Click-Through Rate Prediction with the User Memory Network

Wentao Ouyang, Xiuwu Zhang, Shukui Ren, Li Li, Zhaojie Liu, Yanlong Du
Intelligent Marketing Platform, Alibaba Group
Beijing, China
{maiwei.oywt,xiuwu.zxw,shukui.rsk,ll98745,zhaojie.lzj,yanlong.dyl}@alibaba-inc.com

ABSTRACT

Click-through rate (CTR) prediction is a critical task in online advertising systems. Models like Deep Neural Networks (DNNs) are simple but stateless. They consider each target ad independently and cannot directly extract useful information contained in users' historical ad impressions and clicks. In contrast, models like Recurrent Neural Networks (RNNs) are stateful but complex. They model temporal dependency between users' sequential behaviors and can achieve improved prediction performance than DNNs. However, both the offline training and online prediction process of RNNs are much more complex and time-consuming. In this paper, we propose Memory Augmented DNN (MA-DNN) for practical CTR prediction services. In particular, we create two external memory vectors for each user, memorizing high-level abstractions of what a user possibly likes and dislikes. The proposed MA-DNN achieves a good compromise between DNN and RNN. It is as simple as DNN, but has certain ability to exploit useful information contained in users' historical behaviors as RNN. Both offline and online experiments demonstrate the effectiveness of MA-DNN for practical CTR prediction services. Actually, the memory component can be augmented to other models as well (e.g., the Wide&Deep model).

CCS CONCEPTS

• **Information systems** → **Online advertising**;

KEYWORDS

Click-through rate prediction; Online advertising; Deep learning

ACM Reference Format:
Wentao Ouyang, Xiuwu Zhang, Shukui Ren, Li Li, Zhaojie Liu, Yanlong Du. 2019. Click-Through Rate Prediction with the User Memory Network. In *1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data (DLP-KDD'19), August 5, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3326937.3341258

1 INTRODUCTION

Click-through rate (CTR) prediction is to predict the probability that a user will click on an item. It plays an important role in online advertising systems. For example, the ad ranking strategy generally depends on CTR × bid, where bid is the benefit the system receives if an ad is clicked by a user. Moreover, according to the common cost-per-click charging model, advertisers are only charged once their ads are clicked by users. Therefore, in order to maximize the revenue and to maintain a desirable user experience, it is crucial to estimate the CTR of ads accurately.

CTR prediction has attracted lots of attention from both academia and industry [2, 8, 9, 17, 20]. For example, Logistic Regression (LR) [15] models linear feature importance. Factorization Machine (FM) [14] further models pairwise feature interactions and shows improved performance. In recent years, Deep Neural Networks (DNNs) are exploited for CTR prediction in order to automatically learn feature representations and high-order feature interactions [4, 18]. To take advantage of both shallow and deep models, hybrid models are also proposed. For example, Wide&Deep [2] combines LR and DNN to improve both the memorization and generalization abilities of the model. DeepFM [7] combines FM and DNN to further improve the model ability of learning feature interactions.

One major limitation of the aforementioned models is that they mainly consider each target ad independently, but cannot directly extract useful information contained in users' historical ad impressions and clicks. To fill this gap, Zhang et al. [19] use Recurrent Neural Networks (RNNs) to model the dependency on users' sequential behaviors for CTR prediction. Tan et al. [16] propose improved RNNs for session-based recommendations. These RNN-based models do achieve improved performance. However, state-of-the-art RNN cell structures like Long Short-Term Memory (LSTM) [10] and Gated Recurrent Unit (GRU) [3] are recursive and complex, which makes both the offline training and the online prediction process time-consuming. Moreover, the data preparation for RNNs is also complex. These issues make the application of RNNs to practical CTR prediction services rather complex (will be illustrated in §2.2).

In this paper, we propose Memory Augmented Deep Neural Network (MA-DNN) for CTR prediction in online advertising systems. In particular, we augment a DNN with an external user memory component, memorizing high-level abstractions of what a user possibly likes and dislikes. As a result, the proposed MA-DNN achieves a good compromise between DNN and RNN. It is as simple as DNN, but has certain ability to exploit useful information contained in users' historical behaviors as RNN. Actually, our proposal is flexible that the memory component can be augmented to other models such as Wide&Deep [2] and DeepFM [7] as well. Both offline and online experiments demonstrate the effectiveness of the memory augmented neural networks for practical CTR prediction services.

2 MODEL DESIGN

2.1 Problem Statement

The task of CTR prediction in online advertising is to estimate the probability of a user clicking on a specific ad. Table 1 shows some

DLP-KDD'19, August 5, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6783-7/19/08…$15.00
https://doi.org/10.1145/3326937.3341258

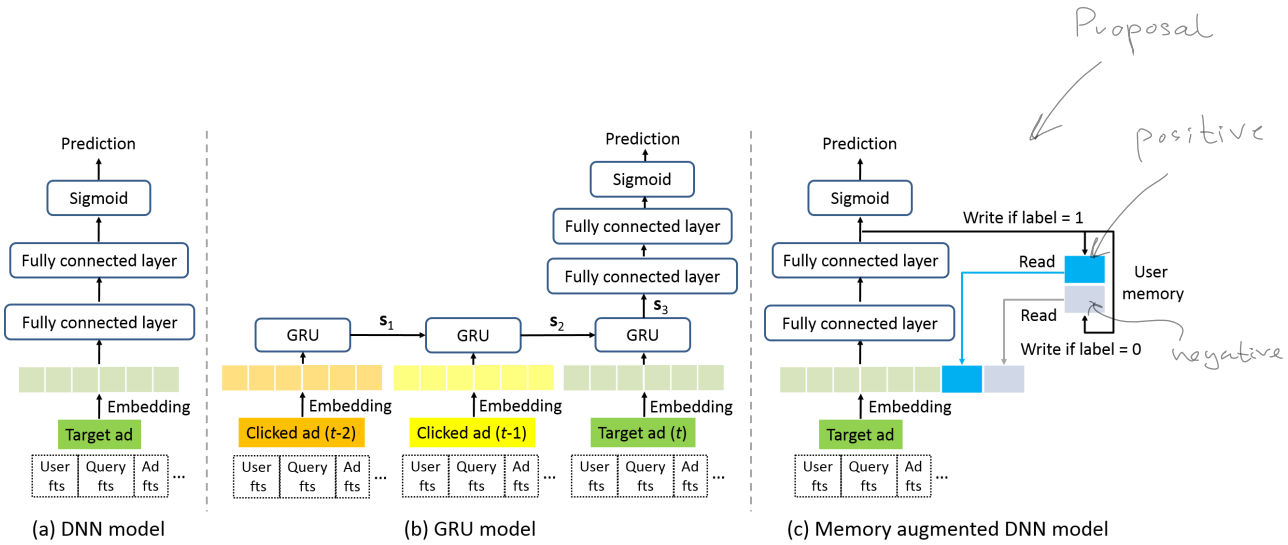

Figure 1: Structures of neural network models for CTR prediction.

Table 1: Each row is an instance for CTR prediction. The first column is the label (1 - clicked, 0 - unclicked). Each of the other columns is a field.

Label	User ID	User Age	Ad Title
1	2135147	24	Beijing flower delivery
0	3467291	31	Nike shoes, sporting shoes
0	1739086	45	Female clothing and jeans

example instances. Each instance can be described by multiple *fields* such as user information (User ID, City, etc.) and ad information (Creative ID, Title, etc.). The instantiation of a field is a *feature*.

2.2 Motivation

It is known that the DNN model is stateless. When DNN is used for CTR prediction, it cannot utilize useful information contained in past training instances. In contrast, the RNN model is stateful and it maintains a hidden state corresponding to each past training instance. It can thus achieve improved prediction performance. For example, Zhang et al. [19] use RNNs to model the dependency on users' sequential behaviors for CTR prediction.

Nevertheless, the application of RNNs to practical CTR prediction services is rather complex. We take GRU [3], one of the most advanced RNN cell structures, as an example. We analyze the complexity from two aspects: 1) model complexity and 2) data preparation complexity. We compare GRU with DNN.

2.2.1 Model Complexity. We illustrate the structures of DNN and GRU models for CTR prediction in Figure 1. The DNN model (Figure 1(a)) takes one ad instance as input. The model contains an embedding layer, several fully connected layers and an output layer (i.e., the sigmoid function). Parameters in each fully connected layer only contain a weight matrix and a bias vector.

The GRU model (Figure 1(b)) takes a sequence of ad instances as input. Each instance first goes through an embedding layer and the resulting embedding vector $\mathbf{x}$ then goes through a GRU cell to result in a hidden vector $\mathbf{s}$. In particular, $\mathbf{s}_t = GRU(\mathbf{x}_t, \mathbf{s}_{t-1})$, i.e., the tth GRU cell takes the current instance embedding $\mathbf{x}_t$ and the hidden vector $\mathbf{s}_{t-1}$ from the last GRU cell as input. The final hidden representation ($\mathbf{s}_3$ in Figure 1(b)) goes through several fully connected layers and an output layer to result in the predicted CTR.

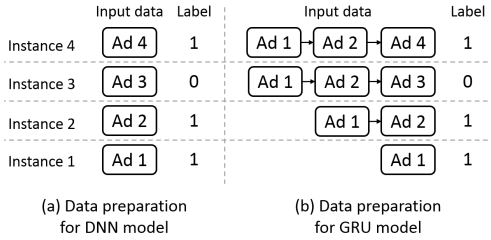

Figure 2: Illustration of data preparation.

Each GRU cell is defined as

$$\mathbf{z} = sigmoid(\mathbf{x}_t\mathbf{U}^z + \mathbf{s}_{t-1}\mathbf{W}^z),\ \mathbf{r} = sigmoid(\mathbf{x}_t\mathbf{U}^r + \mathbf{s}_{t-1}\mathbf{W}^r),$$

$$\mathbf{h} = tanh(\mathbf{x}_t\mathbf{U}^h + (\mathbf{s}_{t-1} \circ \mathbf{r})\mathbf{W}^h),\ \mathbf{s}_t = (1 - \mathbf{z}) \circ \mathbf{h} + \mathbf{z} \circ \mathbf{s}_{t-1},$$

where $\circ$ denotes element-wise product, $\mathbf{x}_t$ and $\mathbf{s}_{t-1}$ are the input, $\mathbf{s}_t$ is the output and others are all model parameters. We can clearly observe that GRU is much more complex than DNN. The complexity exists for both offline training and online prediction.

2.2.2 Data Preparation Complexity. We illustrate the data preparation complexity in Figure 2. It is observed in Figure 2(a) that DNN takes one ad instance as input. That is to say, it can directly use the log data where each ad impression is logged as a row. In Figure 2(b), GRU takes a sequence of ad instances as input. Therefore, we have to search and concatenate the log data of a given user to form sequences as GRU input. Moreover, it is observed that the input data for GRU have clear redundancy. For example, "Ad 1" is repeated 4 times in 4 training instances. Nevertheless, data redundancy is not a problem for Natural Language Processing (NLP) tasks with GRU (e.g., sentence classification). This is because words in a sentence naturally form a sequence and there is no need to repeat words.

2.3 Memory Augmented Deep Neural Network

Based on the above analysis, we wonder whether we can design a new model that is as simple as DNN, but has certain ability to exploit useful information contained in users' historical behaviors as RNN. To this end, we propose to augment the DNN model with an external user memory component. This component stores high-level information about a user's preferences. We illustrate the memory augmented DNN (MA-DNN) model structure in Figure 2(c).

2.3.1 *Model.* For a user u, we create two memory vectors $\mathbf{m}_{u1}$ and $\mathbf{m}_{u0}$, which memorize high-level information about what the user possibly likes and dislikes respectively.

First, each feature $f_i \in \mathbb{R}$ (e.g., a user ID) in an instance (whose CTR is to be predicted) goes through an embedding layer and is mapped to its embedding vector $\mathbf{e}_i \in \mathbb{R}^K$, where $\mathbf{e}_i$ is to be learned and K is the embedding dimension. We then concatenate the embedding vectors from all the features as a long input vector $\mathbf{x}$.

Given the user ID u, we then find from the external memory the corresponding $\mathbf{m}_{u1}$ and $\mathbf{m}_{u0}$, and concatenate them with $\mathbf{x}$ to form a vector $\mathbf{v} = [\mathbf{x}, \mathbf{m}_{u1}, \mathbf{m}_{u0}]$. Compared with DNN which takes $\mathbf{x}$ as input, the vector $\mathbf{v}$ not only contains information about current instance (i.e., $\mathbf{x}$), but also contains information about historical instances through the two memory vectors $\mathbf{m}_{u1}$ and $\mathbf{m}_{u0}$.

The vector $\mathbf{v}$ then goes through several fully connected (FC) layers with the ReLU activation function ($\text{ReLU}(x) = \max(0, x)$), in order to exploit high-order nonlinear feature interactions [8]. Nair and Hinton [12] show that ReLU has significant benefits over sigmoid and tanh activation functions in terms of the convergence rate and the quality of obtained results.

Formally, the FC layers are defined as follows:

$$\mathbf{z}_1 = \text{ReLU}(\mathbf{W}_1\mathbf{v} + \mathbf{b}_1),\ \mathbf{z}_2 = \text{ReLU}(\mathbf{W}_2\mathbf{z}_1 + \mathbf{b}_2),\ \cdots$$
$$\mathbf{z}_L = \text{ReLU}(\mathbf{W}_L\mathbf{z}_{L-1} + \mathbf{b}_L),$$

where L denotes the number of hidden layers; $\mathbf{W}_l$ and $\mathbf{b}_l$ denote the weight matrix and bias vector (to be learned) for the lth layer.

Finally, the output $\mathbf{z}_L$ of the last FC layer goes through a sigmoid function to generate the predicted CTR as

$$\hat{y} = \frac{1}{1 + \exp[-(\mathbf{w}^T\mathbf{z}_L + b)]},$$

where $\mathbf{w}$ and b are model parameters to be learned.

2.3.2 *Training.* We have two aims in the MA-DNN model: 1) we would like to make the predicted CTR as close to the true label as possible and 2) we would like the memory component to summarize what a user possibly likes and dislikes.

We achieve Aim 1 by minimizing the average logistic loss as

$$\text{loss}_1 = -\frac{1}{|\mathbb{Y}|}\sum_{y\in\mathbb{Y}}[y\log\hat{y} + (1-y)\log(1-\hat{y})], \quad (1)$$

where $y \in \{0, 1\}$ is the true label of the target instance corresponding to $\hat{y}$ and $\mathbb{Y}$ is the collection of labels.

We achieve Aim 2 by minimizing the mean square error of the memory component and the output $\mathbf{z}_L$ of the last FC layer as

$$\text{loss}_2 = \frac{1}{|\mathbb{Y}|}\sum_{y\in\mathbb{Y}}[y\mathbf{m}_{u1} + (1-y)\mathbf{m}_{u0} - \mathbf{z}_L]^2, \quad (2)$$

where the user u is identified by the user ID in the original input. As can be seen, if the true label $y = 1$, $\mathbf{z}_L$ is memorized by $\mathbf{m}_{u1}$ but not $\mathbf{m}_{u0}$; if the true label $y = 0$, $\mathbf{z}_L$ is memorized by $\mathbf{m}_{u0}$ but not $\mathbf{m}_{u1}$. That is to say, $\mathbf{m}_{u1}$ and $\mathbf{m}_{u0}$ do summarize high-level information about what a user possibly likes and dislikes respectively by design. We choose to memorize the output $\mathbf{z}_L$ of the last FC layer because it contains the highest-level abstraction.

Since Aim 2 is to update $\mathbf{m}_{u1}$ and $\mathbf{m}_{u0}$ according to $\mathbf{z}_L$, we should not change the value of $\mathbf{z}_L$. To do so, the gradient of loss_2 is only computed with respect to $\mathbf{m}_{u1}$ and $\mathbf{m}_{u0}$, but not $\mathbf{z}_L$. In this way, the value of $\mathbf{z}_L$ is updated only by loss_1.

The final loss function is given by $\text{loss} = \text{loss}_1 + \alpha\text{loss}_2$, where α is a tunable parameter.

Table 2: Statistics of experimental large-scale datasets.

Dataset	# Training	# Testing	# Fields	# Features
Avito	168,255,929	2,332,738	27	42,301,586
Company	62,727,007	1,406,954	38	42,554,432

3 EXPERIMENTS

Why not to use more popular data sets such as cited, etc ?

3.1 Datasets

Table 2 lists the statistics of two large-scale experimental datasets.

1) **Avito advertising dataset**[1]**.** This dataset contains a random sample of ad logs from avito.ru, the largest general classified website in Russia. We use the ad logs on 2015-05-20 for testing and others for training. No sampling is performed. The features used include 1) ad features such as ad ID, ad title and ad category, 2) user features such as user ID, IP ID, user agent and user device, 3) query features such as search query, search category and search parameters, and 4) context features such as day of week and hour of day. The number of test samples is over 2.3×10^6.

2) **Company advertising dataset.** This dataset contains a random sample of ad impression and click logs from a commercial advertising system in Alibaba. We use ad logs of 30 consecutive days in 2019 for training and logs of the next day for testing. The number of test samples is over 1.4×10^6.

2019. Where other sota models ?

3.2 Methods Compared

(1) **LR**. Logistic Regression [15]. It is a generalized linear model.
(2) **FM**. Factorization Machine [14]. It models both first-order feature importance and second-order feature interactions.
(3) **DNN**. Deep Neural Network. Its structure is in Figure 1(a).
(4) **W&D**. The Wide&Deep model in [2]. It combines a wide component (LR) and a deep component (DNN).
(5) **MA-DNN**. Memory Augmented Deep Neural Network model. Its structure is shown in Figure 1(c).
(6) **MA-W&D**. Memory Augmented Wide&Deep model.

3.3 Settings

Parameter Settings. The embedding dimension of each feature is set to $K = 10$. The number of FC layers in neural network-based models is set to 3, with dimensions 512, 256 and 64. The dimensions of memory vectors $\mathbf{m}_{u1}$ and $\mathbf{m}_{u0}$ for a user u are both set to 64. The batch size is set to 128 and the balancing parameter α is set to 1. No dropout is performed. Samples are fed to models in the time order (i.e., no data shuffling). All the methods are implemented in Tensorflow and optimized by Adagrad [5] for offline evaluation. We make the code publicly available[2]. Online implementation is based on a private system with a distributed MPI cluster.

Evaluation Metrics. 1) **AUC**: the Area Under the ROC Curve over the test set. The larger the better. 2) **Logloss**: the value of Eq. (1) over the test set. The smaller the better.

[1] https://www.kaggle.com/c/avito-context-ad-clicks/data
[2] https://github.com/rener1199/deep_memory

Table 3: Test AUC and Logloss on two large-scale datasets (no dropout is performed).

	Avito (full)		Company	
Model	AUC	Logloss	AUC	Logloss
LR	0.7374	0.04328	0.6251	0.2582
FM	0.7844	0.03990	0.6458	0.2490
DNN	0.7939	0.03954	0.6601	0.2379
W&D	0.7951	0.03948	0.6629	0.2355
MA-DNN	**0.7968**	**0.03933**	**0.6703**	**0.2331**
MA-W&D	**0.7976**	**0.03929**	**0.6710**	**0.2323**

3.4 Effectiveness

Table 3 lists the AUC and Logloss values of different models. It is observed that on both datasets, memory augmented models outperform all the baselines. Moreover, MA-DNN outperforms DNN and MA-W&D outperforms W&D. These results show that the memory component can indeed extract useful information from users' historical behaviors and improve the prediction performance.

3.5 Online A/B Test

We conducted online experiments in an A/B test framework over 6 days in April 2019. The benchmark model is W&D. Online CTR is defined as the number of clicks over the number of ad impressions. A larger online CTR indicates the enhanced effectiveness of a CTR prediction model. Online A/B test results in Table 4 show that MA-W&D outperforms W&D and results in an average increase of daily CTR of 2.56% and an average increase of daily CPM of 0.74%.

4 RELATED WORK

CTR prediction has attracted lots of attention from both academia and industry [2, 8, 9]. Generalized linear models, such as Logistic Regression (LR) [15] and Follow-The-Regularized-Leader (FTRL) [11], have shown decent performance in practice. However, a linear model lacks the ability to learn sophisticated feature interactions. Factorization Machines (FMs) [14] are proposed to model pairwise feature interactions and they show improved performance.

In recent years, Deep Neural Networks (DNNs) are exploited for CTR prediction and item recommendation in order to automatically learn feature representations and high-order feature interactions [4, 8, 17, 18]. Qu et al. [13] propose the Product-based Neural Network where a product layer is introduced between the embedding layer and the fully connected layer. Cheng et al. [2] propose Wide&Deep, which combines LR and DNN to improve both the memorization and generalization abilities of the model. Guo et al. [7] propose DeepFM, which models low-order feature interactions like FM and models high-order feature interactions like DNN. To capture dependency on users' sequential behaviors, Zhang et al. [19] propose Recurrent Neural Network (RNN) based models for CTR prediction. Nevertheless, the application of RNNs to practical CTR prediction services is rather complex.

In this paper, we propose Memory Augmented Deep Neural Network (MA-DNN) for CTR prediction. The proposed MA-DNN achieves a good compromise between DNN and RNN. We are aware of recent work like [1] that also utilizes memory networks [6].

Table 4: Online A/B test results.

Model	CTR gain	CPM gain
W&D	0%	0%
MA-W&D	+2.56%	+0.74%

However, [1] is proposed for recommender systems and our way of designing the user memory component is different from that in [1].

5 CONCLUSION

In this paper, we propose Memory Augmented DNN (MA-DNN) for CTR prediction in advertising systems. In particular, we create two external memory vectors for each user, memorizing high-level abstractions of what a user possibly likes and dislikes. The proposed MA-DNN achieves a good compromise between DNN and RNN. Actually, the memory component can be augmented to other models as well (e.g., the Wide&Deep model). Both offline and online experiments demonstrate the effectiveness of memory augmented neural networks for practical CTR prediction services.

REFERENCES

[1] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *WSDM*. ACM, 108–116.

[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 *DLRS*. ACM, 7–10.

[3] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *RecSys*. ACM, 191–198.

[5] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[6] Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401* (2014).

[7] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. Deepfm: a factorization-machine based neural network for ctr prediction. In *IJCAI*. 1725–1731.

[8] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *SIGIR*. ACM, 355–364.

[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 *ADKDD*. ACM, 1–9.

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

[11] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. 2013. Ad click prediction: a view from the trenches. In *KDD*. ACM, 1222–1230.

[12] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*. 807–814.

[13] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *ICDM*. IEEE, 1149–1154.

[14] Steffen Rendle. 2010. Factorization machines. In *ICDM*. IEEE, 995–1000.

[15] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *WWW*. ACM, 521–530.

[16] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *DLRS*. ACM, 17–22.

[17] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *ADKDD*. ACM, 12.

[18] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep learning over multi-field categorical data. In *ECIR*. Springer, 45–57.

[19] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. 2014. Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks.. In *AAAI*, Vol. 14. 1369–1375.

[20] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *KDD*. ACM, 1059–1068.