Sampling Is All You Need on Modeling Long-Term User Behaviors for CTR Prediction

Yue Cao
Meituan Inc.
Beijing, P.R. China
yuecao@pku.edu.cn

Xiaojiang Zhou
Meituan Inc.
Beijing, P.R. China
zhouxiaojiang@meituan.com

Jiaqi Feng
Meituan Inc.
Beijing, P.R. China
fengjiaqi03@meituan.com

Peihao Huang
Meituan Inc.
Beijing, P.R. China
huangpeihao@meituan.com

Yao Xiao
Meituan Inc.
Beijing, P.R. China
xiaoyao06@meituan.com

Dayao Chen
Meituan Inc.
Beijing, P.R. China
chendayao@meituan.com

Sheng Chen
Meituan Inc.
Beijing, P.R. China
chensheng19@meituan.com

ABSTRACT

Rich user behavior data has been proven to be of great value for Click-Through Rate (CTR) prediction applications, especially in industrial recommender, search, or advertising systems. However, it's non-trivial for real-world systems to make full use of long-term user behaviors due to the strict requirements of online serving time. Most previous works adopt the retrieval-based strategy, where a small number of user behaviors are retrieved first for subsequent attention. However, the retrieval-based methods are sub-optimal and would cause information losses, and it's difficult to balance the effectiveness and efficiency of the retrieval algorithm.

In this paper, we propose **SDIM** (**S**ampling-based **D**eep **I**nterest **M**odeling), a simple yet effective sampling-based end-to-end approach for modeling long-term user behaviors. We sample from multiple hash functions to generate hash signatures of the candidate item and each item in the user behavior sequence, and obtain the user interest by directly gathering behavior items associated with the candidate item with the same hash signature. We show theoretically and experimentally that the proposed method performs on par with standard attention-based models on modeling long-term user behaviors, while being sizable times faster. We also introduce the deployment of SDIM in our system. Specifically, we decouple the behavior sequence hashing, which is the most time-consuming part, from the CTR model by designing a separate module named BSE (Behavior Sequence Encoding). BSE is latency-free for the CTR server, enabling us to model extremely long user behaviors. Both offline and online experiments are conducted to demonstrate the effectiveness of SDIM. SDIM now has been deployed online in the search system of Meituan APP.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557082

CCS CONCEPTS

• **Information systems** → **Personalization**; **Recommender systems**.

KEYWORDS

CTR prediction, Hash-based sampling, Long-term user behavior modeling

ACM Reference Format:
Yue Cao, Xiaojiang Zhou, Jiaqi Feng, Peihao Huang, Yao Xiao, Dayao Chen, and Sheng Chen. 2022. Sampling Is All You Need on Modeling Long-Term User Behaviors for CTR Prediction. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557082

1 INTRODUCTION

Click-Through Rate (CTR) prediction is an essential task in industrial applications systems. User interest modeling, which aims at learning user's implicit interest from historical behavior data, has been widely introduced for real-world systems and contributes remarkable improvement [4, 14, 18].

Various models are proposed for modeling users' interests [5, 14, 18]. Among them, DIN [18] adaptively calculates user interests by taking the relevance of historical behaviors given a target item into consideration. DIN introduces a new attention module named **target attention**, where the target item acts as the query **Q** and historical user behaviors act as the key **K** and the value **V**. Due to its superior performance, DIN-based methods have become the mainstream solution for modeling user interests in recent years. However, the strict requirements of online serving time limit the length of user behavior sequences that can be used. As a result, most of the industrial systems truncate the user behavior sequences and just feed recent 50 behaviors [17, 18] for user interest modeling, which leads to information losses.

With the rapid development of the internet, user accumulates more and more behavior data on E-commerce platforms. Take Taobao for example, they report that 23% of users have more than 1,000 behaviors in Taobao APP in six months [13]. In Meituan APP, there are more than 60% of users have at least 1,000 behaviors, and more than 10% of users have at least 5,000 behaviors in one year. How to effectively utilize more user behaviors for a more accurate user interest estimation becomes more and more important for industrial systems.

Recently, some methods are proposed to model users' long-term interests from long behavior sequences [3, 10–12]. MIMN [10] decouples the user interest modeling from the entire model by designing a separate User Interest Center (UIC) module. Although UIC can save lots of online serving time, it makes it impossible for the CTR model to exploit the information from the target item, which has been proved to be crucial for users interest modeling [3]. As a consequence, MIMN can only model shallow user interests. SIM [11] and UBR4CTR [12] adopt two-stage frameworks for handling long-term user behaviors. They retrieves top-k similar items from the sequence, and then feed these items to the subsequent attention module [18]. As pointed by [3], the retrieve objectives of these approaches are divergent with the goal of the CTR model, and the pre-trained embedding in offline inverted index is not suiting for online learning systems. To improve the quality of the retrieve algorithm, ETA [3] proposes an LSH-based method to retrieve top-k similar items from user behaviors in an end-to-end fashion. They use locality-sensitive hash (LSH) to convert items into hash signatures, and then retrieve top-k similar items based on their hamming distance to the candidate item. The use of LSH greatly reduces the cost to calculate the similarity between items, and ETA achieves better results than SIM [11] and UBR4CTR [12].

SIM, UBR4CTR, and ETA are all retrieval-based approaches. The retried-based methods suffer from following drawbacks: Retrieving top-k items from the whole sequence is sub-optimal and would produce biased estimation of user's long-term interests. In the case that a user has rich behaviors, the retrieved top-k items may all similar to the candidate item and the estimated user interest representation would be inaccurate. Besides, it's difficult to balance to effectiveness and efficiency of the retrieval algorithm. Take SIM (hard) as an example, they use a simple retrieve algorithm, therefore its performance is inferior compared with other methods. In contrast, UBR4CTR achieves significant improvement with the aid of a complex retrieval module, but its inference speed becomes 4× slower [12], which prevents UBR4CTR from being deployed online, especially for long-term user behaviors modeling.

In this paper, we propose a simple hash sampling-based approach for modeling long-term user behaviors. First, we sample from multiple hash functions to generate hash signatures of the target item and each item in user behavior sequence. Instead of retrieving top-k similar items using a certain metric, we directly gather behavior items that share the same hash signature with the target item from the whole sequence to form the user interest. The intrinsic idea behind our method is to approximate the softmax distribution of user interest with LSH collision probability. *We show theoretically that this simple sampling-based method produces very similar attention patterns as softmax-based target attention and achieves consistent model performance, while being much more time-efficient.* As a consequence, our method behaviors like computing attention directly on the original long sequence, without information loss. We named the proposed method as **SDIM** (means **S**ampling-based **D**eep **I**nterest **M**odeling).

We also introduce our hands-on practice on deploying SDIM online. Specifically, we decouple our framework into two parts: (1) Behavior Sequence Hashing (BSE) server, and (2) CTR server, where the two parts are deployed separately. The behavior sequence hashing is the most time-consuming part of the whole algorithm, and the decoupling of this part greatly reduces the serving time. More details will be introduced in Section 4.4.

We conduct experiments on both public and industrial datasets. Experimental results show that SDIM achieves results consistent with standard attention-based methods, and outperforms all competitive baselines on modeling long-term user behaviors, and with sizable speed ups. SDIM has been deployed in the search system in Meituan[1], the largest e-commerce platform for lifestyle services in China, and bringing 2.98% CTR and 2.69% VBR lift, which is very significant to our business.

2 RELATED WORK

2.1 Click-Through Rate (CTR) Prediction

With the rapid development of deep learning, deep neural-based methods have achieved remarkable performance in CTR prediction.

User interest modeling is the key problem for CTR prediction. Many work [4, 8, 17, 18] focus on learning better representation from user historical behaviors in recent years. DIN [18] introduces target attention which learns different user interests with regard to different target items. In [17], considering interest evolving process in user behaviors, DIEN proposes an interest evolving layer to capture dynamic interest about target item. In DSIN [4], based on a prior knowledge that user behaviors are highly homogeneous in each session and heterogeneous in different sessions, the session-based model is proposed. It is worth noting that the target attention mechanism in DIN [18] has been widely used in CTR models nowadays.

2.2 Long-Term Sequential User Behavior Modeling

User behavior modeling has shown great performance in industrial applications. Exploring richer user behavior in CTR models has attracted much attention. However, long-term user behavior modeling is faced with several challenges, such as complex model deployment and system latency.

In MIMN [10], the user interest center (UIC) block with a memory-based architecture design is proposed to tackle the challenge. The block is updated by user behavior event and only need to store limited user interest memory. MIMN is hard to model the interaction between user behavior sequence and target item which has proved to be important in CTR modeling. In SIM [11], a two-stage method is proposed to model long-term user behavior sequences. Firstly, a general search unit (GSU) is utilized to extract relevant behaviors with regard to the target item. Secondly, an exact search unit is

[1] https://meituan.com/

proposed to model relevant behaviors in an end-to-end manner. UBR4CTR [12] uses a similar search-based method as SIM [11] to face challenges. Recently, ETA [3] proposes an end-to-end target attention method to model long-term user behavior sequence. They apply locality-sensitive hashing (LSH) to reduce the training and inference time cost.

Besides the above related works in the field of CTR prediction, there are also plenty of works in Natural Language Processing (NLP) that aim to improve the efficiency of self-attention [6, 15, 16, 19]. These approaches can reduce the time complexity of self-attention from $O(L^2)$ to $O(L\log(L))$, where L is the sequence length, but can not be applied to reduce the $O(L)$ time complexity of the target attention.

3 PRELIMINARIES

3.1 Task Formulation

CTR prediction is a core task in the industrial recommendation, search, and advertising systems. The goal of the CTR task is to estimate the probability of a user clicking on the item, which is defined as follows:

$$\text{prob}_i = P(y_i = 1|x_i;\theta) \tag{1}$$

In above equation, θ represents trainable parameters in CTR model. Given input features x_i, CTR model is trained to predict the click probability prob_i by minimizing the cross-entropy loss:

$$L = -\frac{1}{N}\sum_{i=1}^{N}(y_i\log(\text{prob}_i) + (1-y_i)\log(1-\text{prob}_i)) \tag{2}$$

3.2 Target Attention

The concept of target attention is first proposed by DIN [18] and is widely applied for modeling user interests on CTR tasks [3, 4, 10, 11, 17]. Target attention takes the target item as query and each item in the user behavior sequence as key and value, and uses an attention operator to soft-search relevant parts from the user behavior sequence. The user interests are then obtained by taking a weighted sum over user behavior sequence.

Specifically, designate the target item as $\mathbf{Q} \in \mathbb{R}^{B\times d}$ and user sequence representations as $\mathbf{S} \in \mathbb{R}^{B\times L\times d}$, where B is the number of candidate items to be scored by the CTR model for each request, L is the length of user behavior sequence, and d is the model's hidden size. Let $\mathbf{q}_i$ be the i-th target item, target attention calculates the dot product similarity between $\mathbf{q}_i$ and each item in behavior sequence $\mathbf{S}$, and then uses the normalized similarities as the weights to obtain user's interest representation.

$$\text{TA}(\mathbf{q}_i, \mathbf{S}) = \frac{\exp(\mathbf{q}_i^\top \mathbf{s}_j/t)}{\sum_{j=1}^{L}\exp(\mathbf{q}_i^\top \mathbf{s}_j/t)}\cdot \mathbf{s}_j \tag{3}$$

The matrix form of Eq. 3 can be written as:

$$\text{TA}(\mathbf{Q}, \mathbf{S}) = softmax(\frac{\mathbf{Q}^\top \mathbf{S}}{t})\mathbf{S} \tag{4}$$

The scaling factor t is used to avoid large value of the inner product [3].

The complexity of calculating $\text{TA}(\mathbf{Q}, \mathbf{S})$ is $O(BLd)$ where B is the number of candidate items for each request, L is the length of user behavior sequence, and d is the model's hidden size. In large-scale industrial systems, B is about 10^3 and d is about 10^2, and **it's infeasible to deploy target attention on long-term user behaviors modeling for online systems** [3, 10].

3.3 Locality-Sensitive Hash (LSH) and SimHash

Locality-sensitive hash (LSH) [1] is an algorithmic technique for finding nearest neighbors efficiently in high-dimensional spaces. LSH satisfies the locality-preserving property: nearby vectors get the same hash with high probability while distant ones do not. Benefiting from this property, LSH-based signatures has been widely used in many applications such as web search system [9]. The **random projection scheme (SimHash)** [2, 7] is an efficient implementation of LSH. The basic idea of SimHash is to sample a random projection (defined by a normal unit vector $\mathbf{r}$) as hash function to hash input vectors into two axes (+1 or -1). Specifically, given an input $\mathbf{x}$ and a hash function $\mathbf{r}$, the corresponding hash is calculated as:

$$h(\mathbf{x}, \mathbf{r}) = \text{sign}(\mathbf{r}^\top \mathbf{x}) \tag{5}$$

where $\mathbf{r} \in \mathbb{R}^d$, $r_i \sim \mathcal{N}(0,1)$. $h(\mathbf{x}, \mathbf{r}) = \pm 1$ depends on which side of the hashed output lies. For two vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, we say $\mathbf{x}_1$ collides with $\mathbf{x}_2$ only when they have the same value of the hash code:

$$p^{(\mathbf{r})} = \mathbb{1}_{h(\mathbf{x}_1,\mathbf{r})=h(\mathbf{x}_2,\mathbf{r})} \tag{6}$$

While a single hash would work in estimating similarities, the outputs can be the average of multiple hashes to lower the estimation error. In practice, the (m,τ)-parameterized SimHash (multi-round hash) algorithm [7] is usually adopted, where m is the number of hash functions and τ is the width parameter. In the first step, SimHash randomly samples m hash functions and generates m hash codes for each input:

$$h(\mathbf{x}, \mathbf{R}) = \text{sign}(\mathbf{R}\mathbf{x}) \tag{7}$$

where $\mathbf{R} \in \mathbb{R}^{m\times d}$, $h(\mathbf{x}, \mathbf{R}) \in \mathbb{R}^m$. To reduce the probability of dissimilar items having the same hash codes, the algorithm merge the results of every τ codes to form a new hash signature. For more details, please refer to [7]. In SimHash, $\mathbf{x}_1$ collides with $\mathbf{x}_2$ only when they have the same value of the hash signature, i.e., their hash codes in a batch of τ codes are all the same:

$$\begin{aligned}\tilde{p}^{(\mathbf{R})} &= p^{(\mathbf{r}_1)} \,\&\, p^{(\mathbf{r}_2)} \,\&\, \cdots \&\, p^{(\mathbf{r}_\tau)} \\ &= \&_{k=1}^{\tau} p^{(\mathbf{r}_k)}\end{aligned} \tag{8}$$

where "&" represents the operator of logical "AND", and $\mathbf{r}_k$ is the k-th hash function. The bottom of Figure 1 shows an example of $(4,2)$-parameterized SimHash algorithm, where we use 4 hash functions and aggregate every 2 hash codes into 2 hash signatures (yellow and green).

4 METHODOLOGY

In this section, we introduce our framework for implementing SDIM in our system. A high-level overview can be seen in Figure 1. The framework is consist of two separate servers: Behavior Sequence Encoding (BSE) server and CTR Server, which will be introduced in detail later.

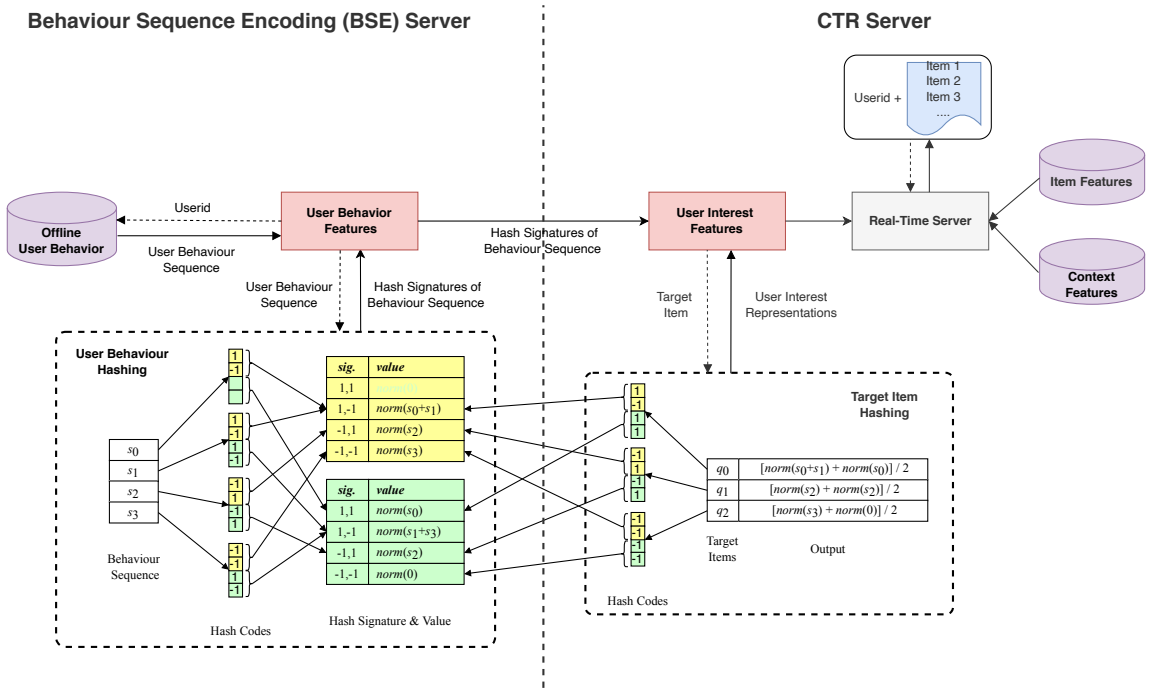

Figure 1: High-level overview of our framework. It consists of two separate servers: Behavior Sequence Encoding (BSE) server and CTR server. The BSE server calculates the hashing of user behavior sequence, and the CTR server calculates the hashing of candidate items and gathers behavior items that share the same hash signature with the candidate item to form user interests. The CTR server contains a real-time CTR model, which takes user interest features, item features, and context features as inputs to predict the click probability corresponding to each candidate item.

4.1 User Behavior Modeling via Hash-Based Sampling

4.1.1 Hash Sampling-Based Attention. At the first step, we sample multiple hash functions and generate the hash codes of user behaviors and candidate items. Similar to ETA [3], we use fixed random hash vectors drawn from normal distribution as "hash functions". After hashing, ETA calculates the Hamming distance between items as an approximation of user interest distribution to select top-k similar items, here we propose a more efficient and more effective way to approximate the user's interest.

With the locality-preserving property, similar vectors fall in the same hash bucket with high probability, therefore the similarity between user behavior items and the candidate item can be approximated by their frequency of having the same hash codes (signatures), or the collision probability. This leads us to hypothesize that **the probability of hash collision can be an effective estimator of user's interests.**

Based on this observation, we propose a new method to obtain the user's interest with LSH. After hashing, we directly form the user interest by summing together behavior items $\mathbf{s}_j$ associated with the candidate item $\mathbf{q}$ with the same signature. In a single hash function $\mathbf{r}$, the proposed method for estimating user interest can be calculated by:

$$\ell_2\left(\mathbf{P}^{(\mathbf{r})}\mathbf{S}\right) = \ell_2\left(\sum_{j=1}^{L} p_j^{(\mathbf{r})}\mathbf{s}_j\right) \tag{9}$$

where $\mathbf{S}$ is the user behavior sequence and $\mathbf{s}_j$ is the j-th item in this sequence, $p_j^{(\mathbf{r})} = \{0, 1\}$ specifies whether $\mathbf{s}_j$ contributes to the user interests. Concretely, if $\mathbf{s}_j$ shares the same hash signature with the candidate item $\mathbf{q}$ under hash function $\mathbf{r}$, then $p_j^{(\mathbf{r})} = 1$, else $p_j^{(\mathbf{r})} = 0$:

$$p_j^{(\mathbf{r})} = \mathbb{1}_{h(\mathbf{q},\mathbf{r})=h(\mathbf{s}_j,\mathbf{r})} \tag{10}$$

where $h(\cdot,\cdot)$ is defined in Eq. 5. $\ell_2(\cdot)$ in Eq. 9 refers to the ℓ_2 normalization, which is used to normalize the interest distribution such that the attention weights sum up to 1.[2]

4.1.2 Multi-Round Hashing. There is always a small probability that dissimilar items share the same hash codes with the candidate item, thus bringing noise to the model. To reduce this probability, we use the multi-round hash algorithm described in Section 3.3.

Specifically, we use the (m, τ)-parameterized SimHash algorithm. We sample and perform m times hashes in parallel, and merge the

[2]We also tried to normalize the distribution using the sum of $P_j^{(\mathbf{r})}$, and the resulting model performs on par with ℓ_2 normalized model. However, the implementation of ℓ_2 normalization will be more efficient, therefore we use the ℓ_2 normalization.

results of every τ hash codes to form a new hash signature. We regard $\mathbf{s}_j$ collides with $\mathbf{q}$ only when they have the same value of aggregated hash signature, i.e., the codes of $\mathbf{s}_j$ within a batch of signature should be all the same with $\mathbf{q}$'s.

$$\tilde{p}_j^{(\mathbf{R}_i)} = \&_{k=1}^{\tau} p_j^{(\mathbf{r}_{i,k})}, \quad \text{where } p_j^{(\mathbf{r}_{i,k})} = \mathbb{1}_{h(\mathbf{q},\mathbf{r}_{i,k})=h(\mathbf{s}_j,\mathbf{r}_{i,k})} \tag{11}$$

The outputs of m/τ hash signatures are averaged for a low-variance estimation of user interest. It can be formulated as:

$$\begin{aligned} \text{Attn}(\mathbf{q}, \mathbf{S}) &= \frac{1}{m/\tau} \sum_{i=1}^{m/\tau} \ell_2(\tilde{\mathbf{P}}^{(\mathbf{R}_i)}\mathbf{S}) \\ &= \frac{1}{m/\tau} \sum_{i=1}^{m/\tau} \ell_2(\sum_{j=1}^{L} \tilde{p}_j^{(\mathbf{R}_i)} \mathbf{s}_j) \end{aligned} \tag{12}$$

4.2 Analysis of Attention Patterns

4.2.1 Expectation of Hash Sampling-Based Attention. In our method, as $\mathbf{s}_j$ becomes more similar to $\mathbf{q}$, their collision probability becomes higher and the expectation of the coefficient $\tilde{p}_j$ is higher. It can prove that the expectation of $\tilde{p}_j$ is the angle between $\mathbf{s}_j$ and $\mathbf{q}$ on the unit circle [2][3]:

$$\mathbb{E}\left[\tilde{p}_j\right] = \left(1 - \frac{\arccos\left(\mathbf{q}^\top \mathbf{s}_j\right)}{\pi}\right)^{\tau} \tag{13}$$

Therefore, the expectation of user interest representation produced by SDIM is:

$$\begin{aligned} \mathbb{E}\left[\text{Attn}(\mathbf{q}, \mathbf{S})\right] &= \mathbb{E}\left[\ell_2(\sum_{j=1}^{L} \tilde{p}_j^{(\mathbf{R}_k)} \mathbf{s}_j)\right] \\ &= \frac{(1 - \frac{\arccos(\mathbf{q}^\top \mathbf{s}_j)}{\pi})^{\tau}}{\sum_{j=1}^{L}(1 - \frac{\arccos(\mathbf{q}^\top \mathbf{s}_j)}{\pi})^{\tau}} \cdot \mathbf{s}_j \end{aligned} \tag{14}$$

As the number of hash signatures m/τ increases, $\tilde{p}_j$ converges to $\mathbb{E}\left[\tilde{p}_j\right]$, and $\text{Attn}(\mathbf{q}, \mathbf{S})$ converges to $\mathbb{E}\left[\text{Attn}(\mathbf{q}, \mathbf{S})\right]$. The empirical results show that when $m/\tau \geq 16$, the estimation error will be very small. In practice, we use $m = 48$ and $\tau = 3$ for our online model.

We plot the attention weights $\left((1 - \frac{\arccos(\mathbf{q}^\top \mathbf{s}_j)}{\pi})^3\right)$ produced by SDIM in Figure 2. For comparison, we also plot the attention weights $\left(\exp(\mathbf{q}^\top \mathbf{s}_j / 0.5)\right)$ produced by target attention in the same figure. From Figure 2, we can see that the attention weights of SDIM align well with the softmax function in target attention, therefore, **theoretically, our method can obtain outputs that are very close to the target attention**. We name the proposed method as SDIM, which means **S**ampling-based **D**eep **I**nterest **M**odeling.

4.2.2 The Property of τ and Relation to Other Methods. In this subsection, we describe that the width parameter τ in SDIM plays a role in controlling the strength of the model on paying more attention to more similar items.

Denote w_j as the attention weight of target item $\mathbf{q}$ to item $\mathbf{s}_j$, i.e.,

$$w_j = \frac{(1 - \frac{\arccos(\mathbf{q}^\top \mathbf{s}_j)}{\pi})^{\tau}}{\sum_{j=1}^{L}(1 - \frac{\arccos(\mathbf{q}^\top \mathbf{s}_j)}{\pi})^{\tau}} \tag{15}$$

[3]We normalize $\mathbf{q}$ and $\mathbf{s}_j$ before calculating user interests.

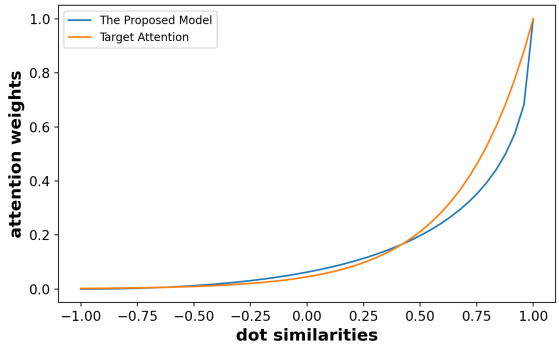

Figure 2: Visualization of attention weights produced by target attention and our model. We left shift the curve of target attention by one unit ($\exp((x-1)/\sqrt{d})$) such that the input is in $[-1, 1]$ (Since the weights will be normalize, this will not change the attention weights). It can be seen that the proposed model produces very similar attention patterns as the target attention.

As τ increases, the entropy of the attention distribution $H(w_j)$ decreases strictly, and the distribution of $\mathbf{w}$ becomes sharper on large similarity region, which encourages the model to pay more attention to more similar items.

Let us consider two extreme cases:

- When $\tau \to +\infty$ (can be approximated by assigning a large value), the algorithm only attends to items that are exactly the same as the candidate item. If we use the category attributes for hashing, then the algorithm behaviors like **SIM (hard)** [11]. Therefore, our algorithm can be seen as an extension of SIM (hard), where it will also considers very similar behavior items but with different category ids.
- When $\tau = 0$, or $m = 1$, the algorithm degenerates into **Mean Pooling** as the target item and user behavior items will always have the same hash signature.

Therefore, SDIM is very flexible that can model different attention patterns by assigning different value of τ.

4.3 Implementation and Complexity Analysis

In this subsection, we elaborate that SDIM is sizable times faster than standard attention-based methods.

Let's review the formulation for calculating user interest in target attention (Eq. 4). This algorithm first gets the attention weights by multiplying the target vector with the behavior sequence, followed by a weighted sum of the behavior sequence using the attention weights. Therefore, two matrix multiplications are needed for the representation of user interest in target attention, and the time complexity of this algorithm is $O(BLd)$.

SDIM decouples the calculation of user interest into two parts: (1) behavior sequence hashing, and (2) target vector hashing. **Note that user behavior sequence are users' inherent features and is independent of the candidate item, which means that the**

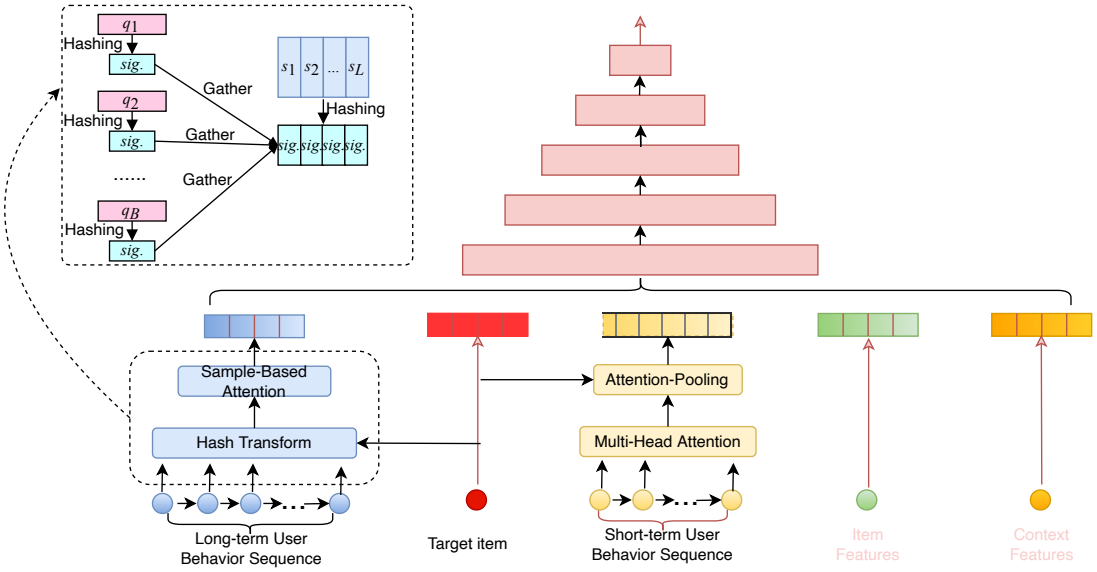

Figure 3: Illustration of our online CTR model. The model takes user interest features, item features and context features as inputs to predict the click probability corresponds to each candidate item.

results of users' behavior sequence hashing are fixed no matter what the candidate item is. Therefore for each user, **we only need to compute the hash transform of user behavior sequence once in each request**. As a result, SDIM reduces B to 1 in the time complexity, thus is sizable times faster than the standard target attention. After hashing, sequence items that share the same hash signature with the target item are selected and summed together to form the user interest. In Tensorflow, this step can be implemented via a **tf.batch_gather** operator, which is an atomic operator of Tensorflow and the time cost is negligible.

The most time-consuming part of SDIM is the multi-round hashing of the behavior sequence, i.e., transform a d-dimensional matrix into m-dimensional hash codes. The time complexity of this operation is $O(Lmd)$ and can be reduced to $O(Lm\log(d))$ using the Approximating Random Projection algorithm [1]. Since $m \ll B$ and $\log(d) \ll d$, **SDIM is much faster than standard attention-based methods**. In our scenario, SDIM achieves **10x-20x speedups** in training the user behavior modeling part.

Method	Training	Online-Serving
DIN	$O(BLd)$	$O(BLd)$
SIM	$O(B\log(M) + Bkd)$	$O(B\log(M) + Bkd)$
ETA	$O(Lm\log(d) + BLm + Bkd)$	$O(BLm + Bkd)$
SDIM	$O(Lm\log(d) + Bm\log(d))$	$O(Bm\log(d))$

Table 1: Time complexity of different methods at training and online-serving stage. B is the number of candidate items for each request, m is the number of hashes, L and k are the length of original and retrieved user behavior sequence respectively, M is the size of attribute inverted index in SIM, and d is the model's hidden size. In our system, B is about 10^3, $L = 1024$, $m = 48$, and $d = 128$.

4.4 Deployment of the Whole System

We introduce how we successfully deploy SDIM online in this subsection. As described above, the whole algorithm is decoupled into two parts: (1) behavior sequence hashing, and (2) target vector hashing. The behavior sequence hashing is independent of the candidate item, which motivates us to build a specialized server to maintain user-wise behavioral sequence hashes.

We divide our system into two parts: **Behavior Sequence Encoding (BSE) server** and **CTR Server**, as shown in Figure 1. The BSE server is responsible for maintaining user-wise behavior sequence hashes. When received a list of user behaviors, BSE server samples from multiple hash functions and generates hash signatures for each item in behavior sequence. The items are then assigned to different buckets according to their signatures, where each bucket corresponds to a hash signature, as shown in Figure 1. The hash buckets are passed to the CTR Server to model candidate item-aware user interests.

The CTR Server is responsible for predicting the click probability of candidate items. When received a batch of B candidate items, the CTR Server hashes each item into signatures, and gathers item representations from the corresponding buckets. The user interest features concatenated with other features are fed to a complex CTR model to get the predicted probabilities of each item. The overall structure of our CTR model is shown in Figure 3. The model takes item features, context features, short-term user interest features, and long-term user interest features as inputs, and uses a multi-layer perceptron to predict the click probability of candidate items.

Please note that SDIM does not require changing the structure of the CTR model, which can be naturally plugged into existing popular CTR architectures. The proposed framework is **end-to-end** at the training stage: the user interest modeling part is jointly trained with the rest of the CTR model, and we deploy them separately only at the online serving stage.

After decoupling BSE and CTR Server, the computation of the BSE is **latency-free** for the CTR Server. In practice, We put the BSE Server before the CTR Server and compute it in parallel with the retrieval module.

After deploying separately, the time cost of calculating user's long-term interest only lies in the hashing of candidate items, which has a $O(Bm\log(d))$ time complexity and is independent of the sequence length L, which means that our framework can handle the user interest modeling with extremely long behaviors theoretically. From the perspective of CTR Server, this time complexity is just feel like adding a common feature. In Table 1, we compare the time complexity of different methods[4].

Our serving system is somewhat similar to MIMN [10]. The biggest difference is that our system can model users' deep interests, while MIMN can only model shallow interests.

4.4.1 Remark on Transmission Cost of Servers. For each request, we need to transmit bucket representations from the BSE server to the CTR server. Notice that we use a fixed number of hash functions, therefore no matter how long the user's behavior is, we only need to transmit fixed-length vectors of bucket representations to the CTR server. In our online system, the size of this vector is 8KB and the transmission cost is about 1ms.

5 EXPERIMENTS

5.1 Dataset and Evaluation Metrics

To verify the effectiveness of SDIM, we conduct experiments on both public and real-world industrial datasets. For public dataset, we follow previous work [3, 10, 11] to choose Taobao dataset[5]. For industrial dataset, we use real-world data collected from the Meituan search system for experiments.

Taobao Dataset: Taobao dataset is released by [20] and is widely used for offline experiments in previous work [11, 12]. Each instance in this dataset is consist of five fields of features: user ID, item ID, category ID, behavior type, and timestamp. Following [12], we additionally introduce the feature of "is_weekend" according to the timestamp to enrich context features. We pre-process the data in the same way with MIMN [10] and ETA [3]. Concretely, we use the 1st to $(L-1)$-th behaviors as inputs to predict the L-th behavior. We split the samples into training set (80%), validation set (10%), and test set (10%) according to the timestep. The recent 16 behaviors are selected as short-term sequence, and the recent 256 behaviors are selected as long-term sequence.

Industrial Dataset: This dataset is collected from the platform searching system of Mobile Meituan APP, which is the largest e-commerce platform for lifestyle services in China. We select consecutive 14-day samples for training and the next two days for evaluation and testing, the number of training examples is about 10 billion. The recent 50 behaviors are selected as short-term sequence, and the recent 1,024 behaviors are selected as long-term sequence. If the number of user behaviors doesn't reach this length, then we pad the sequence to the maximum length with the default value. Besides the user behavior features, we additionally introduce about 20 important id features to enrich the inputs.

Evaluation Metrics: For offline experiments, we follow previous work to adopt the widely used **Area Under Curve (AUC)** for evaluation. We also use the **Training & Inference Speed (T&I Speed)** as a supplement metric to show the efficiency of each model. For online experiments, we use **CTR (Click-Through Rate)** and **VBR (Visited-Buy Rate)** as online metrics.

5.2 Competitive Models

Following previous work [3, 11], we compare SDIM with the following mainstream industrial models for modeling long-term user behaviors:

- **DIN** [18]. DIN is one of the most popular models for modeling user interest in industrial systems. However, DIN is infeasible to be deployed on modeling long-term user behaviors due to its high time complexity. In this baseline, we only use the short-term user behavior features but not long-term features.
- **DIN (Long Seq.)**. For offline experiments, to measure the information gain of long-term user behavior sequences, we equip DIN with long behavior sequences. We set $L = 256$ for Taobao dataset and $L = 1024$ for industrial dataset.
- **DIN (Avg-Pooling Long Seq.)**. This baseline is introduced by [3] and [11], where DIN is applied to modeling short-term user interest, and the long-term user interest is obtained by a mean pooling operation on long-term behaviors. We denote this baseline as **DIN(Avg-Pooling)**.
- **SIM** [11]. SIM first retrieves top-k similar items from the whole sequence via category id, and then applies target attention on top-k items to get user interest. We follow previous work to compare SIM(hard) as the performance are almost the same they deploy SIM(hard) online.
- **UBR4CTR** [12]. UBR4CTR is a two-stage method. At the first stage, they design a feature selection module to select features to form the query, and store the user behaviors in an inverted index manner. At the second stage, the retrieved behaviors are fed to a target attention-based module to get user interest.
- **ETA** [3]. ETA applies LSH to encode the target item and user behavior sequence into binary codes, and then computes the item-wise hamming distance to select top-k similar items for subsequent target attention.

MIMN [10] is proposed by the same team with SIM. As the authors claim that SIM defeats MIMN and they deploy SIM online, we only compare with SIM and omit the baseline of MIMN for the space limitation.

For all the baselines and SDIM, we use the same features (include timeinfo features) as input and adopt the same model structure except for the long-term user behavior modeling module. All models

[4]Since UBR4CTR uses a complex neural network to select features, its time complexity cannot be accurately estimated.
[5]https://tianchi.aliyun.com/dataset/dataDetail?dataId=649

Method	AUC ↑	T&I Speed ↑
DIN(Long Seq.) [18]	0.8848	-
DIN [18]	0.8627*	7.4x
DIN(Avg-Pooling) [3, 11]	0.8669*	2.6x
SIM [11]	0.8692*	2.4x
UBR4CTR [12]	0.8752*	0.8x
ETA [3]	0.8753*	1.8x
SDIM	**0.8854**	5.0x

Table 2: Performance comparison of different model on Taobao dataset. The Training-Speed improvements are calculated on the basis of DIN(Long Seq.). "*" indicates that the improvement of SDIM over this baseline is statistically significant at p-value < 0.05 over paired Wilcoxon-test.

Method	AUC	T&I Speed
DIN(Long Seq.) [18]	**0.7049**	-
DIN [18]	0.6652*	13.5x
DIN(Avg-Pooling) [3, 11]	0.6749*	11.0x
SIM [11]	0.6852*	10.8x
UBR4CTR [12]	0.6836*	1.2x
ETA [3]	0.6906*	3.7x
SDIM	0.7044	11.4x

Table 3: Performance comparison of different model on Industrial dataset. The Training-Speed improvements are calculated on the basis of DIN(Long Seq.). "*" indicates that the improvement of SDIM over this baseline is statiscally significant at p-value < 0.05 over paired Wilcoxon-test.

use the same length of long-term user behaviors (T=256 for Taobao and T=1024 for industrial dataset).

6 RESULTS AND ANALYSIS

6.1 Results on Taobao Dataset

The overall results of different models on Taobao dataset are shown in Table 2. We can draw the following conclusions: (1) SDIM performs on par with DIN(Long Seq.) on modeling long-term user behaviors, while being 5x times faster. As described above, SDIM can simulate an attention pattern that very similar as the target attention, therefore SDIM can match or even surpass the performance of DIN(Long Seq.).

(2) SDIM performs better than all baseline models that are proposed for modeling long-term user behaviors. Concretely, SDIM outperforms SIM by 1.62%, outperforms UBR4CTR by 1.02%, and outperforms ETA by 1.01%. We also notice that DIN(Long Seq.) brings 2.21% improvement on AUC compared to DIN, which indicates the importance of modeling the long-term user behaviors for CTR prediction. SIM, UBR4CTR and ETA performs worse than DIN(Long Seq.), which is due to the loss of information caused by the retrieval of user behaviors. These retrieval operations may be helpful in removing noise away from the sequence, but is harmful when there is not enough informative behavior for top-k retrieval.

(3) SDIM is much more efficient than DIN(Long Seq.), SIM and UBR4CTR. The efficiency improvement can be ascribed to reducing the time complexity of the algorithm to $O(Lm\log(d))$. SDIM is also much more efficient than ETA [3]. ETA also uses LSH to hash the target item and behavior sequence, the time complexity of the hash operation is the same as SDIM. After hashing, ETA calculates the hamming distance and selects top-k items for target attention, the time complexity is $O(BL + Bkd)$. While SDIM only introduces a gather operator followed by a mean pooling of m/τ hashes, thus is much more efficient than ETA.

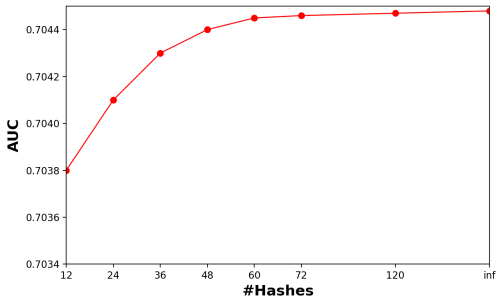

Figure 4: AUC results of SDIM on industrial dataset when altering the number of hashes m.

6.2 Results on Industrial Dataset

The overall results of different models on industrial dataset are shown in Table 3. Similar to the results from Taobao dataset, SDIM outperforms all competitive baselines, and performs on par with DIN(Long Seq.). Our SDIM achieves 1.92%, 2.08%, 1.38% improvements compared with SIM, UBR4CTR, and ETA respectively, while being much faster than these methods.

Since the user sequence length in industrial Dataset is large enough which is friendly to retrieve-based methods, it seems that they should perform on par with DIN(Long Seq.). However, the results in Table 3 show that their performance has some gaps with DIN(Long Seq.). We argue that this is because user's interest are typically diverse and people often want to buy new categories of goods, especially in our food search cases. When faced with a candidate item in a new category, it's difficult for these retrieve algorithms to accurately pick out the most valuable items from users' historical behaviors.

Compared with the Taobao dataset, the industrial dataset contains more candidate items per request, so SDIM can achieve more speedups on this dataset. The industrial dataset also has longer user behavior sequences (T=1024), so retrieve-based methods also achieve more speedups. The experimental results demonstrate the superiority of SDIM.

6.3 Hyper-Parameter Analysis

There are two important hyper-parameters in SDIM: (1) the number of hashes m, and (2) the width of hash signatures τ.

Hyper-Parameter	AUC
$\tau = 1$	0.6911
$\tau = 2$	0.7032
$\tau = 3$	**0.7044**
$\tau = 5$	0.7034
$\tau = 10$	0.6923

Table 4: AUC results of SDIM on industrial dataset when altering the width parameter τ.

Method	AUC	T&I Speed
DIN(T=16)	0.8627	-
SDIM(T=16)	0.8637	2.0x

Table 5: Performance comparison of different model on modeling short-term user's interests.

6.3.1 Analysis on m. The number of hashes m affects the estimation error of the proposed hashing-based attention. As the number of sampled hash function increases, the estimated user interest will be more close to $\mathbb{E}\left[\text{Attn}(\mathbf{q}, \mathbf{S})\right]$ in Eq. 14.

To assess the estimation error, we test the performance of SDIM using m hashes, where $m \in \{24, 36, 48, 60, 72, 90, 120\}$. We also implement a variation of SDIM that directly compute attention weights using the expected collision probability $\mathbb{E}\left[\tilde{p}_j\right]$ in Eq. 13. This baseline simulates the behavior of SDIM when the number of hash signatures tends to be infinite. The results are shown in Figure 4. It can be seen that when $m > 48$ the models perform almost the same. For the sake of efficiency, we use $m = 48$ in our model.

6.3.2 Analysis on τ. As we described in subsection 4.2.2, τ controls the strength of the model on paying more attention to more similar items. We investigate different attention patterns of SDIM by varying τ in {1,2,3,5,10}. The results are shown in Table 4.

From Table 4, we can see that SDIM performs well when $2 \leq \tau \leq 5$. For the balance of effectiveness and efficiency, we use $\tau = 3$ in our online model. The model performs poorly when $\tau = 1$ because it encodes too many noise behaviors. On the contrary, the model performs poorly when $\tau = 10$ as only very similar items have the chance to contribute to user interest, which is unfriendly to users with few behaviors.

6.4 Experiments on Short-Term User Behavior Modeling

We also conduct an extra experiment to test SDIM's performance on modeling short-term user behaviors. But please note that SDIM is mainly proposed to solve the problem of long-term user interest modeling for industrial recommendation systems, and one can directly plug in the full target attention or more complexity module to model short sequence. We conduct this experiments to just show model's performance on special cases. We conduct this experiment on Taobao dataset, and the results are shown in Table 5.

Method	CTR	VBR	Inf. Time
Base(w/o long seq.)	-	-	≈ 60ms
DIN(T=1024)	can not deploy		
SDIM(T=1024)	+2.39%	+2.21%	+1ms
SDIM(T=2000)	+2.98%	+2.69%	+1ms

Table 6: Online A/B testing results.

The results demonstrate that SDIM can still achieve comparable results with standard target attention model on modeling short sequence, while being more efficient.

6.5 Online A/B Test

A strict online A/B testing is also conducted to verify the effectiveness of SDIM. For online A/B testing, the baseline model is the previous online CTR model in Meituan search system, where only short-term user behavior sequences are used. The test model adopts the same structure and features as the base model, but incorporates a long-term user interests modeling module with users recent 1,024 or 2,000 behaviors on this basis. We use the proposed SDIM to model long-term user interests, and denote this test model as SDIM(T=1024) and SDIM(T=2000). The testing lasts for 14 days, with 10% of Meituan search traffic are distributed for each model respectively. The results of A/B testing are shown in Table 6.

SDIM(T=2000) achieves 2.98% improvements (p-value < 0.001) on CTR and 2.69% improvements (p-value < 0.005) on VBR compared with the base model, which can greatly increase online profit considering the large traffic of Meituan APP. The inference time of SDIM(T=2000) increased by 1ms compared with the Base(w/o long seq.). The increase in inference time is mostly caused by the transmission time between BSE Server and CTR Server.

We also tried to deploy the model that directly uses the target attention to model long-term user behavior sequences with $T = 1024$. However, its inference time is greatly increased by about 50% (25ms-30ms), which is unacceptable for our system. Therefore we can't leave this model online for 14 days for A/B Testing. SDIM performs on par with this model, but saves 95% online inference time. SDIM has now been deployed online and serve the main traffic of Meituan's home search system.

7 CONCLUSIONS

In this paper, we propose a hash sampling-based method named SDIM for modeling long-term user behaviors. Instead of designing complicated modules to retrieve from long-term user behaviors, we directly gather behavior items associated with the candidate item with the same hash signature to form the user interest. We also propose a new online serving framework that decouples the hashing of user behavior sequence from the entire model, which makes it latency-free for the CTR server. We show that the proposed method performs on par with DIN(Long Seq.), while being sizable times faster. SDIM has been deployed online in Meituan APP.

Future work includes reducing the transmission cost, exploring more complex structures such as multi-head hashing, and so on.

REFERENCES

[1] Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya P. Razenshteyn, and Ludwig Schmidt. 2015. Practical and Optimal LSH for Angular Distance. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. 1225–1233. https://proceedings.neurips.cc/paper/2015/hash/2823f4797102ce1a1aec05359cc16dd9-Abstract.html

[2] Moses Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, John H. Reif (Ed.). ACM, 380–388. https://doi.org/10.1145/509907.509965

[3] Qiwei Chen, Changhua Pei, Shanshan Lv, Chao Li, Junfeng Ge, and Wenwu Ou. 2021. End-to-End User Behavior Retrieval in Click-Through RatePrediction Model. *CoRR* abs/2108.04468 (2021). arXiv:2108.04468 https://arxiv.org/abs/2108.04468

[4] 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. In *IJCAI 2019, Macao, China, August 10-16, 2019*. ijcai.org, 2301–2307. https://doi.org/10.24963/ijcai.2019/319

[5] Wang-Cheng Kang and Julian J. McAuley. 2018. Self-Attentive Sequential Recommendation. In *IEEE International Conference on Data Mining, ICDM 2018, Singapore, November 17-20, 2018*. IEEE Computer Society, 197–206. https://doi.org/10.1109/ICDM.2018.00035

[6] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=rkgNKkHtvB

[7] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. 2014. *Mining of Massive Datasets, 2nd Ed.* Cambridge University Press. http://www.mmds.org/

[8] 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, CIKM 2019, Beijing, China, November 3-7, 2019*. ACM, 2615–2623. https://doi.org/10.1145/3357384.3357814

[9] Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. 2007. Detecting near-duplicates for web crawling. In *WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*. ACM, 141–150. https://doi.org/10.1145/1242572.1242592

[10] 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 *KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. ACM, 2671–2679. https://doi.org/10.1145/3292500.3330666

[11] Qi Pi, Guorui Zhou, Yujing Zhang, Zhe Wang, Lejian Ren, Ying Fan, Xiaoqiang Zhu, and Kun Gai. 2020. Search-based User Interest Modeling with Lifelong Sequential Behavior Data for Click-Through Rate Prediction. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*. ACM, 2685–2692. https://doi.org/10.1145/3340531.3412744

[12] Jiarui Qin, Weinan Zhang, Xin Wu, Jiarui Jin, Yuchen Fang, and Yong Yu. 2020. User Behavior Retrieval for Click-Through Rate Prediction. In *SIGIR 2020, Virtual Event, China, July 25-30, 2020*. ACM, 2347–2356. https://doi.org/10.1145/3397271.3401440

[13] Kan Ren, Jiarui Qin, Yuchen Fang, Weinan Zhang, Lei Zheng, Weijie Bian, Guorui Zhou, Jian Xu, Yong Yu, Xiaoqiang Zhu, and Kun Gai. 2019. Lifelong Sequential Modeling with Personalized Memorization for User Response Prediction. In *SIGIR 2019, Paris, France, July 21-25, 2019*. ACM, 565–574. https://doi.org/10.1145/3331184.3331230

[14] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*. ACM, 565–573. https://doi.org/10.1145/3159652.3159656

[15] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-Attention with Linear Complexity. *CoRR* abs/2006.04768 (2020). arXiv:2006.04768 https://arxiv.org/abs/2006.04768

[16] Zhanpeng Zeng, Yunyang Xiong, Sathya N. Ravi, Shailesh Acharya, Glenn Moo Fung, and Vikas Singh. 2021. You Only Sample (Almost) Once: Linear Cost Self-Attention Via Bernoulli Sampling. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 12321–12332. http://proceedings.mlr.press/v139/zeng21a.html

[17] 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 *AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 5941–5948. https://doi.org/10.1609/aaai.v33i01.33015941

[18] Guorui Zhou, Xiaoqiang Zhu, Chengru 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 2018, London, UK, August 19-23, 2018*. ACM, 1059–1068. https://doi.org/10.1145/3219819.3219823

[19] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *AAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 11106–11115. https://ojs.aaai.org/index.php/AAAI/article/view/17325

[20] 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 2018, London, UK, August 19-23, 2018*. ACM, 1079–1088. https://doi.org/10.1145/3219819.3219826