PAL: a position-bias aware learning framework for CTR prediction in live recommender systems

Conference Paper · September 2019

DOI: 10.1145/3298689.3347033

5 authors, including:

Huifeng Guo
Huawei Technologies
37 PUBLICATIONS **1,177** CITATIONS
SEE PROFILE

Ruiming Tang
Huawei Technologies
82 PUBLICATIONS **1,296** CITATIONS
SEE PROFILE

PAL: A Position-bias Aware Learning Framework for CTR Prediction in Live Recommender Systems

Huifeng Guo, Jinkai Yu, Qing Liu, Ruiming Tang*, Yuzhou Zhang
* Corresponding author.
Noah's Ark Lab, Huawei, China.
{huifeng.guo,yujinkai,liuqing48,tangruiming,zhangyuzhou3}@huawei.com

ABSTRACT

Predicting Click-Through Rate (CTR) accurately is crucial in recommender systems. In general, a CTR model is trained based on user feedback which is collected from traffic logs. However, position-bias exists in user feedback because a user clicks on an item may not only because she favors it but also because it is in a good position. One way is to model position as a feature in the training data, which is widely used in industrial applications due to its simplicity. Specifically, a default position value has to be used to predict CTR in online inference since the actual position information is not available at that time. However, using different default position values may result in completely different recommendation results. As a result, this approach leads to sub-optimal online performance. To address this problem, in this paper, we propose a **P**osition-bias **A**ware **L**earning framework (PAL) for CTR prediction in a live recommender system. It is able to model the position-bias in offline training and conduct online inference without position information. Extensive online experiments are conducted to demonstrate that PAL outperforms the baselines by 3% - 35% in terms of CTR and CVR (**C**on**V**ersion **R**ate) in a three-week AB test.

ACM Reference Format:
Huifeng Guo, Jinkai Yu, Qing Liu, Ruiming Tang*, Yuzhou Zhang . 2019. PAL: A Position-bias Aware Learning Framework for CTR Prediction in Live Recommender Systems. In *Thirteenth ACM Conference on Recommender Systems (RecSys '19), September 16–20, 2019, Copenhagen, Denmark*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3298689.3347033

1 INTRODUCTION

Click-Through Rate (CTR) prediction is to estimate the probability that a user will click on a recommended item under a specific context. It plays a crucial role in recommender systems, especially in the industry of App Store and online advertising [2, 5, 7–9, 12, 18]. To maximize revenue and user satisfaction, the recommended items are presented in descending order of the scores computed by a function of the predicted CTR and bid [10, 19], i.e., $f(CTR, bid)$, where "bid" is benefit that the system receives if the item is clicked by a user. Therefore, the accuracy of the predicted CTR directly determines the revenue and user experience [16].

RecSys '19, September 16–20, 2019, Copenhagen, Denmark

ACM ISBN 978-1-4503-6243-6/19/09…$15.00
https://doi.org/10.1145/3298689.3347033

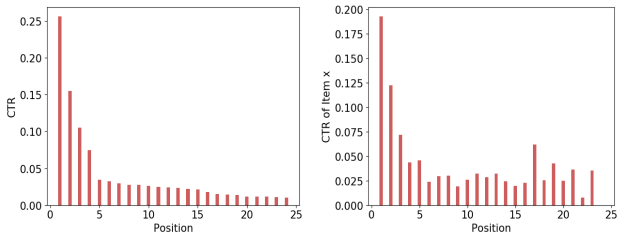


Figure 1: CTRs at different positions

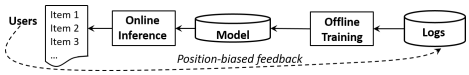


Figure 2: Workflow of Recommendation

The recommender systems in a real production environment usually consists of two important procedures [2, 6, 19]: *offline training* and *online inference*, as shown in Figure 2. In offline training, a CTR prediction model is trained based on the user-item interaction information (as well as user and item information), collected from traffic logs of recommender systems [3]. In online inference, the trained model is deployed to the live recommender systems to predict the CTR and then make recommendations.

One problem of these procedures is that the user-item interaction is affected by the positions that the items are displayed. As discussed in [14], the CTR declines rapidly with the display position. Similarly, we also observe such exogenous phenomena in a mainstream App Store. As shown in Figure 1, either for App Store (Figure 1(a)) or a specific App (Figure 1(b), an app is an item of App Store), we observe that the normalized CTR drops dramatically with position.

These observations imply that a user clicks on an item may not only because she favors it but also it is in a good position, so that the training data collected from traffic logs contains positional bias. We denote it as *position-bias* through this paper. As an important factor to CTR signal, it is necessary to model the position-bias into the CTR prediction model in offline training [10, 15].

Although various click models are proposed to model the position-bias in the training data [1, 3], there is limited research on the realistic issue that the position information is unavailable in online inference. One practical approach is inverse propensity weighting [15], in which a user-defined transformation on the position information is applied and then the transformed value is fixed. However, as mentioned in [10], it is hard to design a good transformation manually for the position information, which leads to worse performance than a automatically-learned transformation. Therefore, the

authors of [10] propose to model position as a feature in the training data, which is widely used in industrial applications due to its simplicity. Specifically, a default position value has to be used to predict CTR in online inference since the actual position information is not available at that time. Unfortunately, using different default position values may result in completely different recommendation results which leads to sub-optimal online performance.

In this paper, we propose a **P**osition-bias **A**ware **L**earning framework (PAL) to model the position-bias in offline training and to conduct online inference without position information. The idea of PAL is based on the assumption that the probability of an item is clicked by a user depends on two factors: a) the probability that the item is seen by the user and b) the probability that the user clicks on the item, given that the item has been seen by the user. Each factor is modeled as a module in PAL and the product of the outputs of these two modules is the probability that an item is clicked by a user. If the two modules are optimized separately, it may lead the overall system to a sub-optimal status, because of the inconsistency between training objectives of the two modules, as claimed in [18]. To avoid such limitation and facilitate better CTR prediction performance, the two modules in PAL are optimized jointly and simultaneously. Once these two modules are well trained through offline training, the second module, i.e., the probability that the user clicks on the item, given that the item has been seen by the user, is deployed to predict CTR in online inference.

We conduct online experiment and user case analysis to demonstrate the superiority of PAL over competitive methods. The results show that PAL improves CTR and CVR over the baseline methods by 3% - 35%, across three-week AB test.

2 METHOD

2.1 Notation

We assume the offline click dataset to be $S = \{(\mathbf{x_i}, pos_i \rightarrow y_i)\}|_{i=1}^{N}$, where N is the total number of samples, $\mathbf{x_i}$ is the feature vector of sample i which includes user profile, item features and context information, pos_i is the position information of sample i, y_i is the user feedback ($y_i = 1$ if user clicks on this item, $y_i = 0$, otherwise). We use $\mathbf{x}$, pos and y to denote the feature vector, the position information and the label of a data sample in general.

2.2 Preliminary

There are two directions to model the position-bias in offline training, namely **as a feature** and **as a module**.

As a feature: This approach models the position information as a feature. In offline training, the input feature vector of the CTR model is the concatenation of $\mathbf{x}$ and pos, i.e., $\hat{\mathbf{x}} = [\mathbf{x}, pos]$. A CTR prediction model is trained based on the concatenated feature vector.

As the position information is modeled as a feature in the offline training, a feature representing "position" should also be included in the online inference[1], as shown in the right part of Figure 3. However, position information is unavailable when online inference is performed. One way to resolve the problem of lacking position information for inference is to decide, for each position, the most suitable item, sequentially from the top-most position to bottom-most position. As can be analyzed, it is a brute-force method with $O(lnT)$ time complexity (where l is the length of ranking list, n is the number of candidate items, and T is the latency for one inference), which is unacceptable in a low-latency online environment.

To shorten the response latency, an alternative method with $O(nT)$ time complexity, as presented in [10], is to select a position for all the items as the value of the position feature, i.e., *position value* in short. However, different position values may result in completely different recommendation results[2]. So we have to find a proper position value to achieve good online performance. There are two ways to compare the performance of performing inference with different position values: online experiment and offline evaluation. The former is better but very expensive for a live recommender system. Therefore, we have to adopt offline evaluation to select the suitable position value. Moreover, no matter online experiment or offline evaluation is utilized to select position value, it is not of good generalization, as the position value for online inference in one application scenario may not be suitable for another scenario.

As a module: To address the above limitations of taking position information as a feature, we propose a novel framework to take position information as a module, so that we can model the position-bias in offline training and conduct online inference without position information. We present our framework in the next section.

2.3 Our Framework

Our framework is motivated based on the assumption that an item is clicked by a user only if it has been seen by her. More specifically, we consider the probability that an item is clicked by a user to be dependent on two factors: a) the probability that the item is seen by a user; b) the probability that the user clicks on the item, given that the item has been seen by the user [14] as shown in Eq. (1).

$$p(y = 1|\mathbf{x}, pos) = p(seen|\mathbf{x}, pos)p(y = 1|\mathbf{x}, pos, seen). \quad (1)$$

Eq. (1) is simplified to Eq. (2) if we further assume: a) the probability that an item has been seen only relates to the probability that the associated position has been observed; b) the probability that an item is clicked is independent of its position if it has been seen.

$$p(y = 1|\mathbf{x}, pos) = p(seen|pos)p(y = 1|\mathbf{x}, seen). \quad (2)$$

As shown in the left part of Figure 3, our proposed framework PAL is designed based on Eq. (2) and consists of two modules. The first module models the probability $p(seen|pos)$ which we denote as "ProbSeen" in Figure 3 and takes the position information pos as the input. The second module models the probability $p(y = 1|\mathbf{x}, seen)$ which we denote as "pCTR" in Figure 3, meaning the predicted CTR by the model. Its input is feature vector $\mathbf{x}$ in the training data. Any CTR prediction models, such as linear models and deep learning models [5, 13, 16], can be applied for these two modules. Then, the learned CTR as denoted "bCTR" in Figure 3 that considers the position bias in offline training is the multiplication of the outputs of these two modules. As mentioned in [18], if the two modules are optimized separately, the inconsistency between different training objectives leads the overall system to a sub-optimal status. To avoid such sub-optimal performance, we optimize these two modules in

[1] Assigning "null" to the position feature usually results in unreliable inference results.

[2] Significantly different recommendation results are observed in our experiments.

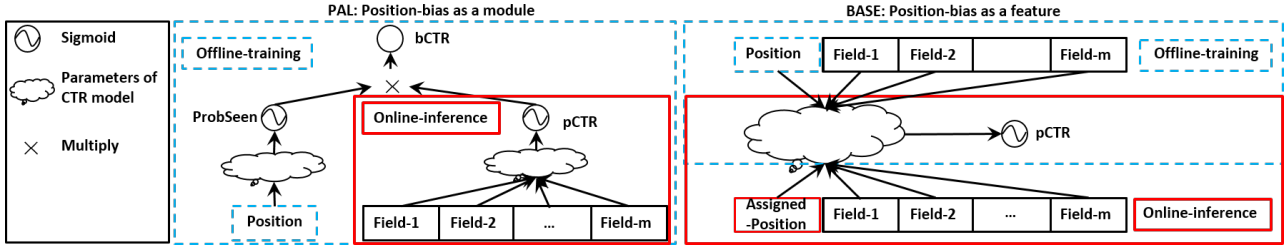


Figure 3: Framework of PAL V.S. BASE.

our framework jointly and simultaneously. To be more specific, the loss function of PAL is defined as:

$$L(\theta_{ps}, \theta_{pCTR}) = \frac{1}{N}\sum_{i=1}^{N} l(y_i, bCTR_i)) = \frac{1}{N}\sum_{i=1}^{N} l(y_i, ProbSeen_i \times pCTR_i)), \quad (3)$$

where θ_{ps} and θ_{pCTR} are the parameters of ProbSeen and pCTR modules, $l(\cdot)$ is cross-entropy loss function. The pCTR module, used in online inference procedure, is not directly optimized. In fact, when the logloss between label and predicted bCTR is minimized, the parameters of ProbSeen and pCTR modules are optimized as Eq. (4) and Eq. (5) (η is the learning rate) by stochastic gradient descent (SGD), such that the influence of position-bias and user preference is learned implicitly and respectively.

$$\theta_{ps} = \theta_{ps} - \eta \cdot \frac{1}{N}\sum_{i=1}^{N}(bCTR_i - y_i) \cdot pCTR_i \cdot \frac{\partial ProbSeen_i}{\partial \theta_{ps}}. \quad (4)$$

$$\theta_{pCTR} = \theta_{pCTR} - \eta \cdot \frac{1}{N}\sum_{i=1}^{N}(bCTR_i - y_i) \cdot ProbSeen_i \cdot \frac{\partial pCTR_i}{\partial \theta_{pCTR}}. \quad (5)$$

In the offline training procedure, similar with [5, 13, 16] and other related works, the early stop strategy is used to obtain well-trained model in training procedure. Once PAL is well trained, the module pCTR is deployed online for CTR inference. As can be easily observed, position is not required in the pCTR module in PAL, so that we do not need to assign position values to the items in online inference as the "as a feature" does.

3 ONLINE EXPERIMENTS

We design online experiments in a live recommender system to verify the performance of PAL. Specifically, we conduct a three-week AB test to validate the superiority of PAL over the "as a feature" baseline methods. The AB test is conducted in game recommendation scenario in the game center of Company X's App Store.

3.1 Datasets

In the production environment of Company X's App Store, we sample around 1 billion records from traffic logs as our offline training dataset. To update our model, the training dataset is refreshed in a sliding time-window style. The features for training consist of app features (e.g., app identification, category and etc), user features (e.g., downloaded, click history and etc), and context features (e.g., operation time and etc).

3.2 Baseline

The baseline framework refers to "as a feature" strategy in Section 2.2. Actually, this baseline is the method adopted in [10]. As stated, we need to select a proper position value for online inference. However, due to resource limit, it is impossible to evaluate the baseline framework with all possible positions. Therefore, we conduct an offline experiment to select proper positions.

Settings. To select proper position(s), we collect two datasets from two business scenarios in Company X's App Store. The test dataset is collected from the next day's traffic logs. We apply different position values in the test dataset, ranging from position 1 to position 10. Similar to the related works [5, 11, 13, 16], AUC and LogLoss are adopted as the metrics to evaluate the offline performance of the cases with different assigned position values.

Results and analysis. The results of the offline experiments are presented in Figure 5, where BASE_pk is the baseline framework with position value k assigned to all the items in the test data. PAL is our proposed framework of which the test data is collected without position values. From Figure 5, we can see that AUC and Logloss values are varying as we assign different position values to the test data. In addition, BASE_p9 achieves the highest AUC, BASE_p5 achieves the lowest LogLoss, and BASE_p1 achieves the worst performance in both AUC and LogLoss. We select two best ones (namely, BASE_p5 and BASE_p9) and the worse one (namely, BASE_p1) as baselines to conduct online AB test with PAL. It is worthy to notice that PAL does not achieve the best performance in this offline experiment in terms of either AUC or LogLoss.

3.3 AB test

Settings. For the control group, 2% of users are randomly selected and presented with recommendation generated by the baseline framework. For the experimental group, 2% of users are presented with recommendation generated by PAL. The model used in baseline and PAL is DeepFM [5] with the same network structure and the same set of features. Due to resource limit, we do not deploy the three baselines (namely, BASE_p1, BASE_p5 and BASE_p9) online at the same period of time. Instead, they are deployed one by one, each for one week, to compare with PAL. More specifically, we compare PAL v.s. BASE_p1, PAL v.s. BASE_p5 and PAL v.s. BASE_p9 for the first, second, and third week, respectively.

Metrics. We adopt two metrics to compare the online performance of PAL and the baselines, namely realistic Click Through Rate: $rCTR = \frac{\#downloads}{\#impressions}$ and realistic Conversion Rate: $rCVR = \frac{\#downloads}{\#users}$, where $\#downloads$, $\#impressions$ and $\#users$ are the number of downloads, impressions and visited users in the day of AB test, respectively. Different from the predicted CTR, i.e., "pCTR" in Figure 3, "rCTR" is the realistic CTR we observe online.

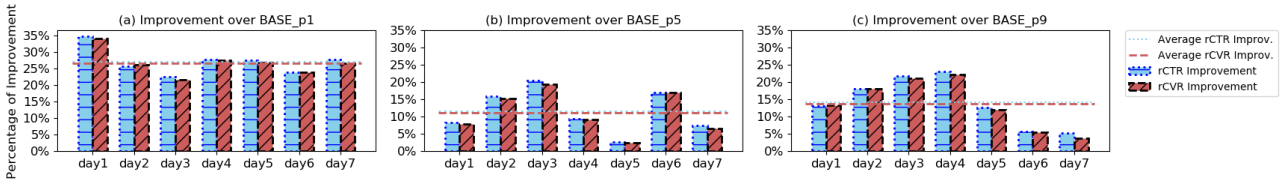
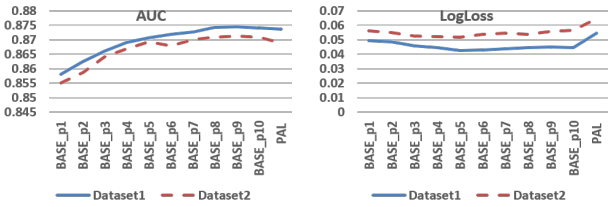
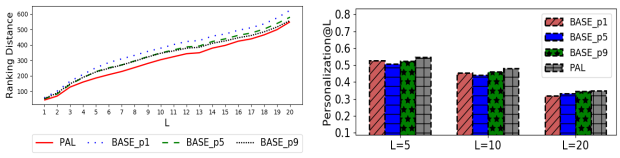


Figure 4: Results of Online AB Test.



(a) (b)

Figure 5: Offline Experimental Results



(a) Ranking Distance (b) Results of Personalization

Figure 6: Online Analysis

Results. Figure 4 presents the results of the online experiments. The blue and red histograms show rCTR and rCVR improvement of PAL over the baselines. Firstly, the metrics of rCTR and rCVR are both consistently improved across the entire three-week AB test, which validates the superiority of PAL over the baselines. Secondly, we also observe that the average improvement of both rCTR and rCVR (the dash lines in Figure 4) in the first week (where the baseline is BASE_p1) is the highest and it is the lowest in the second week (where the baseline is BASE_p5). This phenomenon tells us that the performance of the baseline varies significantly by assigning different position values, because the recommendations may be completely different with different position values assigned.

3.4 Online Analysis

To fully understand the result of AB test and verify whether our proposed framework eliminates position bias in online inference, we analyze the recommendations generated by PAL and the baselines.

The **first** experiment is to compare the **ranking distance to ground truth ranking**. We define the *ground truth ranking* as the list of items that are ranked by descending order of $f(rCTR, bid)$ value. Spearman's Footrule [4] is adopted to measure the displacement of the items in two rankings, which is widely used to measure the distance between two rankings. We define the distance between the ground truth ranking and a ranking σ_M generated by either PAL or baselines at top-L, as:

$$D(\sigma_M, L) = \frac{1}{|U|} \sum_{u \in U} \left(\sum_{i=1}^{L} |i - \sigma_{M,u}(i)| \right) \tag{6}$$

where u is a user in the user group U with popularity $|U|$, $\sigma_{M,u}$ is the recommendation list to user u by model M. The i^{th} item in the ground truth ranking is at position $\sigma_{M,u}(i)$ in the recommendation $\sigma_{M,u}$. We compare $D(\sigma_M, L)$ for $M \in$ {PAL, BASE_p1, BASE_p5, BASE_p9} and $L \in [1, 20]$, as displayed in Figure 6(a), where the solid red line is the result of PAL and the other lines are the results of the baselines. We can see that PAL yields the shortest distance to the ground truth ranking, which means that the recommendation generated by PAL is most similar to the real ranking we observed online. This is achieved by PAL modeling position-bias wisely in offline training and eliminating position-bias in online inference, which explains why PAL outperforms the baselines consistently and significantly.

The **second** experiment is to compare the **personalization** between PAL and the baselines. Personalization@L [6, 17] can measure the inter-user diversity, an important factor of the recommendation results, of top-L in a ranking across different users. The Personalization@L is defined in Eq. (7):

$$Personalization@L = \frac{1}{|U| \times (|U|-1)} \sum_{a \in U} \sum_{b \in U} \left(1 - \frac{q_{ab}(L)}{L}\right) \tag{7}$$

where $|U|$ is the size of the user group U, $q_{ab}(L)$ is the number of common items in the top-L of both user a and user b. A higher Personalization@L means more diverse recommended items in the top-L positions across different users.

We compute the personalization@L of PAL and the baselines (namely, BASE_p1, BASE_p5, BASE_p9), respectively. Figure 6(b) presents the personalization at top-5 ($L = 5$), top-10 ($L = 10$) and top-20 ($L = 20$) of the recommendations by different frameworks. We can see that PAL yields the highest level of personalization, which demonstrates that the top items in the recommendation generated by PAL are more diverse than that generated by the baselines, because PAL is able to capture specific interests of different users better and recommend items according to users' personal interest after eliminating position-bias affect.

4 CONCLUSION

In this paper, we propose a framework PAL that can model the position-bias in the training data in offline training and predict CTR without position information in online inference. Compared to the baselines, PAL yeilds better results in a three-week online AB test. Extensive online experimental results verify the effectiveness of our proposed framework.

REFERENCES

[1] Ye Chen and Tak W. Yan. 2012. Position-normalized click prediction in search advertising. In *KDD*. ACM, 795–803.

[2] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, et al. 2016. Wide & Deep Learning for Recommender Systems. In *DLRS@RecSys*. ACM, 7–10.

[3] Nick Craswell, Onno Zoeter, Michael J. Taylor, and Bill Ramsey. 2008. An experimental comparison of click position-bias models. In *WSDM*. 87–94.

[4] Persi Diaconis and Ronald L Graham. 1977. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society. Series B (Methodological)* (1977), 262–268.

[5] 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.

[6] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, Xiuqiang He, and Zhenhua Dong. 2018. DeepFM: An End-to-End Wide & Deep Learning Framework for CTR Prediction. *CoRR* (2018).

[7] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, and Stuart Bowers. 2014. Practical Lessons from Predicting Clicks on Ads at Facebook. In *Eighth International Workshop on Data Mining for Online Advertising*. 1–9.

[8] Kuang Chih Lee, Burkay Orten, Ali Dasdan, and Wentong Li. 2012. Estimating conversion rate in display advertising from past performance data. In *SIGKDD*. 768–776.

[9] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. *arXiv preprint arXiv:1803.05170* (2018).

[10] Xiaoliang Ling, Weiwei Deng, Chen Gu, Hucheng Zhou, Cui Li, and Feng Sun. 2017. Model Ensemble for Click Prediction in Bing Search Ads. In *Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia, April 3-7, 2017*. ACM, 689–698.

[11] Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, and Yuzhou Zhang. 2019. Feature Generation by Convolutional Neural Network for Click-Through Rate Prediction. In *The World Wide Web Conference, San Francisco, CA, USA, May 13-17*. ACM, 1119–1129.

[12] H. Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. 2013. Ad click prediction: a view from the trenches. In *ACM SIGKDD*. https://doi.org/10.1145/2487575.2488200

[13] Yanru Qu, Bohui Fang, Weinan Zhang, Ruiming Tang, Minzhe Niu, Huifeng Guo, Yong Yu, and Xiuqiang He. 2019. Product-based Neural Networks for User Response Prediction over Multi-field Categorical Data. *ACM Trans. Inf. Syst.* 37, 1 (2019), 5:1–5:35.

[14] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting Clicks: Estimating the Click-through Rate for New Ads. In *WWW*. 521–530.

[15] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to Rank with Selection Bias in Personal Search. In *SIGIR*. ACM, 115–124.

[16] 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 *SIGKDD*. ACM, 1059–1068.

[17] Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling, and Yi-Cheng Zhang. 2010. Solving the apparent diversity-accuracy dilemma of recommender systems. *PNAS* 107, 10 (2010), 4511–4515.

[18] Han Zhu, Daqing Chang, Ziru Xu, Pengye Zhang, Xiang Li, Jie He, Han Li, Jian Xu, and Kun Gai. 2019. Joint Optimization of Tree-based Index and Deep Model for Recommender Systems. *CoRR* abs/1902.07565 (2019).

[19] Han Zhu, Junqi Jin, Chang Tan, Fei Pan, Yifan Zeng, Han Li, and Kun Gai. 2017. Optimized Cost per Click in Taobao Display Advertising. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. ACM, 2191–2200.