End-to-End User Behavior Retrieval in Click-Through Rate Prediction Model

Qiwei Chen*
Changhua Pei*
chenqiwei.cqw@alibaba-inc.com
changhua.pch@alibaba-inc.com
Alibaba Group.
Beijing, China

Shanshan Lv
lss271346@alibaba-inc.com
Alibaba Group.
Beijing, China

Chao Li
ruide.lc@alibaba-inc.com
Alibaba Group.
Beijing, China

Junfeng Ge
beili.gjf@alibaba-inc.com
Alibaba Group.
Beijing, China

Wenwu Ou
santong.oww@taobao.com
Alibaba Group.
Beijing, China

ABSTRACT

Click-Through Rate (CTR) prediction is one of the core tasks in recommender systems (RS). It predicts a personalized click probability for each user-item pair. Recently, researchers have found that the performance of CTR model can be improved greatly by taking user behavior sequence into consideration, especially long-term user behavior sequence. The report on an e-commerce website shows that 23% of users have more than 1000 clicks during the past 5 months. Though there are numerous works focus on modeling sequential user behaviors, few works can handle long-term user behavior sequence due to the strict inference time constraint in real world system. Two-stage methods are proposed to push the limit for better performance. At the first stage, an auxiliary task is designed to retrieve the top-k similar items from long-term user behavior sequence. At the second stage, the classical attention mechanism is conducted between the candidate item and k items selected in the first stage. However, information gap happens between retrieval stage and the main CTR task. This goal divergence can greatly diminishing the performance gain of long-term user sequence. In this paper, inspired by Reformer, we propose a locality-sensitive hashing (LSH) method called ETA (End-to-end Target Attention) which can greatly reduce the training and inference cost and make the end-to-end training with long-term user behavior sequence possible. Both offline and online experiments confirm the effectiveness of our model. We deploy ETA into a large-scale real world E-commerce system and achieve extra 3.1% improvements on GMV (Gross Merchandise Value) compared to a two-stage long user sequence CTR model.

*Both authors contributed equally to this research.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

KEYWORDS

recommendation, behavior sequence, real-time retrieval

ACM Reference Format:
Qiwei Chen, Changhua Pei, Shanshan Lv, Chao Li, Junfeng Ge, and Wenwu Ou. 2021. End-to-End User Behavior Retrieval in Click-Through Rate Prediction Model. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Recommendation systems (RS) are widely deployed to address the information overload problem. Among all those deep learning models used in RS, Click-Through Rate (CTR) prediction model is one of the most important one. Both industry and academy pay much attention on improving the AUC (area under the ROC curve) of CTR model in order to improve the online performance of RS. In the past decade, the performance of CTR model has been improved greatly. One of the remarkable milestones is the introducing of *user behavior sequence*, especially the long-term user behavior sequence [20–22, 35, 36]. According to the report of [24], 23% of users in an e-commerce website have more than 1000 clicks during the past 5 months. How to effectively utilize the massive and informative user behaviors has become more and more important, which is also the goal of this paper.

Various methods are proposed to model sequential user behavior data. Early approaches, such as the sum/mean pooling methods, RNN-based methods [4, 5, 11], CNN-based methods [14, 27] and self-attention-based methods [12, 28] encode different length of user behavior sequence into fixed dimensional hidden vector. However, they fail to capture the dynamic local interests of an user when scoring different candidate items. These methods also introduce noises by encoding all user historical behaviors. To overcome the drawbacks of the global pooling methods, DIN [36] is proposed to generate various user sequence representations according to different candidate items by **target attention** mechanism, where the target candidate item acts as query $\boldsymbol{Q}$ and each item in the sequence acts as key $\boldsymbol{K}$ and value $\boldsymbol{V}$. However, due to the expensive computation and storage resources, DIN uses the recent 50 behaviors for target attention, which ignores the resourceful information in the long user behavior sequence and is obviously sub-optimal.

Recently, methods such as SIM [21] and UBR4CTR [22] are proposed to capture user dynamic interests from longer user behavior sequence and become the SOTA (state-of-the-art) methods. These methods act in a two-stage way. In the first stage, an auxiliary task is designed to retrieve the top-k similar items from long-term user behavior sequence, such that the top-k similar items are prepared in advance. In the second stage, the *target attention* mechanism is conducted between target item and k items selected in the first stage. However, the information used for retrieval stage is divergent or outdated with the main CTR model. For example, UBR4CTR [22] and SIM [21] use attributes such as category to select items from user behavior sequence which share the same attribute with target candidate item, which is divergent with the target of CTR model. SIM [21] also tried to building an offline inverted index based on the pre-trained embedding. During training and inference, the model can search the top-k "similar" items. But most of the CTR model is in online learning paradigm and the embedding is updated continuously. Thus the pre-trained embedding in offline inverted index is outdated compared with the embedding in online CTR model. Whether the divergent target or the outdated retrieval vector, it will prevent the long-term user behavior sequence to be full utilized.

In this paper, we propose a method called ETA, enabling end-to-end long-term user behavior retrieval to mitigate the aforementioned information gap (*i.e.,* divergent target and outdated embedding) in CTR prediction task. We use SimHash to generate a fingerprint for each item in user behavior sequence. Then the hamming distance is used to help select top-k items for target attention. Our method reduces the retrieval complexity from $O(L * B * d)$ multiplication to $O(L * B)$ hamming distance calculation, where L is the length of behavior sequence, B is the number of candidate items to be scored by CTR model at each recommendation and d is the dimension of item embedding. The reduction of complexity helps us removing the offline auxiliary model and conducting real-time retrieval during training and serving procedure. This improves the ranking improvements greatly compared with SOTA models. The contributions of our paper can be summarized in three-fold.

- We propose an End-to-end Target Attention method for CTR prediction task, which is called as **ETA**. To the best of our knowledge, ETA is the first work to model the long-term user behavior sequence with CTR model in an end-to-end way.
- Both offline experiments and online A/B tests show that ETA achieves significant performance improvements compared with the SOTA models. We get an extra 3.1% improvements on GMV after deploying ETA into a large-scale real world E-commerce platform when compared with a two-stage CTR model.
- Comprehensive ablation studies are conducted to reveal the hands-on practical experiences for better modeling sequential user behaviors under the limitation of inference time constraint.
- Our method can also be extended to other scenario to other models which need to handle extreme long sequence, *e.g.,* long sequence time-series forecasting models.

2 RELATED WORK

CTR prediction task is one of the crucial tasks in recommender systems, online advertising and information retrieval. The CTR model predicts the probability of an user clicking on a certain target item. The output probability can be used as the ranking score for the downstream ranking tasks. The accuracy of CTR model can greatly affect the online performance of online systems. For example, in our online RS, 0.1% AUC improvement of CTR model can bring millions of real-world clicks and revenue. Tremendous works focus on improving the accuracy of CTR model in different ways, which can be divided into three categories: **feature interaction**, **user behavior sequence** and **long-term user behavior sequence**.

Feature Interaction: The intuition of feature interaction is to memorize the co-occurrence pattern in the feature space together with label. For example, a feature $AND(user\_installed\_app = netflix, impression\_app = pandora)$ can better capture the pattern for an user who clicked or did not click a certain recommended App. A series of works are published to model the feature interactions more effectively. The representative works are FM[25], FFM[19], GBDT+LR[9], Wide&Deep[3], FNN[33], AFM[30], DeepCross[29], DeepFM[8], PNN[23] and xDeepFM[17]. Various differences can be found between any two models, *e.g.*, whether the deep learning technique is used, whether the embedding of weights and features are shared or whether feature engineering is needed.

User Behavior Sequence: The user behavior sequence is highly personalized for each user and contains spatio-temporal user interest information. Introducing the sequential user behaviors into CTR model is a remarkable milestone. Youtube[6] uses watched video sequence and search tokens in their model to capture the user interests. To better extract user interests from the user behavior sequence, various models are proposed, including CNN [27, 32], RNN [10, 35], Attention [7, 26] and Capsule Network [15]. However, the user interest vectors learned from the above models are global for a certain user. DIN [36] proposes an attention based method called target attention to capture the diverse local interests for a certain user facing with different target items.

Long-term User Behavior Sequence: Despite the powerful ability to capture the diverse user interests, the computation of target attention is costly. The strict limit of online inference time prevents DIN-like models from using longer user sequence. MIMN [20] can handle long-term user behavior sequence by decoupling the user interest modeling with the rest of CTR task. The user interest vector is updated offline in an asynchronous way whenever a new behavior is observed. As there is no inference time limit offline, MIMN can model any sequence lengths theoretically. However, MIMN can not learn various user interest vectors for different target items. SIM [21] and UBR4CTR [22] defeat MIMN in CTR task and become the SOTA models. Both SIM and UBR4CTR adopt two-stage architecture to model long-term user behavior sequence. At first stage, an auxiliary task is designed to retrieval the top-k similar items from long-term user behavior sequence. At second stage, target attention is conducted between target item and k items selected in the first stage.

Besides the above related works on CTR prediction task, plenty of works aim to improve the efficiency and effectiveness of transformer. Reformer [13] and Informer [37] are the most relevant

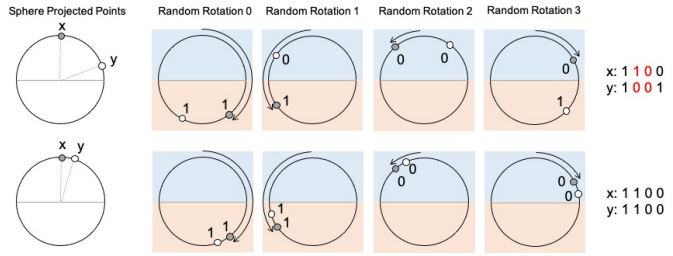

Figure 1: Illustration of *Locality-sensitive* of SimHash for two vectors x and y. The detailed implementation is shown in Algorithm 1. Each d-dimensional vector is converted to a m-length signature vector. Each random rotation here can be regarded as one "hash function". The rotation is implemented by multiplying a random hash column vector $H(i)$ (Line 2 in Algorithm 1). After random rotation, the spherical points are projected to signed axes (Line 3-7 in Algorithm 1). Here we use 4 hash function and two projection axes to map each vector into 4 binary bits. We can observe that only those vectors who are close to each other can share more same 0/1(s), which is shown at the bottom part.

works. However, they only focus on the optimization of classical transformer and cannot be directly used on CTR prediction task because of the strict inference time constraint for large scale online recommendation systems.

3 PRELIMINARIES

In this section, we first give the formulation of CTR prediction task. Then we introduce how the fingerprint of a d-dimensional embedding vector is generated by SimHash mechanism.

3.1 Formulation of CTR Prediction Task

CTR prediction task is widely deployed in online advertising, recommender systems and information retrieval. It aims to solve the following problem:

Given an impression j where an item is displayed to an user, predict the probability of user click (labeled as y_j) using the feature vector x_j.

$$p_j = P(y_j = 1 | \, x_j; \theta); j \in \mathcal{I}. \tag{1}$$

CTR task is usually modeled as a binary classification problem. For each impression $j \in \mathcal{I}$, a binary label y_j is recorded according to whether the item is clicked or not. Then CTR model is trained in a supervised way to minimize the cross entropy loss, which is shown as Equation 2, where N is the number of impressions. x_j and y_j are feature vector and label of impression j respectively. θ represents the trainable parameters of CTR model. For ease of presentation, we list the notations used in this paper in Table 1.

$$\mathcal{L}_{CTR}(\theta) = -\frac{1}{N} \sum_{j=1}^{N} \Big(y_j * log(p_j) + (1 - y_j) * log(1 - p_j) \Big). \tag{2}$$

Algorithm 1: Pseudo-code of SimHash algorithm.

Input : A d-dimensional embedding vector $\boldsymbol{e}_k \in \mathbb{R}^{1 \times d}$
A fix random hash matrix $\boldsymbol{H} \in \mathbb{R}^{d \times m}$, each column can be regraded as one hash function.
Output: A binary signature vector $\boldsymbol{sig}_k \in \mathbb{R}^{1 \times m}$ for $\boldsymbol{e}_k$.

1 **for** $i \leftarrow 0$ **to** $m - 1$ **do**
2 $\boldsymbol{sig}_k[i] = \sum_{j=1}^{d} \text{sgn}(\boldsymbol{e}_k[j] * H[j][i])$
3 **if** $\boldsymbol{sig}_k[i] > 0$ **then**
4 $\boldsymbol{sig}_k[i] = 1$
5 **else**
6 $\boldsymbol{sig}_k[i] = 0$
7 **end if**
8 **end for**
9 **return** $\boldsymbol{sig}_k$

3.2 SimHash

SimHash algorithm is first proposed by [2] and one of the well-known application is [18] which detects duplicate web pages by SimHash based fingerprints. SimHash function takes the embedding vector of an item as input and generates its binary fingerprint. Algorithm 1 shows the pseudo code of one possible SimHash implementation. SimHash satisfies the *Locality-sensitive* properties: the outputs of SimHash are similar if the input vectors are similar to each other, which is illustrated in Figure 1. Each random rotation in Figure 1 can be regarded as one "hash function". The rotation is implemented by multiplying the input embedding vector with a random projection column vector $\boldsymbol{H}(i)$, which is shown in Line 2 of Algorithm 1. After random rotation, the spherical points are projected to signed axes (Line 3-7 in Algorithm 1). In Figure 1, we use 4 hash functions and two projection axes to map each vector into a signature vector with 4 elements. Each element in the signature vector is either 1 or 0. This vector can further be decoded using an integer to save storage cost and to speed up the following hamming distance calculation. From Figure 1, we can observe that nearby embedding vectors can get the same hashing signature with high probability (see the bottom part of Figure 1 compared with the upper part of Figure 1). This observation is the so called "locality-sensitive" properties. With the local sensitive properties, the similarity between the embedding vectors can be replaced by the similarity between the hashed signature. In other words, the inner product between two vectors can be replaced by hamming distance. It is noteworthy that the SimHash algorithm is not sensitive to the selection each rotation "hashing function". Any fixed random hash vectors are enough (see $\boldsymbol{H}(i)$ in Algorithm 1). It is easy to implement and can be easily applied to batches of embedding vectors.

4 MODEL

In this section, we first introduce the detailed architecture of our **ETA** (End-to-End Target Attention) model. Then we introduce different sub-modules of ETA model. At last, we introduce the hands-on experiences for the deployment of ETA.

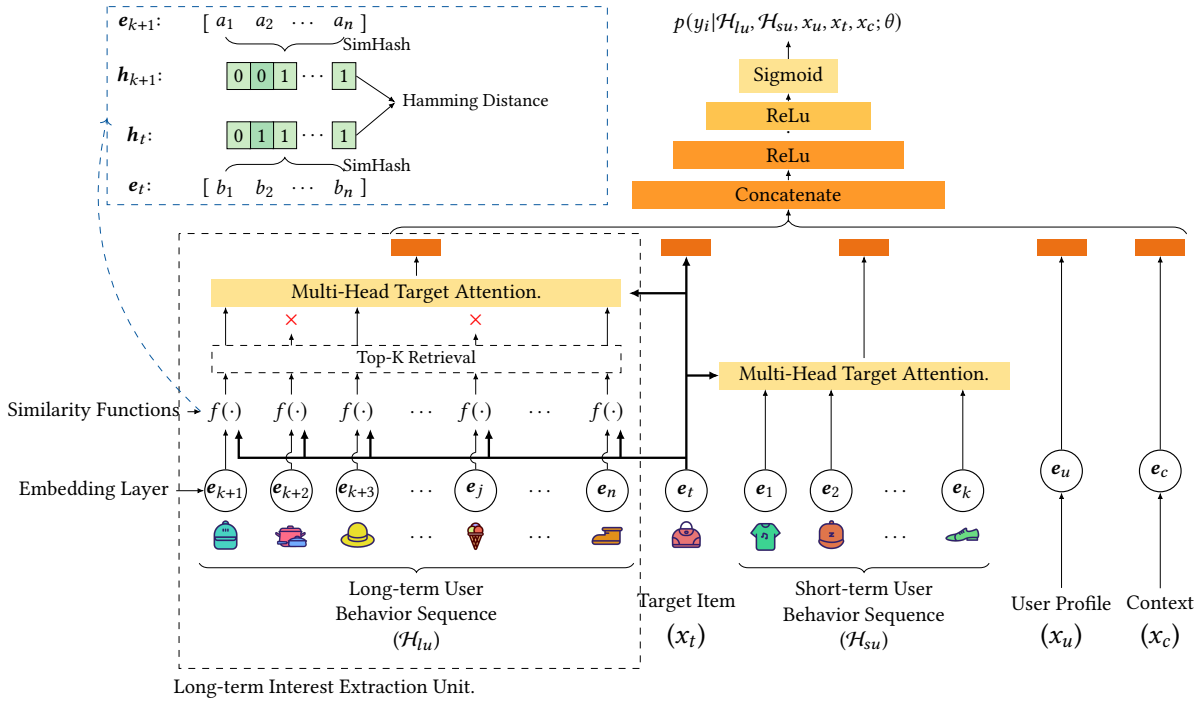

Figure 2: The illustration of our ETA (End-to-end Target Attention) model. $e_{k+1} \in \mathbb{R}^{d\times 1}$ and $e_t \in \mathbb{R}^{d\times 1}$ represent the embedding vectors of behavior item k+1 and target item t. d is the dimension of item embedding. h_{k+1} and h_t are the fingerprints of item k+1 and t generated by SimHash function. A d-dimensional embedding e_{k+1} can be hashed into a m-bit integer. Note that the projections of embedding vectors e_{k+1} and e_t before SimHash are omitted in the figure for clarity. The other notations can be find in Table 1.

4.1 Model Overview

As shown in Figure 2, our model takes user/item-side features as input and outputs the click probability of a certain user-item pair. $\mathcal{H}_{lu}$, $\mathcal{H}_{su}$, x_u, x_t and x_c are raw input features. θ represents the trainable parameters. Having these features, we use *Long-term Interest Extraction Unit* (Section 4.4), *Multi-head Target Attention* (Section 4.3) and *Embedding Layer* (Section 4.2) to convert $\mathcal{H}_{lu}$, $\mathcal{H}_{su}$, x_u, x_t and x_c into hidden vectors respectively. Then the hidden vectors are concatenated together and are fed into the MLP (Multi-layer Perception) part. At the last layer of MLP, sigmoid function is used to map the hidden vector into a scalar $p(y_j|\mathcal{H}_{lu}, \mathcal{H}_{su}, x_u, x_t, x_c; \theta)$ which represents the click probability of a certain user-item pair. This probability can be used as the ranking score for the downstream tasks.

4.2 Embedding Layer

For different types of features, we adopt different embedding techniques. The raw input features are mainly divided into two types: the categorical features and the numerical features. In our model, we use one-hot encoding for categorical features. For numerical features, we first divide the features into different numerical buckets. Then we apply one-hot encoding to identify different buckets, which is the similar way with [16]. Note that the one-hot encoding vectors can be extremely sparse because there are billions of item ids. Thus we map all one-hot embedding vectors into low-dimensional hidden vectors to reduce the number of parameters. We use $\boldsymbol{e}_i \in \mathbb{R}^{d\times 1}$ to represent the embedding vectors of item i. All the embedding vectors of user behavior items are then packed together into a matrix $\boldsymbol{E}_s \in \mathbb{R}^{L\times d}$, which is shown in Equation 3. L is the length of user behavior sequence and d is the embedding size.

$$\boldsymbol{E}_s = \begin{bmatrix} \boldsymbol{e}_1^\top \\ \boldsymbol{e}_2^\top \\ \vdots \\ \boldsymbol{e}_L^\top \end{bmatrix}. \tag{3}$$

4.3 Multi-head Target Attention

Multi-head attention is first proposed by [28] and is widely applied on CTR prediction tasks [21, 22, 30, 31, 34]. In CTR prediction task, target item acts as query ($\boldsymbol{Q}$) and each item in user behavior sequence acts as key ($\boldsymbol{K}$) and value ($\boldsymbol{V}$). We call this multi-head attention structure as *multi-head target attention*, which is abbreviated as TA. The calculation of TA is shown in Equation 4. The main part of TA is dot-product attention, which is shown in Equation 5. The dot-product attention is made up of two steps. Firstly, similarities are calculated between each behavior item and target item according to their embedding matrices $\boldsymbol{Q}$ and $\boldsymbol{K}$. Secondly, the normalized similarities are used as the attention weights to

Table 1: Notations used in this paper.

Notation.	Description.
$\mathcal{I}$	The set of impressions.
p_j	The click probability of a certain impression j.
y_j	The label of click on impression j.
x_j	The feature vector on impression j.
θ	The parameters of CTR model.
d	The dimension of item embedding.
$\boldsymbol{e} \in \mathbb{R}^{d\times 1}$	The item embedding vector.
$\boldsymbol{h} \in \mathbb{R}$	The hashed fingerprint of embedding vector $\boldsymbol{e}$.
sig_i	The bit vector generated by i^{th} hash function.
m	The bit-length of hashed fingerprint $\boldsymbol{h}$.
$\mathcal{H}_{lu}$	Long-term user behavior sequence.
$\mathcal{H}_{su}$	Short-term user behavior sequence.
x_u	Raw features of user profile.
x_c	Raw features of context information.
x_t	Raw features of target item.
L	The length of long-term user behavior sequence.
B	The number of candidate items to be predicted in each user request.
$\boldsymbol{e}_u, \boldsymbol{e}_c, \boldsymbol{e}_t$	The embedding vectors of user profile, context and target item respectively.
$\boldsymbol{E}_s \in \mathbb{R}^{L\times d}$	Embedding matrix of long-term user behavior sequence.
$\boldsymbol{E}_t \in \mathbb{R}^{1\times d}$	Embedding matrix of target item.
$f(\cdot)$	The similarity function of two embedding vectors.

calculate the weighted sum embedding of all the behavior items, whose embedding matrix is represented as $\boldsymbol{V}$.

$$\begin{aligned} \text{TA}(\boldsymbol{E}_t, \boldsymbol{E}_s) &= \text{Concat}(\text{head}_1, ..., \text{head}_h)\boldsymbol{W^O}, \\ \text{where head}_i &= \text{Attention}(\boldsymbol{E}_t\boldsymbol{W}_i^Q, \boldsymbol{E}_s\boldsymbol{W}_i^K, \boldsymbol{E}_s\boldsymbol{W}_i^V), \end{aligned} \tag{4}$$

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V}, \tag{5}$$

where $\boldsymbol{E}_t \in \mathbb{R}^{1\times d}, \boldsymbol{E}_s \in \mathbb{R}^{L\times d}$ are input embedding matrices for target item and behavior sequence respectively. L is sequence length and d is the embedding size of hidden vector for each behavior item. Note that we only select one piece of sample instead of a batch of samples for clarity. Matrices $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ represent queries, keys and values respectively. d_k, d_q, d_v is the embedding size for each row vector of $\boldsymbol{K}, \boldsymbol{Q}, \boldsymbol{V}$. $\sqrt{d_k}$ is used to avoid large value of the inner product. *softmax* function is used to convert the value of inner-product into the adding weight of the value vector $\boldsymbol{V}$. $\boldsymbol{W}_i^Q \in \mathbb{R}^{d\times d_k}, \boldsymbol{W}_i^K \in \mathbb{R}^{d\times d_k}, \boldsymbol{W}_i^V \in \mathbb{R}^{d\times d_v}$. $\boldsymbol{W^O} \in \mathbb{R}^{hd_v\times d}$ is the projection matrix. h is the number of headers.

4.4 Long-term Interest Extraction Unit

This part is the main contribution of our ETA model. It extends the encoding length of user behavior sequence from tens to thousands or longer to capture the long-term user interests. As mentioned before, the complexity of multi-head target attention is $O(L * B * d)$ where L is the length of user sequence, B is the number of candidate items and d is the representation dimension. In large-scale online system, B is close to 1000 and d is close to 128. Thus directly conducting multi-head target attention on thousands of long-term user behaviors is infeasible.

According to Equation 5, softmax is dominated by the largest elements, for each query we only need to focus on the keys that are closest to the query, which is also confirmed by [13, 21, 22]. Thus we can first retrieval top-k items from the behavior sequence and conduct the multi-head target attention on these k behaviors. However, a good retrieval algorithm should satisfy two constraints: 1) the target of retrieval part should keep the same with the whole CTR model. Only in this way can the top-k retrieved items contribute the most to the CTR model. 2) the retrieval time should satisfy the strict inference time limit to make sure the algorithm can be applied in real-world system to serve millions of request per second. We compare different retrieval algorithms in Table 2. SIM [21] and UBR4CTR [22] build offline inverted index to enable quick searching during training and inference. However, the inputs they used for building the index are attribute information (*e.g.,* category) or pre-trained embedding of items, which are different with the embedding used in CTR model. This gap violate the above constraint 1) and may lead to performance degradation. If we directly use the embedding in CTR model and search the k-nearest neighbor by inner product, $O(L * B * d)$ multiplications are needed and the inference time increases greatly. d is the dimension of the embedding vector. L and B are the number of behavior items and target items respectively. This will violate the above constraint 2) and can not be deployed online. Our ETA uses SimHash to convert the inner product of two vectors into hamming distance calculation, which is shown in Figure 2. This makes it possible to be deployed in real world recommendation system. Besides, the local sensitive property of SimHash ensures that the fingerprint can always keep in sync with the original embedding in CTR model. The evaluation in Section 5 shows that this compatibility can greatly improve the performance. How to choose the right hash function and the joint learning between the retrieval part and the rest part of ETA will be explained in Section 4.5.2 and Section 4.5.1.

After SimHash function and hamming distance layer, top-k similar behavior items are selected from $\mathcal{H}_{lu}$, then the aforementioned multi-head target attention is conducted to generate the hidden vector. This vector acts as the representation of long-term user interest and is fed into the MLP (Multi-layer Perception) layers together with other vectors. The formulation of long-term user interest unit is shown in the following Equation 6.

$$\text{LTI}(\boldsymbol{E}_t, \boldsymbol{E}_s) = \text{TA}(\boldsymbol{E}_t, \boldsymbol{E}_s^{'}), \tag{6}$$

Table 2: Comparison between different retrieval algorithms. d is the dimension of the embedding vector. L and B are the number of behavior items and target items respectively. m is the dimension of fingerprint generated by SimHash. M is the size of attribute inverted index for each user. In real world CTR model, $L = 1024, B = 1024, d = 128, m = 4, M = 300$. It is not worthy that directly conduct inner product online violate the time constraint and can not be deployed in large scale online RS.

Retrieval input.	Retrieval method.	Gap of goal between retrieval and CTR model.	Retrieval complexity.	Representative.
Attribute.	Offline inverted index	Big	$O(B * log(M))$	SIM(hard) [21] and UBR4CTR [22]
Pre-trained embedding	Offline inverted index and Euclidean distance	Medium	$O(B * M * d)$	SIM(soft) [21]
SimHash-based fingerprint.	**Hamming distance**	**Small**	$O(B * L * m)$	**ETA**
Embedding of CTR model.	Inner product	No	$O(B * L * d)$	Can not deploy.

$$\boldsymbol{E}_s^{'} = \begin{bmatrix} \boldsymbol{e}_1^\top \\ \boldsymbol{e}_i^\top \\ \vdots \\ \boldsymbol{e}_k^\top \end{bmatrix}, \tag{7}$$

$$\boldsymbol{e}_i \in topk(HammingDistance(\boldsymbol{h}_i, \boldsymbol{h}_t)), \tag{8}$$

$$\boldsymbol{h}_i = SimHash(\boldsymbol{e}_i), \boldsymbol{h}_t = SimHash(\boldsymbol{e}_t), \tag{9}$$

where LTI and TA is short for long-term user interest extraction unit and multi-head target attention respectively. $\boldsymbol{E}_s \in \mathbb{R}^{|\mathcal{H}_{lu}| \times d}$ is the embedding matrix of long-term user behavior sequence $\mathcal{H}_{lu}$. $\boldsymbol{E}_s^{'} \in \mathbb{R}^{k \times d}$ consists of top k rows selected from $\boldsymbol{E}_{s_1}$ which have the largest hamming distances with target item $\boldsymbol{E}_t \in \mathbb{R}^{1 \times d}$.

Similarity Function. As shown in Figure 2, we use SimHash function and hamming distance to calculate the similarity of two embedding vectors instead of inner product. SimHash function takes the output of aforementioned embedding layer as input. For each input embedding vector, the SimHash function generates its compressed number as the fingerprint. SimHash satisfies the *locality sensitive* properties: the hashing output is similar if the input features are similar to each other. Thus, the similarity between the embedding vectors can be replaced by the similarity between the hashed fingerprints. A d-dimensional embedding vector can be encoded into m-bit number. Then the similarity between two fingerprints can be measured by hamming distance.

Top-K Retrieval. The top-k retrieval layer can find the top-k similar user behavior items with target item more efficiently by hamming distance compared with the inner-product based models. The hamming distance for two integers is defined as the number of different bits positions at which the corresponding bits are different. To get the hamming distance of two m-bit numbers, we first conduct XOR and then count the number of bits in 1. If we define the multiplication as the atomic operation, then the complexity of hamming distance for two m-digit numbers is $O(1)$). The total complexity of the hamming distance based top-k retrieval is $O(L * B * 1)$, where L is the sequence length, B is the number of candidate items. It is noteworthy that the hashed fingerprints can be stored in an embedding table with the model each time the SimHash function is conducted. During the inference time, only embedding lookup is needed and its complexity is negligible.

4.5 Deployment

In this section, we show how ETA is trained with the retrieval part. Then we introduce how to select the "hash function" used in SimHash algorithm. Then we introduce the engineer optimization tricks.

4.5.1 Joint learning of retrieval part. During the training stage, the retrieval part does not need updates of gradients. The goal of retrieval is to select the nearest neighbors of the keys to query for the following multi-head target attention part. After the top-k closest keys to query are selected, the normal attention and back propagation are conducted on the original embedding vectors of these top-k items. The only thing for retrieval is to initialize a fix random matrix $\boldsymbol{H} \in \mathbb{R}^{d \times m}$ (see Algorithm 1) at the beginning of training. As long as the input embedding vector $\boldsymbol{e}_k \in \mathbb{R}^{1 \times d}$ is updated, the signature of SimHash is updated correspondingly. The *Locality-sensitive* properties ensure that the top-k nearest keys to query in each iteration are selected using the latest embedding of CTR model seamlessly. Thus the gap of goal between retrieval and CTR model is much smaller than those other retrieval methods, *e.g.,* offline inverted index based method shown in Table 2. From the perspective of the CTR model, the retrieval part is transparent but can ensure the model use the most closet items to conduct multi-head attention. The evaluation section (Sec. 5) shows that this end-to-end training without any pre-training or offline inverted index building can greatly improve the performance of CTR prediction task.

4.5.2 Selection of "Hash Function". The SimHash is a well-known locality-sensitive hashing (LSH) [1] algorithm. The implementation of SimHash is shown in Algorithm 1 where we use fix random hash vector to act as "hash function". Any traditional hashing functions which hash the string to a random integer can also be used. However, in our algorithm we choose random hash vector and the implementation of Algorithm 1 for the consideration of scalability and efficiency for matrix computation, which is the same with Reformer [13]. The locality-sensitive hashing is implemented by random rotation and projection. The random rotation refers to the multiplication between the embedding vector and a fix random hashing vector $\boldsymbol{H}(j)$. Any random d-dimensional vector can be used here. It is noteworthy that as we need to project the result of inner product into two singed axes to get binary signature, the element in $\boldsymbol{H}(j)$ should be generated randomly around 0.

Table 3: Corpus sizes of dataset used in this paper.

	Users	Items	Categories	Instances
Taobao.	987,994	4,162,024	9,439	100,150,807
Industrial(Our Own)	0.4 billion	0.7 billion	24,568	142 billion

4.5.3 Engineered Optimization Tricks. When the model is deployed online, the calculation of SimHash can be reduced one step further. For a m-length signature vector $\boldsymbol{sig}_k$ for embedding vector $\boldsymbol{e}_k$ calculated by Algorithm 1, we can use $log(m)$-bits integer to represent the signature vector because each element in $\boldsymbol{sig}_k$ is either 1 or 0. This can greatly reduce the cost of memory and can speed up the calculation of hamming distance. The calculation time of two integers can be conducted in $O(1)$ time complexity and can be neglected.

5 EXPERIMENTS

In this section, we conduct experiments to answer the following research questions.

- **RQ1**: Does our ETA model outperform the baseline models?
- **RQ2**: What is the inference time of our ETA model compared with the baseline models? Inference time is as important as performance because it decides whether the model can be deployed online for serving.
- **RQ3**: Which part of our ETA model contributes the most to the performance and inference time?

Before presenting the evaluation results, we first describe the datasets, baseline models, metrics and experimental settings.

5.1 Datasets

To conduct comprehensive comparisons between our ETA model and the baseline models, both public dataset and industrial dataset are used. Online A/B test is also conducted. For public dataset, we choose Taobao dataset, which is also adopted by baseline models SIM [21] and UBR4CTR [22]. An industrial dataset is prepared as the supplement for public dataset. Table 3 gives a brief introduction of the datasets.

Taobao Dataset [1]: This dataset is first released by [38] and is widely used as the public benchmark for CTR prediction task and sequential recommendation task. It is made up of users' behavior logs from Taobao Mobile App. The user behaviors include click, favorite, add to cart and buy. This dataset contains 100 million instances. In average, each user has about 101 interactions and each item receives over 24 interactions. The recent 16 behaviors are selected as short-term user behavior sequence and the recent 256 behaviors are selected as long-term user behavior sequence.

Industrial Dataset [2]: This dataset is collected from our own online RS, which is one of the top-tier mobile Apps in our country. There are three advantages for our industrial dataset. (i) Our dataset contains impression interaction, which indicates an item is displayed to an user but not clicked by the user. Impression interaction is naturally the negative sample of CTR model. As a result, the tricky negative sampling is not needed. (ii) The user behavior sequence is much longer in our industrial dataset. There are over 142 billion instances and the average length reaches 938, which is 9 times longer than the public Taobao dataset. (iii) Our industrial dataset has more features designed by multiple software engineers which is closer to the real-world RS models. The recent 48 behaviors are selected as short-term user behavior sequence and the recent 1024 behaviors are selected as long-term user behavior sequence. In the ablation study, we also try the long-term user behavior sequence with the lengths in $\{256, 512, 2048\}$.

[1] https://tianchi.aliyun.com/dataset/dataDetail?dataId=649&userId=1

[2] This dataset will be released to public to help the research on long-term user interest modeling.

5.2 Baselines and Metrics

Baselines: We compare our model with the following mainstream baselines for CTR prediction. Each baseline is chosen to answer one or more related research questions mentioned above.

- **Avg-Pooling DNN**: The simplest way to utilize user behavior sequence is average pooling which encodes the various length of user sequences into fixed-size hidden vector. This baseline can be regarded as variant of DIN by replacing the target attention with average pooling, which is similar to YouTube [6]. This baseline is mainly used to show the necessity of target attention when compared with DIN.
- **DIN** [36]: DIN is proposed to model personalized user interests with different target items by an attention mechanism, which is called as target attention. However, DIN only utilizes the short-term user behavior sequence.
- **DIN (Long Sequence)** is DIN equipped with long-term user behavior sequence $\mathcal{H}_{lu}$. $\mathcal{H}_{lu}$ is encoded by mean pooling. This baseline is used to measure the information gain of long-term user behavior sequence itself when compared with DIN.
- **SIM(hard)** [21]: SIM is the CTR prediction model which propose a search unit to extract user interest from long-term user behavior sequence in a two stage manner. SIM(hard) is SIM which searches the top-k behavior items by category id in the first stage.
- **UBR4CTR** [22]: UBR4CTR is also a two-stage method which utilizes the long-term user behavior sequence in CTR prediction task. In UBR4CTR, a query is prepared by a feature selection model to retrieve the most similar behavior items. An inverted index is prepared for online usage. As UBR4CTR and SIM are published almost simultaneously, they do not compare to each other. In our paper, we compare both UBR4CTR and SIM for the first time.
- **SIM(hard)/UBR4CTR + timeinfo** is SIM(hard)/UBR4CTR with time embedding when encoding the user behavior sequence.

In [21], the authors propose SIM(soft) as the variants of base algorithm SIM(hard). They finally adopt SIM(hard) method as their online serving algorithm and deploy SIM(hard)+timeinfo online to serve the main traffic. This is because SIM(hard) does not need pretraining and is more friendly to system evolution and maintenance. Besides, SIM(hard) + timeinfo can achieve comparable performance with SIM(soft). Thus, we choose SIM(hard) and SIM(hard) + timeinfo as our strong baselines.

MIMN [20] is proposed by the same team with DIN. A multi-track offline user interest center is proposed by MIMN to extract the

long-term user interest. At the publishing time, it achieves the state-of-the-art performance by leverage the long-term user behavior sequence. However, MIMN is defeated by SIM [21] from the same team. As MIMN contributes little to our research questions, we omit this baseline for the space limitation.

Metrics: For offline experiments, we adopt widely used *area under ROC curve* (AUC) as our main metric and *Inference Time* as a supplement metric. AUC is suitable for binary classification problem for measuring pairwise ranking performance. *Inference Time* is defined as the round-trip time when scoring a batch of items for a certain model request. We measure the *Inference Time* by deploying models online to serve user requests which are copied from product environment. The machines and the number of user requests are controlled same for fairness comparison.

For online A/B test, we use CLICK and CTR as evaluation metrics. CLICK is defined as the total number of clicked items. CTR is used to measure the willingness of click for users in the platform. It is defined as $CLICK/PV$ where PV is defined as the total number of displayed items.

5.3 Experimental Setup

In this section, we first introduce the pre-processing for offline datasets. Then we list the hyper-parameters of both baselines and our models.

Taobao Dataset only contains positive interactions, *e.g.*, review, click, favorite, add to cart and buy. We use the same data pre-processing method with MIMN [20]. Firstly, for each user, we select the last behavior as the positive sample. Then we randomly sample a new item with same category as the negative sample for this user. The rest behavior items are used as features. The samples are split into training set (80%), validation set (10%) and test set (10%) according to the timestamp t of this sample.

Our Industrial dataset has positive and negative samples naturally because we log all the impressions for each user. An impression is labeled as positive if the item is clicked by the user. Otherwise, it is labeled as negative sample. We use the past two weeks' logs as training set and the following day as the test set, which is similar with SIM [21].

For each model on different datasets, we use the validation set to tune the hyper-parameters to get the best performance. The learning rate is searched from 1×10^{-4} to 1×10^{-2}. The L2 regularization term is searched from 1×10^{-4} to 1. All the models use Adam optimizer. The batch size is 256, 1024 for Taobao and our industrial dataset respectively.

5.4 Performance Comparison

Taobao Dataset: The evaluation results on Taobao dataset are shown in Table 4. From the table, we find that our ETA has stable performance improvements compared with all baselines. ETA outperforms SIM(hard) by 0.46% and outperforms DIN(Long Sequence) by 0.6%. After adding time embedding, ETA+timeinfo outperforms SIM(hard)+timeinfo by 0.38% and outperforms DIN(Long Sequence) by 0.85%. Similar results can be observed on SIM(hard) and UBR4CTR. It is observed that DIN(Long Sequence) brings 0.35% improvement on AUC compared to DIN, which shows the effectiveness of modeling the long-term user behavior sequence for CTR prediction. We also find that UBR4CTR performs worse than DIN(Long Sequence). It is because that the feature selection model of UBR4CTR only selects the behaviors whose features(eg. category, weekday) are same with the target item. This kind of filtering in UBR4CTR helps removing noises away from the sequence, but can also lead to shorter user sequence, which is harmful when there are not enough items for top-k retrieval. From Table 4, we find that DIN outperforms Avg-Pooling DNN by 1.84%, demonstrating the fact that using target attention to encode the user sequence can greatly improve the performance.

Industrial Dataset: The evaluation results on our own Industrial Dataset are shown in Table 5. Note that 0.1% AUC improvement of CTR model can bring millions of real-world clicks and revenue in our online RS. Our ETA achieves the best performance compared with all baselines. Our base ETA achieves 0.34% and 0.43% improvements compared with SIM(hard) and UBR4CTR respectively. Our ETA+timeinfo achieves 0.35% and 0.42% improvements

Table 4: Experimental results on Taobao Dataset.

Method	AUC
Avg-Pooling DNN	0.8442
DIN	0.8626
DIN (Long Sequence)	0.8661
UBR4CTR	0.8651
UBR4CTR+timeinfo	0.8683
SIM(hard)	0.8675
SIM(hard)+timeinfo	0.8708
ETA	0.8721
ETA+timeinfo	0.8746

Table 5: Experimental results on Industrial Dataset.

Method	AUC	Inference Time(ms)
Avg-Pooling DNN	0.7216	8
DIN	0.7279	11
DIN (Long Sequence)	0.7311	14
UBR4CTR	0.7318	41
UBR4CTR+timeinfo	0.7331	41
SIM(hard)	0.7327	21
SIM(hard)+timeinfo	0.7338	21
ETA	0.7361	19
ETA+timeinfo	0.7373	19

Table 6: Relative performance improvements and Inference Time in online A/B test compared with a DIN-based method without long-term user behavior sequence. Note that 1% improvement on GMV is a significant improvement because it means that millions of more revenues are brought to the recommender system.

Method	CTR	GMV	Inf. Time
SIM(hard)+timeinfo	4.53%	6.6%	21ms
ETA+timeinfo	6.33%	9.7%	19ms

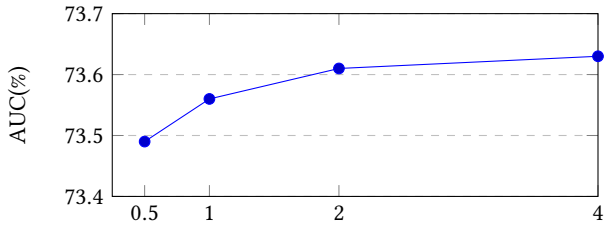

Figure 3: AUC of ETA with different bit-lengths of hashed finger. The other model settings are the same with ETA in Table 5.

compared with SIM(hard)+timeinfo and UBR4CTR+timinfo respectively. Different from the experimental results in public dataset, SIM(hard)+timeinfo becomes the strongest baseline on Industrial Dataset and outperforms DIN(Long Sequence) by 0.27%. This is because of two reasons. On one hand, the user sequence length in Industrial Dataset is large enough, which is friendly to long-term user sequence based models. The average length of Industrial Dataset reaches 938, which is 9 times longer than the public Taobao dataset. On the other hand, DNN (Long Sequence) uses average pooling to encode the whole sequence without selection, which may import noises compared with retrieval based models such as UBR4CTR, SIM and ETA.

We can find another fact that SIM(hard) has 0.09% performance improvement compared with UBR4CTR. This is mainly caused by different processing methods on user behavior sequence. In SIM(hard), the user sequence is split into two separate sub-sequences, which is similar with our ETA in Figure 2. The short-term user behavior sequence $\mathcal{H}_{su}$ is made up of recent k user behaviors from item 1 to item k. The long-term user behavior sequence $\mathcal{H}_{lu}$ is made up of another k behaviors selected from item $k+1$ to item n. However, UBR4CTR selects a $2 * k$-length behavior sequence from item 1 to item n. As a result, the most recent k items ($\boldsymbol{e}_1$ to $\boldsymbol{e}_k$ in Figure 2) are selected by SIM(hard) in 100% probability and are selected by UBR4CTR in p probability decided by the feature selection model. However, timeinfo plays an important role in user interest modeling because user interest is dynamic and changes frequently. Thus SIM(hard) performs betther than UBR4CTR.

Online A/B Test: The evaluation results of online A/B test is shown in Table 6. Table 6 shows the performance improvements over a DIN-based method, where the DIN-based method does not have long-term user behavior sequence. From Table 6, we find that our ETA+timestamp achieves 6.33% improvements on CTR and brings 9.7% extra GMV compared with the DIN-based method. Compared with the strongest baseline SIM(hard)+timeinfo, our ETA+timeinfo has extra 1.8% improvements on CTR and 3.1% improvements on GMV. Note that 1% improvement on GMV is a significant improvement because it means that millions of more revenues are brought to the recommender system.

Table 7: Ablation study of our ETA model on Industrial Dataset. v0 is the base version of ETA. avg(·) and ta(·) represent encoding user behaviors by average pooling and target attention respectively. ta(1024 -s- 48) represents conducting target attention on top-48 user behaviors selected from 1024 sequential user behavior items. Symbol s in ta(1024 -s- 48) represents SimHash is used to select top-48 from 1024. Similarly, symbol i in ta(1024 -i- 48) represents inner product is used to select top-48 from 1024.

ETA Version	Encoding Manner	AUC	Inference Time(ms)
v0	ta(1024 -s- 48)	0.7361	19
v1	avg(1024)	0.7311	14
v2.1	ta(256 -s- 48)	0.7339	14
v2.2	ta(512 -s- 48)	0.7348	16
v2.3	ta(2048 -s- 48)	0.7394	23
v3	ta(1024 -i- 48)	0.7368	32
v4	ta(1024)	0.7371	35

5.5 Inference Time Comparison

Though the performance of CTR prediction is improved using long-term user behavior sequence, the model complexity increases accordingly. We measure the inference time of different models which are shown in Table 4. Avg-Pooling DNN has the smallest inference time of 8 milliseconds (ms). It only encodes the recent behavior items with the average pooling method. After replacing average pooling to target attention, the inference time increases by 3ms (8ms to 11ms). After importing the long-term user behavior sequence, the inference time increases by another 3ms (11ms to 14ms) . SIM and our ETA have comparable inference time around 19∼21 ms. UBR4CTR has the largest inference time because an extra feature selection model is used before the retrieval stage, and an relative time-consuming IDF and BM25 based procedure is conducted online to get top-k items.

5.6 Ablation Study

The results of ablation study is shown in Table 7 to answer the research question RQ3. We use encoding manner to distinguish different versions (v0 to v4) of our ETA model. Note that **v0** is the base version of ETA. The encoding manners are listed in the second column of Table 7, where avg(·) and ta(·) represent encoding user behaviors by average pooling and target attention respectively. ta(1024 -s- 48) represents conducting target attention on top-48 user behaviors selected from 1024 sequential user behavior items. Symbol **s** in ta(1024 -**s**- 48) represents SimHash is used to select top-48 from 1024. Similarly, symbol **i** in ta(1024 -**i**- 48) represents inner product is used to select top-48 from 1024.

From Table 7, the following observations are made. (i) Directly conduct multi-head target attention on the original 1024-length user sequence (v4) can achieve the best performance but have highest inference time at the same time. Comparing with v4, our base ETA (v0) selecting the top-k behaviors for attention sacrifices about 0.1% AUC and reduces the inference time by 46%. (ii) Comparing v3 with

v0, replacing the SimHash with inner product at the retrieval stage achieves 0.07% improvement on AUC. However, the inference time increases by 68%, which is not acceptable with our strict online SLA (service level agreement). (iii) Trade-offs between AUC and inference time are observed when we change the length of user behavior sequence (v2.x versus v0). The appropriate sequence length can be decided according to the requirements of online inference time. We also evaluate the performance under different bit-lengths of hashed fingerprints $\boldsymbol{h}$ generated by SimHash in Figure 3. As mentioned in Section 3.2, the bit-length of fingerprint can be controlled by the number of hash functions used in SimHash. We can find the fact that AUC can be improved by increasing the bit-length of $\boldsymbol{h}$. However, when bit-length of $\boldsymbol{h}$ is larger than 2× embedding size, the improvement on AUC becomes marginal.

6 CONCLUSION

In this paper, we propose ETA model for CTR prediction task. To the best of our knowledge, ETA is the first method which can model the CTR together with long-term user behavior sequence in an end-to-end way. Compared to the SOTA two-stage models, the end-to-end paradigm enables the retrieval part share the information with the main part of CTR model seamlessly, which improves the prediction performance significantly. Besides, it is friendly for the maintenance and evolution of CTR model in large-scale online RS. To achieve the goal of end-to-end online retrieval, we propose a SimHash based method to reduce the complexity of traditional top-k retrieval from $O(L * B * d)$ multiplication to $O(L * B)$ hamming distance calculation, where L is the length of user sequence. B is the number of candidate items for each user request. d is the dimension of item embedding. Both offline and online experiments confirm the effectiveness of our ETA. The total GMV are improved by 3.1% in online A/B test compared with the SOTA model. ETA has been deployed online to serve the mainstream traffic.

REFERENCES

[1] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. 2015. Practical and optimal LSH for angular distance. *arXiv preprint arXiv:1509.02897* (2015).

[2] Moses S Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. 380–388.

[3] 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.

[4] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[6] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[7] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep session interest network for click-through rate prediction. *arXiv preprint arXiv:1905.06482* (2019).

[8] 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).

[9] 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 *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*. 1–9.

[10] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[11] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[12] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[13] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451* (2020).

[14] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation* 1, 4 (1989), 541–551.

[15] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-interest network with dynamic routing for recommendation at Tmall. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2615–2623.

[16] Zeyu Li, Wei Cheng, Yang Chen, Haifeng Chen, and Wei Wang. 2020. Interpretable Click-Through Rate Prediction through Hierarchical Attention. In *Proceedings of the Thirteenth ACM International Conference on Web Search and Data Mining*. ACM.

[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] Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. 2007. Detecting near-duplicates for web crawling. In *Proceedings of the 16th international conference on World Wide Web*. 141–150.

[19] Junwei Pan, Jian Xu, Alfonso Lobos Ruiz, Wenliang Zhao, Shengjun Pan, Yu Sun, and Quan Lu. 2018. Field-weighted factorization machines for click-through rate prediction in display advertising. In *Proceedings of the 2018 World Wide Web Conference*. 1349–1357.

[20] 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.

[21] Pi Qi, Xiaoqiang Zhu, Guorui Zhou, Yujing Zhang, Zhe Wang, Lejian Ren, Ying Fan, and Kun Gai. 2020. Search-based User Interest Modeling with Lifelong Sequential Behavior Data for Click-Through Rate Prediction. *arXiv preprint arXiv:2006.05639* (2020).

[22] Jiarui Qin, Weinan Zhang, Xin Wu, Jiarui Jin, Yuchen Fang, and Yong Yu. 2020. User Behavior Retrieval for Click-Through Rate Prediction. *arXiv preprint arXiv:2005.14171* (2020).

[23] 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 *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1149–1154.

[24] 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. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 565–574.

[25] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.

[26] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1441–1450.

[27] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 565–573.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR* abs/1706.03762 (2017). arXiv:1706.03762 http://arxiv.org/abs/1706.03762

[29] 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.

[30] 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. *arXiv preprint arXiv:1708.04617* (2017).

[31] Weinan Xu, Hengxu He, Minshi Tan, Yunming Li, Jun Lang, and Dongbai Guo. 2020. Deep Interest with Hierarchical Attention Network for Click-Through Rate Prediction. *arXiv preprint arXiv:2005.12981* (2020).

[32] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A simple convolutional generative network for next item recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 582–590.

[33] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep learning over multi-field categorical data. In *European conference on information retrieval*. Springer, 45–57.

[34] Guorui Zhou, Weijie Bian, Kailun Wu, Lejian Ren, Qi Pi, Yujing Zhang, Can Xiao, Xiang-Rong Sheng, Na Mou, Xinchen Luo, et al. 2020. CAN: Revisiting Feature Co-Action for Click-Through Rate Prediction. *arXiv preprint arXiv:2011.05625* (2020).

[35] 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.

[36] 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 *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.

[37] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2020. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *arXiv preprint arXiv:2012.07436* (2020).

[38] Han Zhu, Daqing Chang, Ziru Xu, Pengye Zhang, Xiang Li, Jie He, Han Li, Jian Xu, and Kun Gai. 2019. Joint optimization of tree-based index and deep model for recommender systems. *arXiv preprint arXiv:1902.07565* (2019).