Deep Multi-Interest Network for Click-through Rate Prediction

Zhibo Xiao, Luwei Yang, Wen Jiang, Yi Wei, Yi Hu, Hao Wang
Alibaba Group, Hangzhou, China
{xiaozhibo.xzb,luwei.ylw,wen.jiangw}@alibaba-inc.com

ABSTRACT

Click-through rate prediction plays an important role in many fields, such as recommender and advertising systems. It is one of the crucial parts to improve user experience and increase industry revenue. Recently, several deep learning-based models are successfully applied to this area. Some existing studies further model user representation based on user historical behavior sequence, in order to capture dynamic and evolving interests. We observe that users usually have multiple interests at a time and the latent dominant interest is expressed by the behavior. The switch of latent dominant interest results in the behavior changes. Thus, modeling and tracking latent multiple interests would be beneficial. In this paper, we propose a novel method named as Deep Multi-Interest Network (DMIN) which models user's latent multiple interests for click-through rate prediction task. Specifically, we design a Behavior Refiner Layer using multi-head self-attention to capture better user historical item representations. Then the Multi-Interest Extractor Layer is applied to extract multiple user interests. We evaluate our method on three real-world datasets. Experimental results show that the proposed DMIN outperforms various state-of-the-art baselines in terms of click-through rate prediction task.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; *Learning to rank*; *Personalization.*

KEYWORDS

Recommender System; Click-through Rate prediction; Multi-interest

ACM Reference Format:
Zhibo Xiao, Luwei Yang, Wen Jiang, Yi Wei, Yi Hu, Hao Wang. 2020. Deep Multi-Interest Network for Click-through Rate Prediction. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3340531.3412092

1 INTRODUCTION

Click-through rate (CTR) prediction is the task of estimating the likelihood that an item will be clicked by the user, which plays an important role in many fields, such as recommender and advertising systems. For example, there are two main parts in e-commerce recommender system [4], i.e, matching and ranking. Matching is able to retrieve several thousands of candidate items from a huge item set with hundreds of millions of items. Ranking is responsible for scoring these candidates in terms of click-through rate prediction task. Sometimes, a conversion rate task can be used here. Finally, the scored items are presented to the end user by a descending order. Note that, additional reranking can be added in accordance with some rule-based models.

In this paper, we mainly focus on ranking, and treat it as a click-through rate prediction task. Recently, many deep learning-based methods have been proposed to successfully improve the performance of click-through rate prediction task [2, 5, 7, 10, 11, 13, 14]. By observing user's interest is expressed by his/her behavior sequences, several models [4, 9, 16, 17] have focused on extracting user interest from the behavior sequences. In DIN [17], an attention-based method is utilized to capture relative interests from the user behavior sequence with regard to candidate item. However, it ignores the ordering of user behavior sequence. DIEN [16] further uses a specially designed GRU structure to capture the evolution of user interest. DSIN [4] introduces a hierarchical view of behavior sequence by dividing it into sessions. Then it models each session with self-attention network to capture multiple user interests.

Nonetheless, we observe that users have multiple diverse interests at a time and the latent dominant interest is expressed by the behavior. The switch of latent dominant interest results in behavior changes. Motivated by above observations, we propose a Deep Multi-Interest Network (DMIN) to improve CTR task by modeling latent multiple user interests from user behavior sequence. There are two main components: *Behavior Refiner Layer* and *Multi-Interest Extractor Layer. Behavior Refiner Layer* is applied to refine the representation of item in user behavior sequence.*Multi-Interest Extractor Layer* is able to capture latent multiple interests.

The main contributions of this paper are as follows:

- We highlight the multi-interest phenomenon in e-commerce field, and focus on modeling latent multiple user interests.
- We propose a novel model with two main components, *Behavior Refiner Layer* and *Multi-Interest Extractor Layer.*
- We evaluate our proposed method on three real-world datasets in terms of click-through rate prediction. The results show the efficacy of our proposed DMIN.

2 THE PROPOSED METHOD

In this section, we introduce our proposed approach Deep Multi-Interest Network (DMIN). The overall architecture is illustrated by Figure 1. DMIN follows the basic paradigm of embedding & Multilayer Perceptron (MLP) model [17]. The two main components of DMIN, *Behavior Refiner Layer* and *Multi-Interest Extractor Layer* are put into middle to better model and extract user interests.

DMIN takes as input the user historical behaviors, user profile feature, context feature and target item. It firstly embeds these input features as low-dimensional vectors by an embedding layer. Then

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3412092

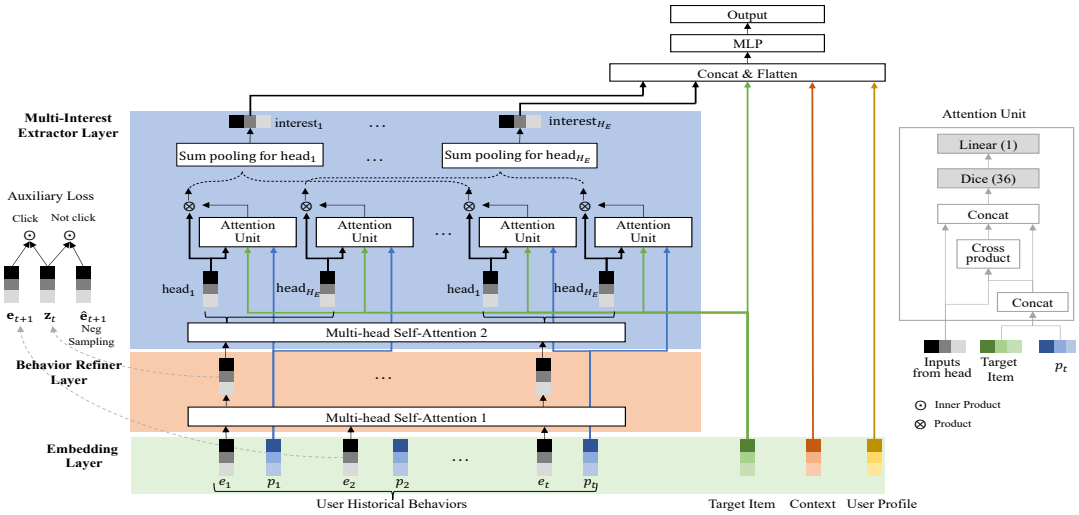

Figure 1: The architecture of DMIN.

Behavior Refiner Layer refines each user historical behavior item representation as z_t with the help of auxiliary loss and multi-head self-attention. In Multi-Interest Extractor Layer, another multi-head self-attention layer and a local attention unit are introduced to extract multiple interests. The multiple interests and embedding vectors of remaining features are concatenated, and fed into a MLP for final CTR prediction.

2.1 Embedding Layer

There are four groups of features: *User Profile*, *User Historical Behavior*, *Context* and *Target Item*. *User Profile* contains features related to the user, e.g. *user id*, *country* and so on. *Target Item* refers to the candidate item with corresponding features such as *item id*, *category id*, *statistical ctr* and so on. *User Historical Behavior* is a list of user interacted items by clicking, purchasing or add-to-cart. Each item in this list has same feature fields as *Target Item*. *Context* is a group of features including but not limited to *time*, *match type*, *trigger id* and so on.

Each feature can be encoded into a one-hot vector with high-dimension. These features are usually very sparse and should be transformed into low-dimensional dense features by embedding layer, which is a common operation in deep learning-based ranking methods [2]. For example, the *item id* can be represented by a matrix $\mathbf{E} \in \mathbb{R}^{K \times d_v}$, where K is the total number of items and d_v is the embedding size with $d_v \ll K$. With embedding layer, *User Profile*, *User Historical Behavior*, *Context* and *Target Item* can be represented as $\mathbf{x}_u$, $\mathbf{x}_b$, $\mathbf{x}_c$ and $\mathbf{x}_t$, respectively. Especially, *User Historical Behavior* contains multiple items and can be represented by $\mathbf{x}_b = \{\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_T\} \in \mathbb{R}^{T \times d_{model}}$, where T is the number of user's history behaviors and d_{model} is the dimension of item embedding $\mathbf{e}_t$. Note that items in *User Historical Behavior* and *Target Item* share the same embedding matrices. $\mathbf{p}_t \in \mathbb{R}^{d_p}$ is the position encoding for the t-th item and d_p is the dimension of $\mathbf{p}_t$.

2.2 Behavior Refiner Layer

In this part, we describe how to refine the item representation using multi-head self-attention [3, 12] from user behavior sequence. One of the naive methods to obtain the simple item representation is just concatenating each item's feature embedding. However, in practice, we found that good item representations are greatly benefit to downstream task. Thus we employ multi-head self-attention to refine the item representation. The capability of keeping contextual sequential information and capturing relationships between elements in the sequence make multi-head self-attention stand out in this task.

Self-attention is a special attention mechanism, which has been successfully applied to a variety of tasks[3, 12, 15]. The input of the self-attention module consists of query, key, and value and these three components come from the same place. Multi-head self-attention is a combination of multiple self-attention structures, which can learn the relationship in different representation subspaces[12]. To be specific, the output of the $\mathbf{head}_h$ is calculated as follows,

$$\begin{aligned}\mathbf{head}_h &= \text{Attention}(\mathbf{x}_b\mathbf{W}_h^Q, \mathbf{x}_b\mathbf{W}_h^K, \mathbf{x}_b\mathbf{W}_h^V) \\ &= \text{Softmax}\left(\frac{\mathbf{x}_b\mathbf{W}_h^Q \cdot (\mathbf{x}_b\mathbf{W}_h^K)^\top}{\sqrt{d_h}} \cdot \mathbf{x}_b\mathbf{W}_h^V\right),\end{aligned} \tag{1}$$

where $\mathbf{W}_h^Q, \mathbf{W}_h^K, \mathbf{W}_h^V \in \mathbb{R}^{d_{model} \times d_h}$ are projection matrices of the h-th head for query, key and value respectively. Thus each $\mathbf{head}_h$ represents a latent item representation in a subspace.

Then vectors of different heads are concatenated to form the refined item representations, which can be defined as follows,

$$\mathbf{Z} = \text{MultiHead}(\mathbf{x}_b) = \text{Concat}(\mathbf{head}_1, \mathbf{head}_2, ..., \mathbf{head}_{H_R})\mathbf{W}^O, \tag{2}$$

where H_R is the number of heads, $\mathbf{W}^O \in \mathbb{R}^{d_{model} \times d_{model}}$ is a linear matrix. Moreover, we also conduct residual connection [6], dropout [8] and layer normalization [1] in the self-attention.

Inspired by [16], an auxiliary loss is used to supervise better item representation learning. It uses behavior $\mathbf{e}_{t+1}$, the $(t+1)$-th behavior original item embedding, to supervise the learnt item representation at time t, $\mathbf{z}_t$, which is the t-th row vector of $\mathbf{Z}$. The positive example is the real next behavior and the negative example is sampled from the whole item set excluding the clicked items. Mathematically, the auxiliary loss is formulated as,

$$L_{aux} = -\frac{1}{N}(\sum_{i=1}^{N}\sum_{t}\log\sigma(\langle \mathbf{z}_t^i, \mathbf{e}_{t+1}^i\rangle) + \log(1-\sigma(\langle \mathbf{z}_t^i, \hat{\mathbf{e}}_{t+1}^i\rangle))), \quad (3)$$

where $\sigma(.)$ is the sigmoid activation function and $\langle , \rangle$ denotes the inner product. $\hat{\mathbf{e}}_{t+1}^i$ is the original embedding of negative example, N represents the number of training examples.

2.3 Multi-Interest Extractor Layer

After obtaining the refined item representations, we need to extract multiple interests in this refined sequence. We use another multi-head self-attention same as previous part to capture multiple interests,

$$\begin{aligned}\mathbf{head}_h' &= \text{Attention}(\mathbf{Z}\mathbf{W}'^{Q}_{h}, \mathbf{Z}\mathbf{W}'^{K}_{h}, \mathbf{Z}\mathbf{W}'^{V}_{h}) \\ &= \text{Softmax}\Big(\frac{\mathbf{Z}\mathbf{W}'^{Q}_{h}\cdot(\mathbf{Z}\mathbf{W}'^{K}_{h})^{\top}}{\sqrt{d_{model}}}\cdot\mathbf{Z}\mathbf{W}'^{V}_{h}\Big),\end{aligned} \quad (4)$$

similarly, $\mathbf{W}'^{Q}_{h}, \mathbf{W}'^{K}_{h}, \mathbf{W}'^{V}_{h} \in \mathbb{R}^{d_{model}\times d_{model}}$ are projection matrices of the h-th head for query, key and value respectively. We pack all output head vectors $\{\mathbf{head}_1', \mathbf{head}_2', ..., \mathbf{head}_{H_E}'\}$ as a matrix $\mathbf{I} \in \mathbb{R}^{T\times H_E\times d_{model}}$. H_E is the number of heads, which is also equivalent to the number of user interests.

Inspired by [17], we use an attention unit to capture the relevance of each output head with respect to the target item, which is shown on the right of Figure 1. Besides, we add a position embedding to incorporate position information. Thus, the h-th interest of user can be formulated as,

$$\mathbf{interest}_h = \sum_{j=1}^{T} a(\mathbf{I}_{jh}, \mathbf{x}_t, \mathbf{p}_j)\mathbf{I}_{jh} = \sum_{j=1}^{T} w_j\mathbf{I}_{jh}, \quad (5)$$

where $\mathbf{I}_{jh} \in \mathbb{R}^{d_{model}}$ represents the h-th head's vector of the j-th item, $\mathbf{p}_j \in \mathbb{R}^{d_p}$ is the position encoding for j-th item. Note that the position of each item in the user behavior is the reverse sequential order sorted by timestamp when it is occurred, i.e. the more recent occurred behavior items will be put at the higher ranks. a denotes the attention unit which is shown on the right of Figure 1. It tells how relevant of the $\mathbf{interest}_h$ to the target item.

The attention unit takes the output of the second multi-head self-attention $\mathbf{I}$, target item embedding $\mathbf{x}_t$ and position embedding $\mathbf{p}_j$ as input. The corresponding position embedding of each head and the target item vector are concatenated and then transformed into a vector with the same size as the input vectors from each head by a linear transformation. Then we apply a cross-product to the concatenated vector of the target item embedding and position embedding with the input from head. The result of them together with the original value are fed into a fully-connected network. The final output is a scalar which indicates to what extent is the interest relevant to the target item.

Therefore, multiple interests of users can be obtained. The final output vectors of *User Historical Behavior* is now represented as $\hat{\mathbf{x}}_b = \{\mathbf{interest}_1, \mathbf{interest}_2, ..., \mathbf{interest}_{H_E}\} \in \mathbb{R}^{H_E\times d_{model}}$. Note that, the number of user's interests directly depends on the number of headers, H_E.

2.4 MLP & Loss Function

All the features vectors, $\mathbf{x}_u$, $\hat{\mathbf{x}}_b$, $\mathbf{x}_c$ and $\mathbf{x}_t$, are concatenated and fed into the MLP layer for final prediction. Since the click-through rate prediction task is a binary classification task, the loss function is chosen as cross-entropy loss, which is usually defined as:

$$L_{target} = -\frac{1}{N}(\sum_{i=1}^{N} y_i\log f(\mathbf{x}_i) + (1-y_i)\log(1-f(\mathbf{x}_i))), \quad (6)$$

where $\mathbf{x}_i = (\mathbf{x}_u, \mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_t) \in \mathcal{D}$, $\mathcal{D}$ is the training set with size N. $y_i \in \{0, 1\}$ is the click label, $f(x)$ is the prediction output of our network. As we use the auxiliary loss to supervise item representation refining, the final loss can be defined as,

$$L_{total} = L_{target} + \lambda L_{aux}, \quad (7)$$

where λ is a hyper-parameter in order to balance the two sub tasks.

3 EXPERIMENTS

In this section, we conduct experiments and compare with several state-of-the-art methods on three real-world datasets.

3.1 Datasets

We use three real-world datasets for evaluation. The statistics of them are summarized in Table 1.

Public Datasets. Amazon Dataset contains product reviews and metadata from Amazon[1]. We conduct experiments on two subsets named *Electronics* and *Books*. These datasets collect user behavior sorted by timestamp. Assuming there are K reviewed products in a user behavior sequence, our aim is to predict whether the user u will write reviews for the K-th product based on the first K-1 reviewed products. We create training, validation and test sets by random sampling from the original dataset with a split rate as 80%, 10% and 10%.

Alibaba.com. We sampled part of the user behavior logs from the alibaba.com online recommendation system[2] as a new dataset. We use logs of the first 7 days as training set, logs in the 8th day as validation set and logs in the 9th day as test set.

Table 1: Dataset

Dataset	Users	Items	Categories	Samples
Books	603,668	367,982	1,600	603,668
Electronics	192,403	63,001	801	192,403
Alibaba.com	177,947	4,996,525	5057	1,200,000

3.2 Compared Models

We compare DMIN[3] with some mainstream CTR prediction methods:

- **Wide&Deep** [2] is a widely applied method in industrial applications. It contains a deep part and a wide part, which combines the memorization of a linear model and the generalization of a DNN model.
- **PNN** [10] uses a product layer to capture high-order feature interactions.
- **DIN** [17] applies attention both to user historical sequence and the target item in order to better model user interest subject to the target item.

[1] http://jmcauley.ucsd.edu/data/amazon/
[2] https://www.alibaba.com/
[3] The source code is available at https://github.com/mengxiaozhibo/DMIN

- **DIEN** [16] can be considered as an improved version of DIN, it uses GRU with attentional update gate to model the evolution user interest.

We use the implementations of above baselines that provided by [16]. Note that, by removing the first multi-head self-attention, the auxiliary loss and the position embedding, and setting H_E equals 1, our model is almost equivalent to DIN.

3.3 Experimental Results

The dimension of item embedding (d_{model}) is set as 36. The embedding size of position encoding is set as 2. The dimension of user profile embedding is 18. The batch size is 128, the learning rate is 0.001 and the λ is 1. Different datasets have different maximum length of user behavior sequences. This maximum length set by us is related to distribution of user behavior lengths in the dataset. Therefore, we set the maximum length to 10, 20, 80 in *Electronics*, *Books* and *Alibaba.com*, respectively. According to the performance and effect of the experiments, H_R is set to be 4 and H_E can be set to 2/4. The dropout rate is set to 0.2.

We use Area Under ROC (AUC) as evaluation metric. All experiments are repeated 5 times and the average and standard deviation of the results are reported. The experimental results on three real-world datasets are shown by Table 2. We can find Wide&Deep with manually designed features performs not well. PNN is table to automatically learn the interaction between features, which beats Wide&Deep. DIN represents the user interests with regard to target item and the result beats Wide&Deep and PNN. DIEN uses a specially designed GRU structure to capture the evolution of user interest, which helps to obtain better interest representation than DIN. DMIN achieves the highest AUC score among all three datasets, which shows the efficacy of modeling and tracking user's latent multiple interests. As shown in Table 2, the usage of auxiliary loss and position embedding bring a greater gain.

Figure 2 shows an example of captured multiple interests for a user in Alibaba.com dataset. The user's historical sequence was alternating between pants and shoes. The **interest**$_1$ and **interest**$_2$ successfully captured user interests in pants and shoes respectively. The color represents the output from the attention unit, the darker the color is the corresponding interest is more dominant. We can see that the switch of latent dominant interest results in the behavior changes.

Figure 2: Example of captured multiple interests for a user in Alibaba.com dataset.

4 CONCLUSIONS

In this paper, we have proposed a novel method named as Deep Multi-Interest Network (DMIN) to model user's latent multiple interests for click-through rate prediction task. Specifically, we design a Behavior Refiner Layer using multi-head self-attention to capture better user historical item representations. Then the Multi-Interest Extractor Layer is applied to extract multiple user interests.

Table 2: Experimental Results (AUC) on Three Real-world Dtasets.

Model	Electronics	Books	Alibaba.com
Wide&Deep	0.7456 ± 0.0018	0.7788 ± 0.0016	0.7769 ± 0.0017
PNN	0.7543 ± 0.0004	0.7824 ± 0.0025	0.7825 ± 0.0015
DIN	0.7589 ± 0.0006	0.7903 ± 0.0013	0.7837 ± 0.0014
DIEN	0.7707 ± 0.0032	0.8445 ± 0.0025	0.7881 ± 0.0015
DIEN-NO-AUX[a]	0.7604 ± 0.0006	0.7980 ± 0.0019	0.7861 ± 0.0022
DMIN-NO-AUX[b]	0.7630 ± 0.0005	0.8020 ± 0.0013	0.7872 ± 0.0012
DMIN-NO-PE[c]	0.7890 ± 0.0009	0.8620 ± 0.0010	0.7930 ± 0.0013
DMIN	**0.7918 ± 0.0004**	**0.8670 ± 0.0011**	**0.7950 ± 0.0008**

[a] DIEN without auxiliary loss
[b] DMIN without auxiliary loss
[c] DMIN without position embedding

Experimental results show that the proposed DMIN outperforms various state-of-the-art baselines in terms of click-through rate prediction task. In the future, we will try more different methods to model user's latent multiple interests.

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).
[2] 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 *the 1st workshop on deep learning for recommender systems*.
[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[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 *the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*.
[5] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*.
[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on CVPR*.
[7] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*.
[8] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* (2012).
[9] Ze Lyu, Yu Dong, Chengfu Huo, and Weijun Ren. 2020. Deep Match to Rank Model for Personalized Click-Through Rate Prediction. In *AAAI*.
[10] 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 *IEEE ICDM*.
[11] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*.
[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*.
[13] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*.
[14] 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. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*.
[15] Shuai Zhang, Yi Tay, Lina Yao, and Aixin Sun. 2018. Next item recommendation with self-attention. *arXiv preprint arXiv:1808.06414* (2018).
[16] 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*.
[17] 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*.