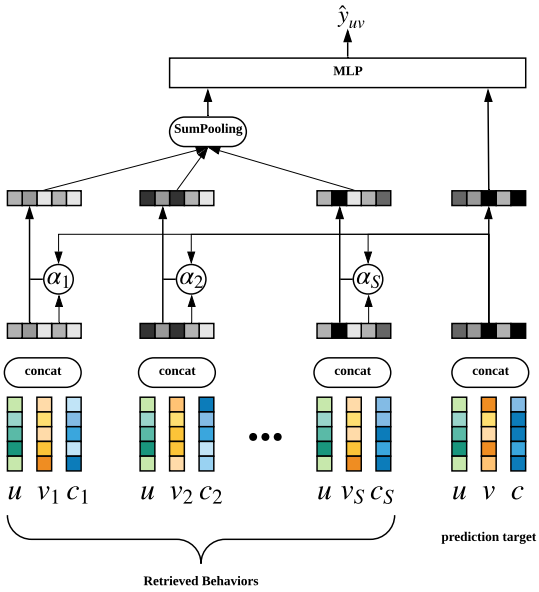

Figure 5: Illustration of attention-based prediction network of the prediction module.

3.4 Model Training

As the objective is to estimate the click-through rate accurately, we utilize the log-likelihood as our objective function which is defined as,

$$J^{\pi_\theta, f_\phi} = \max_\theta \max_\phi \sum_u \sum_v E_{B^u \sim \pi_\theta(B^u|q)}[LL(y_{uv}, f_\phi(B^u, u, v, c))], \tag{15}$$

where LL is the log-likelihood of predicted score on target user-item pair u, i given retrieved user behaviors B^u under context c. It is defined as,

$$\begin{aligned} LL(y_{uv}, f_\phi(B^u, u, v, c)) = y_{uv} \cdot \log(f_\phi(B^u, u, v, c)) \\ + (1 - y_{uv}) \cdot \log(1 - f_\phi(B^u, u, v, c)). \end{aligned} \tag{16}$$

In Eq. 15, q is the query which consists of the selected features as described in Section 3.2.1. Sampling is used to select features and then they form the query string. After the query string is formed, searching results, a.k.a, retrieved behaviors are deterministic. Thus we could regard the behaviors are sampled from a probability distribution $\pi_\theta(B^u|q)$.

3.4.1 Optimize the Prediction Module. To optimize the attention-based prediction network f_ϕ, we could have that,

$$\begin{aligned} \phi^* &= \arg\min_\phi (L_{ce} + \lambda L_r) \\ &= \arg\min_\phi \sum_u \sum_v E_{B^u \sim \pi_\theta(B^u|q)}[-LL(y_{uv}, f_\phi(B^u, u, v, c))] \\ &+ \frac{1}{2}\lambda \left(\|\Phi\|_2^2 \right) , \end{aligned} \tag{17}$$

where L_{ce} is cross entropy loss and L_r is regularization term. When we are optimizing the prediction network, the retrieval module remains unchanged, so if the function f is differential with respect to ϕ, the above optimization can be solved by typical stochastic gradient descent algorithm.

3.4.2 Optimize the Retrieval Module. For the retrieval module, only the feature selection model needs to be optimized. As a sampling process is evolved, we could not directly use SGD to optimize it. Instead, we use the REINFORCE [38, 42] algorithm to deal with the discrete optimization.

Specifically, while keeping the prediction network f_ϕ fixed, the feature selection model is optimized via performing its maximization:

$$\theta^* = \arg\max_\theta \sum_u \sum_v E_{B^u \sim \pi_\theta(B^u|q)}[LL(y_{uv}, f_\phi(B^u, u, v, c))], \tag{18}$$

we denote $LL(y_i, f_\phi(B_i^K, u_i, i_i, c_i))$ as $LL(\cdot)$ because it doesn't have parameter θ. For each query q, we denote objective function as J^q which is

$$J^q = E_{B^u \sim \pi_\theta(B^u|q)}[LL(\cdot)]. \tag{19}$$

We regard the retrieval module $\pi_\theta(B^u|q)$ as a policy and use the likelihood ratio to estimate its gradients as,

$$\begin{aligned} \nabla_\theta(J^q) &= \nabla_\theta E_{B^u \sim \pi_\theta(B^u|q)}[LL(\cdot)] \\ &= \sum_{B_i^u \in \mathcal{B}} \nabla_\theta \pi_\theta(B_i^u|q)[LL(\cdot)] \\ &= \sum_{B_i^u \in \mathcal{B}} \pi_\theta(B_i^u|q) \nabla_\theta \log(\pi_\theta(B_i^u|q))[LL(\cdot)] \\ &= E_{B^u \sim \pi_\theta(B^u|q)} \nabla_\theta \log(\pi_\theta(B^u|q))[LL(\cdot)] \\ &\simeq \frac{1}{L} \sum_{l=1}^{L} \nabla_\theta \log(\pi_\theta(B_l^u|q))[LL(\cdot)], \end{aligned} \tag{20}$$

which is an unbiased estimation on the gradients of Eq. 19.

As the uncertainty of the entire retrieval module actually comes from the feature selection model, we could derive that,

$$\pi_\theta(B^u|q) = \prod_{j=1}^{n} p(f_j), \tag{21}$$

where $f_j \in \{f_1, .., f_n\}$ and $p(f_j)$ is the sampling probability obtained in Eq. 7. So in Eq. 20 the gradient of θ can be further derive as

$$\nabla_\theta(J^q) \simeq \frac{1}{L} \sum_{l=1}^{L} \sum_{j=1}^{n} \nabla_\theta \log(p(f_j^l))[LL(\cdot)], \tag{22}$$

where f_j^l is j-th feature of l-th query. To train the model with a reward that has a better scale, we replace $LL(\cdot)$ with Relative Information Gain (RIG) as the reward function here. RIG is defined as $RIG = 1 - NE$ where NE is the normalized entropy [10]. NE is calculated as,

$$NE = \frac{LL(\cdot)}{p \log(p) + (1-p)\log(1-p)} \tag{23}$$

where p is the average experienced CTR.

3.4.3 Pseudo Code of Training Process. In this subsection, we give a detailed pseudo code of the training process. First we pre-train the prediction network with the initial feature selection model. After the pre-train, the two models are optimized in turn. Algorithm 1 shows the training process.

Algorithm 1 Training the UBR4CTR framework

Require: Dataset $\mathcal{D} = (\mathcal{U}_{target}, \mathcal{V}_{target}, \mathcal{C}_{target})$ containing all the target user-item-context triples; User history archive H_u.
Ensure: final CTR prediction $\hat{Y}$ between all the target user u and target item v.

1: Initialize all parameters.
2: Select the features and form the queries $\mathcal{Q} = \{q, ...\}$ for each prediction target $[u, v, c] \in \mathcal{D}$ using the initialized feature selection model.
3: Obtain the retrieved behaviors $\mathcal{B} = \{B^u, ...\}$ of the queries $\mathcal{Q}$ using the search engine as described in Section 3.2.2.
4: Train the attention-based prediction network using Eq. 17 for one epoch.
5: **repeat**
6: Train retrieval model using Eq. 18 for one epoch.
7: Select the features and form the queries $\mathcal{Q} = \{q, ...\}$ for each prediction target $[u, v, c] \in \mathcal{D}$ using the feature selection model.
8: Obtain the retrieved behaviors $\mathcal{B} = \{B^u, ...\}$ of the queries $\mathcal{Q}$ using the search engine as described in Section 3.2.2.
9: Train attention-based prediction network using Eq. 17 for one epoch.
10: **until** convergence

3.5 Model Analysis

In this section, we analyze the time complexity and feasibility of our method. We use $\mathcal{N}$ to denote the total number of all users' logged behaviors and use F to denote the total number of unique features (equivalent to term) that have ever appeared in the whole dataset. Then the average length of the posting lists in the user history archive is $\frac{\mathcal{N}}{F}$. Recall the searching operation described in Section 3.2.2 and Eq. 8, we first retrieve all the postings of features in q which takes $O(1)$ time. Then the interaction operation takes $O(T + K_q \cdot \frac{\mathcal{N}}{F})$ time where T is the average length of a user sequence and $K_q = K_u + K_v + K_c - 1$ is the upper bound of the number of selected features. The next scoring operations does not increase the complexity because it is linear to $O(T + K_q \cdot \frac{\mathcal{N}}{F})$. The complexity of the self-attention in feature selection model is $O(K_q^2)$. The attention-based prediction network takes $O(C)$ time where C is the cost of computing one Att operation in Eq. 13 because all the attention operations can be paralleled. These two constants can be ignored and the total time complexity of UBR4CTR is $O(T + K_q \cdot \frac{\mathcal{N}}{F})$.

4 EXPERIMENTS

In this section, we present our experimental settings and corresponding results in detail. We compare our model with several strong baselines and achieve the state-of-the-art performance. Furthermore, we have published our code for reproduction[1].

We start with three research questions (RQ) and use them to lead the following discussions.

- **RQ1** Does UBR4CTR achieves the best performance compared to other baselines?
- **RQ2** What is the convergence performance of Algorithm 1? Is the training process effective and stable?
- **RQ3** What is the influence of the retrieval module in UBR4CTR and how the retrieval size affect the performance?

[1]https://github.com/qinjr/UBR4CTR

Table 2: The dataset statistics.

Dataset	Users #	Items #	Interaction #	Avg. Seq. Length	Feature Field #
Tmall	424,170	1,090,390	54,925,331	129	9
Taobao	987,994	4,162,024	100,150,807	101	4
Alipay	498,308	2,200,291	35,179,371	70	6

4.1 Experimental Settings

4.1.1 Datasets. We use three real-world and large-scale datasets of users online behaviors from three different platforms of Alibaba Group. The statistics of the datasets can be found in Table 2.

Tmall [2] is provided by Alibaba Group which contains user behavior history on Tmall e-commerce platform from May 2015 to November 2015.

Taobao [46] is a dataset consisting of user behavior data retrieved from Taobao[3], one of the biggest e-commerce platforms in China. It contains user behaviors from November 25 to December 3, 2017, with several behavior types including click, purchase, add to cart and item favoring.

Alipay [4] is collected by Alipay which is an online payment application. The users online shopping behaviors are from July 1, 2015 to November 30, 2015.

Dataset Preprocessing. For UBR4CTR, the datasets are processed into the format of comma separated features. A line containing user, item and context features is treated as a behavior document. For baselines, the user behaviors are simply sorted by timestamp. As the datasets don't contain specific context features, we manually design some features using behavior timestamp to make it possible to capture periodicity. We design features such as season id (spring, summer, etc), weekend or not, and which half of the month it is.

Search Engine. After the datasets are preprocessed, they are inserted into a search engine using a comma separated tokenizer. We use Elastic Search [5] as the search engine client which is based on Apache Lucene[6].

Train & Test Splitting. We split the datasets using the time step. The training dataset contains the 1st to $(T-2)$th user behaviors, in which the 1st to $(T-3)$th behaviors are used to predict the behavior at $(T-2)$th step. Similarly, the validation set uses 1st to $(T-2)$th behaviors to predict $(T-1)$th behavior and the test set uses 1st to $(T-1)$th behaviors to predict Tth behavior.

Hyperparameters. The learning rate of feature selection model of UBR4CTR is searched from $\{1\times10^{-6}, 1\times10^{-5}, 1\times10^{-4}\}$, learning rate for attention based prediction network is from $\{1\times10^{-4}, 5\times10^{-4}, 1\times10^{-3}\}$ and the regularization term is from $\{1\times10^{-4}, 5\times10^{-4}, 1\times10^{-3}\}$. The search space of learning rate and regularization term for baseline models are the same with prediction network in

[2]https://tianchi.aliyun.com/dataset/dataDetail?dataId=42
[3]https://tianchi.aliyun.com/dataset/dataDetail?dataId=649
[4]https://tianchi.aliyun.com/dataset/dataDetail?dataId=53
[5]https://www.elastic.co
[6]http://lucene.apache.org

UBR4CTR. Batch size is from {100, 200} for all models. The hyper-parameters of each model are tuned and the best performances have been reported in Section 4.2.

4.1.2 Evaluation Metrics. We evaluate the CTR prediction performance with two widely used metrics. The first one is area under ROC curve (AUC) which reflects the pairwise ranking performance between click and non-click samples. The other metric is log loss. Log loss is to measure the overall likelihood of the test data and has been widely used for the classification tasks [24, 25].

4.1.3 Compared Baselines. We compare our framework and models with seven different strong baselines from both sequential CTR prediction and recommendation scenarios.

GRU4Rec [12] is based on GRU and it is the first work using the recurrent cells to model sequential user behaviors for session-based recommendation.

Caser [31] is a CNNs-based model that regards the user sequence as an image thus uses horizontal and vertical convolutional layers to capture temporal patterns of user behaviors.

SASRec [15] uses Transformer [32]. It regards the user behaviors as a sequence of tokens and uses self-attention mechanism and position embedding to capture the dependencies and relations between behaviors.

HPMN [23] is a hierarchical periodic memory network that is proposed to handle very long user historical sequence. Moreover, the user memory state can be updated incrementally.

MIMN [19] is based on Neural Turing Machine [6] which models multiple channels of user interests drifting. The model is implemented as a part of user interest center [19] which could model very long user behavior sequences.

DIN [45] is the first model that uses attention mechanism in CTR prediction of online advertising.

DIEN [44] uses two-layer RNNs with attention mechanism to capture evolving user interests. It uses the calculated attention values to control the second recurrent layer, which is called AUGRU.

UBR4CTR is our proposed framework and models described in Section 3.

4.2 Performance Comparison: RQ1

We conduct two groups of comparisons between our UBR4CTR and baseline models. In the first group of experiments, all of the models are using the same amount of user behaviors which are 20, 20, 12 for the three datasets respectively. The only difference is that baselines are using the most recent behaviors (about 20% of the total length) and UBR4CTR retrieve 20% of behaviors from the whole sequence. The experimental results are shown in Table 3.

From the table, we could find the following facts. (i) The performance is improved significantly compared to the baselines. AUC are improved by 4.1%, 10.9% and 22.3% on three datasets respectively, and log-loss are improved by 9.0%, 12.0% and 32.3% respectively. (ii) The vast improvement is a demonstration that the most recent behaviors do not embed enough temporal patterns so the baselines can not capture them effectively. Although some of the baselines are pretty complex and fancy, they cannot perform well if the patterns that they try to capture are not contained in the recent behavior sequence in the first place.

Table 3: The first group of results of CTR prediction in terms of AUC (the higher, the better) and log-loss (LL, the lower, the better). Bold values are the best in each column, while the second best values are underlined. Improvements are against the second best results.

Model	Tmall		Taobao		Alipay	
	AUC	LL	AUC	LL	AUC	LL
GRU4Rec	0.762	0.585	0.677	0.661	0.6131	0.699
Caser	0.762	0.579	0.673	0.657	0.655	0.676
SASRec	0.755	0.586	0.670	0.658	0.648	0.679
HPMN	0.763	0.579	0.668	0.660	0.615	0.703
MIMN	0.753	0.591	0.662	0.686	0.664	0.675
DIN	0.766	0.576	0.678	0.649	0.732	0.616
DIEN	0.775	0.567	0.677	0.659	0.730	0.616
UBR4CTR	**0.807**	**0.516**	**0.752**	**0.571**	**0.895**	**0.417**
Imprv.	4.1%	9.0%	10.9%	12.0%	22.3%	32.3%

Table 4: The second group of results of CTR prediction in terms of AUC (the higher, the better) and log-loss (LL, the lower, the better). Improvements are against the second best results.

Model	Tmall		Taobao		Alipay	
	AUC	LL	AUC	LL	AUC	LL
GRU4Rec	0.781	0.560	0.677	0.660	0.639	0.684
Caser	0.774	0.566	0.645	0.659	0.705	0.631
SASRec	0.769	0.578	0.669	0.654	0.711	0.637
HPMN	0.767	0.579	0.655	0.664	0.703	0.643
MIMN	0.759	0.590	0.659	0.659	0.719	0.634
DIN	0.791	0.546	0.605	0.679	0.856	0.506
DIEN	0.805	0.538	0.704	0.656	0.843	0.491
UBR4CTR	**0.807**	**0.516**	**0.752**	**0.571**	**0.895**	**0.417**
Imprv.	0.2%	4.1%	6.8%	12.7%	4.6%	15.1%

In the second group of experiment, we use different settings for the baselines and exactly the same settings for UBR4CTR as the first group of experiment. We feed the full-length sequences to all the baseline models which are 100, 100 and 60 respectively on three datasets. They are the maximum lengths of user behaviors in these datasets. And the sizes of the retrieved behaviors remain the same for UBR4CTR which are 20, 20 and 12. The results are shown in Table 4.

In Table 4, the performance of UBR4CTR is the same as that in Table 3 because we don't change any settings.

From the table, we could find the following facts. (i) UBR4CTR still has the best performance even though it uses 80% less behaviors than other baselines. This shows that longer sequence may has more noise and irrelevant information thus it is necessary to obtain only the most useful data out of the whole sequence. (ii) Most of the baselines achieve better performance than themselves compared to Table 3, especially DIN and DIEN. This shows that behaviors from

further history do contain richer information and patterns. And these patterns are easier to be captured using attention mechanism. (iii) Although the improvement of AUC is much smaller due to the better performance of baselines, log-loss still improves significantly. The reason is that the optimization objective of the retrieval module is RIG (equivalent to log-loss) after all.

4.3 Learning Process: RQ2

To illustrate the convergence of our framework, we plot the learning curves of UBR4CTR. In Figure 6, the upper three subplots are the AUC curves of the attentive prediction network when training on three datasets, respectively. Each step of the x-axis is corresponding to the iteration over 4% of the training set.

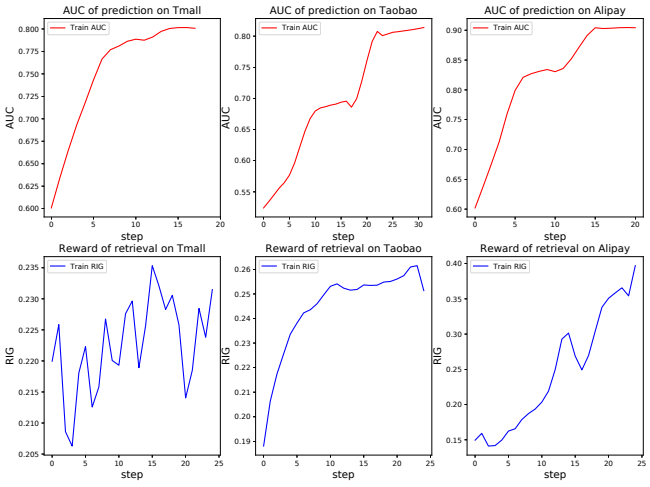

Figure 6: Learning curves of UBR4CTR.

The lower three sub-figures show the "reward" of the REINFORCE algorithm w.r.t the feature selection model of the retrieval module. The "reward" is essentially RIG which is a variant of log-likelihood. Each step of the x-axis means the iteration over 4% of the training set. The rewards increase during the training process implying that the feature selection model actually learns useful patterns.

From the AUC figures, we could find that our models converge effectively. For the prediction network, we can observe that there are flat areas in the middle of training process followed by a rapid increase, especially in the second and third AUC plots. Recall our training procedure described in Algorithm 1, the retrieval module is trained after the prediction network. It means that when the prediction network is about to converge, the retrieval module begins training, and after that, there will be a performance break though for the prediction network.

4.4 Extensive Study: RQ3

In this section, we conduct some extensive and ablation studies on our framework. Figure 7 illustrates the influence of different retrieval sizes on the prediction performance. From the figure, we can find that the fluctuation of AUC and log-loss is not very severe in terms of the absolute values. However, there exists an optimal size for each dataset. This reveals that smaller sizes may not contain enough behaviors and information, while too much retrieved behaviors are not always suitable for performance either. Because it will introduce too much noise.

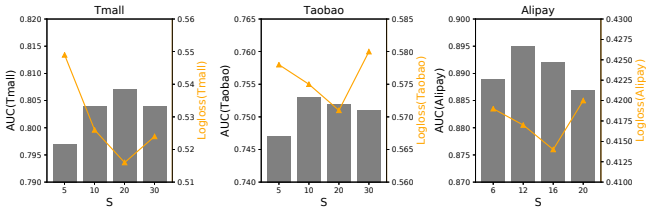

Figure 7: Influence of different retrieval size.

To illustrate the importance of the retrieval module of our framework, we plot the performance comparisons between the sum pooling model and attention network with&without user behaviors retrieval. Sum pooling model just uses a very simple sum pooling operation on user behaviors which means the $\alpha_i = 1$ in Eq. 11. The results are shown in Figure 8. From the figure, we find that the sum pooling model without retrieving (SP) performs very poorly. Once equipped with the retrieval module, the performance of it increases significantly (UBR_SP). The same phenomenon applies for attention network which performance is largely improved when equipped with behavior retrieval module (ATT vs. UBR4CTR).

5 DEPLOYMENT FEASIBILITY

The deployment of UBR4CTR has been made on the engineering schedule of a daily item recommendation platform under a mainstream bank company. In this section, we mainly discuss the feasibility of industrial deployment of UBR4CTR framework. First, it is not difficult to switch the current model pipeline to UBR4CTR because the main change brought from UBR4CTR is the how the historical user behaviors is obtained. To update the model pipeline to UBR4CTR, a search engine of historical user behaviors needs to be built, while the whole CTR prediction model pipeline remain almost the same but adding an additional retrieval module. As Figure 2 shows, the prediction module is not different from that of the traditional solutions.

Efficiency is another essential concerns in industrial applications. We analyze the time complexity of UBR4CTR which is $O(T+K_q \cdot \frac{N}{F})$ in Section 3.5. For most of the sequential CTR models which are normally based on RNN, time complexity of them is $O(C \cdot T)$ where T is the average length of user sequences and C is the cost of one operation (e.g. GRU rolling). The time complexity of UBR4CTR is not totally infeasible because the term $\frac{N}{F}$ is very closed to a constant as F is a big number and it will slow the increase of this term.

From the perspective of system load, UBR4CTR is better because it does not require maintaining all the T behaviors in memory which is a common practice for the traditional methods.

Moreover, we compare the actual inference time between our UBR4CTR and other sequential CTR baselines in experiments. The average inference time of the models are illustrated in Figure 9. The time is calculated by dividing the overall time (only the time that

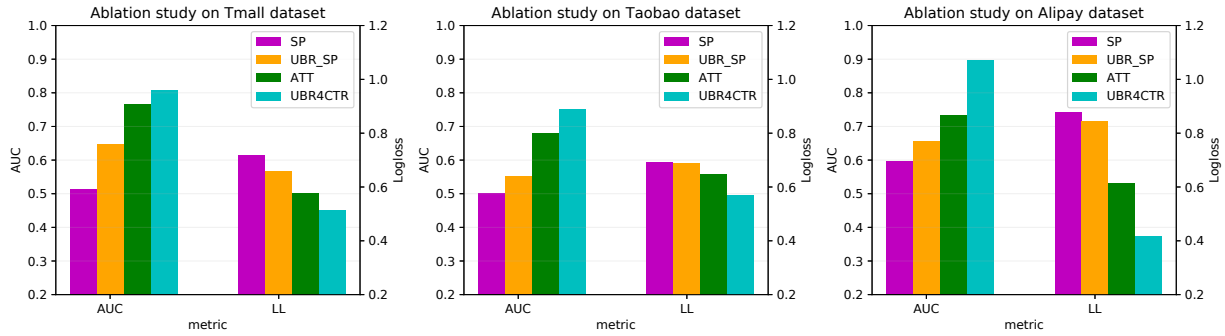

Figure 8: Ablation study on the influence of retrieval module. *Note:* AUC, the higher, the better; log-loss, the lower, the better

contains the forward computations and behavior searching) on test dataset with the number of prediction targets. From the figure, we could find that the absolute value of UBR4CTR's inference time on three datasets is less than 1ms which is efficient enough for online serving [34]. The inference time of UBR4CTR is the longest among all the sequential CTR models but the gap is not that large. Particularly, compared with DIEN which has been deployed in Alibaba online advertising platform [44], the average inference time of UBR4CTR is about 15% to 30% longer, which can be optimized via further infrastructure implementation.

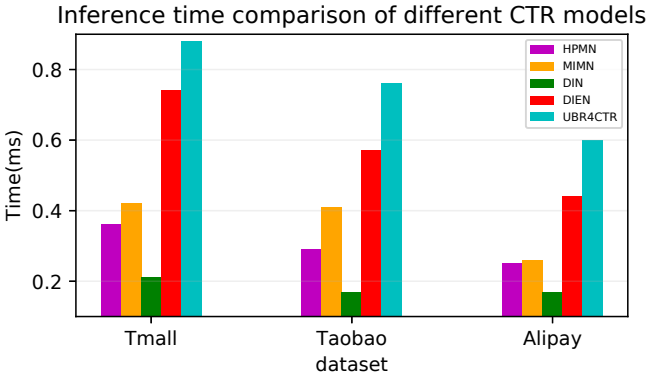

Figure 9: Inference time of different models.

6 RELATED WORK

6.1 User Response Prediction

For user response prediction, there are two main streams of methods. One stream is about modeling the interactions of multiple categorical features. The key point of these models is to design structure that could capture the cross feature interactions. Factorization machine [27] is the pioneer model which uses matrix factorization in CTR prediction task and a lot of variants of FM [14, 30, 41] are proposed in recent years. Except for the early feature interaction models, deep neural networks are also used in CTR prediction. Wide&Deep [4] is the first deep learning model which effectively transform the high dimensional discrete features into dense representation and achieve good prediction performance. After Wide&Deep and FM, more and more models which combine the feature interaction structure and deep neural networks are proposed. DeepFM [7] uses both FM and DNN to improve the CTR prediction performance. DeepCross [36] and PNN [21] automatically models the cross feature interactions by outer product and inner product respectively. There are similar models such as xDeepFM [17], FNN [43] etc.

The other stream of models focuses more on mining temporal patterns from sequential user behaviors. DIN [45] is an attention-based network which attributes different weights on different items that user has interacted with. DIEN [44] utilizes two layers of GRU and attention mechanism to capture evolving user interests. HPMN [23] is a memory network-based method which model very long sequences for user response prediction. MIMN [19] is based on neural Turing machine [6] that models the multiple channels of user evolving interests.

6.2 Sequential User Modeling

Sequential user modeling is about capture user's drifting dynamics of behaviors. It is a research hotspot for recommender systems recently. Multiple types of model are proposed. The first is temporal collaborative filtering [16] which considers the drifting user preferences. The second type is based on Markov chains [8, 9, 28] which models the user state dynamics in an implicitly way and drive the future behaviors. The third category is based on deep learning. There are RNN-based models [2, 11–13, 18, 33, 39] that regard user behaviors as sequence of tokens, CNN-based models [15, 31] which regard the behaviors as an image and Transformer-based models [15]. Furthermore, there are models [40] that not only utilize user-side sequence but also item-side sequence. Qin et al. [20] propose dual side neighbor based CF for sequential user modeling. Memory networks are also used for sequential user modeling [3, 5, 19, 35] which aim to memorize longer sequence of user behaviors.

7 CONCLUSION AND FUTURE WORK

In this paper, we propose the UBR4CTR framework of user response prediction. The retrieval module of our framework generates a query to search from the whole user behaviors archive to retrieve the most useful behavioral data for prediction. The retrieved data is then used by an attention-based deep network to make the final prediction. Our framework overcomes the practical problems of traditional framework which simply uses most recent behaviors and significantly improves the CTR prediction performance. The

deployment of UBR4CTR has been made on the engineering schedule of a daily item recommender system of a mainstream bank company.

For the future work of research, we will put more efforts on distributed training algorithm which will make the framework more efficient to train in a mini-batch manner. Furthermore, we will explore new effective indexing and retrieval methods for storing and searching the user behavioral archive.

Acknowledgement. The corresponding author Weinan Zhang thanks the support of National Natural Science Foundation of China (61702327, 61772333, 61632017).

REFERENCES

[1] Deepak Agarwal, Bee-Chung Chen, and Pradheep Elango. 2009. Spatio-temporal models for estimating click-through rate. In *WWW*.
[2] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. 2018. Latent Cross: Making Use of Context in Recurrent Recommender Systems. In *WSDM*.
[3] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *WSDM*.
[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.
[5] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative Memory Network for Recommendation Systems. In *SIGIR*.
[6] Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401* (2014).
[7] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247* (2017).
[8] Ruining He, Chen Fang, Zhaowen Wang, and Julian McAuley. 2016. Vista: a visually, socially, and temporally-aware model for artistic recommendation. In *RecSys*.
[9] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *ICDM*.
[10] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *ADKDD*.
[11] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. *CIKM* (2018).
[12] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. *ICLR* (2016).
[13] How Jing and Alexander J Smola. 2017. Neural survival recommender. In *WSDM*.
[14] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 43–50.
[15] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. *ICDM* (2018).
[16] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *KDD*.
[17] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1754–1763.
[18] Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. 2016. Context-Aware Sequential Recommendation. In *ICDM*.
[19] Qi Pi, Weijie Bian, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Practice on long sequential user behavior modeling for click-through rate prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2671–2679.
[20] Jiarui Qin, Kan Ren, Yuchen Fang, Weinan Zhang, and Yong Yu. 2020. Sequential Recommendation with Dual Side Neighbor-based Collaborative Relation Modeling. In *Proceedings of the Thirteenth ACM International Conference on Web Search and Data Mining (WSDM '20)*. ACM.
[21] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *ICDM*.
[22] Yanru Qu, Bohui Fang, Weinan Zhang, Ruiming Tang, Minzhe Niu, Huifeng Guo, Yong Yu, and Xiuqiang He. 2018. Product-based neural networks for user response prediction over multi-field categorical data. *ACM Transactions on Information Systems (TOIS)* 37, 1 (2018), 1–35.
[23] Kan Ren, Jiarui Qin, Yuchen Fang, Weinan Zhang, Lei Zheng, Weijie Bian, Guorui Zhou, Jian Xu, Yong Yu, Xiaoqiang Zhu, et al. 2019. Lifelong Sequential Modeling with Personalized Memorization for User Response Prediction. *SIGIR*.
[24] Kan Ren, Weinan Zhang, Ke Chang, Yifei Rong, Yong Yu, and Jun Wang. 2018. Bidding Machine: Learning to Bid for Directly Optimizing Profits in Display Advertising. *TKDE* (2018).
[25] Kan Ren, Weinan Zhang, Yifei Rong, Haifeng Zhang, Yong Yu, and Jun Wang. 2016. User response learning for directly optimizing campaign performance in display advertising. In *CIKM*.
[26] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten de Rijke. 2018. RepeatNet: A Repeat Aware Neural Recommendation Machine for Session-based Recommendation. *arXiv preprint arXiv:1812.02646* (2018).
[27] Steffen Rendle. 2010. Factorization machines. In *ICDM*.
[28] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*.
[29] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at TREC-3. *Nist Special Publication Sp* 109 (1995), 109.
[30] Anh-Phuong Ta. 2015. Factorization machines with follow-the-regularized-leader for CTR prediction in display advertising. In *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE, 2889–2891.
[31] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*.
[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.
[33] Kiewan Villatel, Elena Smirnova, Jérémie Mary, and Philippe Preux. 2018. Recurrent Neural Networks for Long and Short-Term Sequential Recommendation. *KDD* (2018).
[34] Jun Wang, Weinan Zhang, and Shuai Yuan. 2017. Display Advertising with Real-Time Bidding (RTB) and Behavioural Targeting. *Now Publisher* (2017).
[35] Qinyong Wang, Hongzhi Yin, Zhiting Hu, Defu Lian, Hao Wang, and Zi Huang. 2018. Neural Memory Streaming Recommender Networks with Adversarial Training. In *KDD*.
[36] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. 1–7.
[37] Gerhard Widmer and Miroslav Kubat. 1996. Learning in the presence of concept drift and hidden contexts. *Machine learning* 23, 1 (1996), 69–101.
[38] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.
[39] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *WSDM*.
[40] Qitian Wu, Yirui Gao, Xiaofeng Gao, Paul Weng, and Guihai Chen. 2019. Dual Sequential Prediction Models Linking Sequential Recommendation and Information Dissemination. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 447–457.
[41] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *IJCAI* (2017).
[42] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*.
[43] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep Learning over Multi-field Categorical Data: A Case Study on User Response Prediction. *ECIR* (2016).
[44] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5941–5948.
[45] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *KDD*.
[46] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. 2018. Learning Tree-based Deep Model for Recommender Systems. In *KDD*.