Analysis, Modeling, and Implementation of Publisher-side Ad Request Filtering

Liang Lv[1], Ke Xu[1,2], Haiyang Wang[3], Meng Shen[4], Yi Zhao[1], Minghui Li[5], Guanhui Geng[5] and Zhichao Liu[5]
[1]Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
[2]Beijing National Research Center for Information Science and Technology (BNRist), Beijing 100084, China
[3]Department of Computer Science and Engineering, University of Minnesota at Duluth, the United States
[4]School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China
[5]Baidu Inc., Beijing 100193, China
{lvl16@mails.,xuke@}tsinghua.edu.cn, haiyang@d.umn.edu, shenmeng@bit.edu.cn, zhaoyi16@mails.tsinghua.edu.cn,
{liminghui, gengguanhui, liuzhichao04}@baidu.com

***Abstract*—Online advertising has been a great driving force for the Internet industry. To maintain a steady growth of advertising revenue, advertisement (ad) publishers have made great efforts to increase the impressions as well as the conversion rate. However, we notice that the results of these efforts are not as good as expected. In detail, to show more ads to the consumers, publishers have to waste a significant amount of server resources to process the ad requests that do not result in consumers' clicks. On the other hand, the increasing ads are also impacting the browsing experience of the consumers.**

In this paper, we explore the opportunity to improve publishers' overall utility by handling a selective number of requests on ad servers. Particularly, we propose a publisher-side proactive ad request filtration solution *Win2*. Upon receiving an ad request, *Win2* estimates the probability that the consumer will click if serving it. The ad request will be served if the clicking probability is above a dynamic threshold. Otherwise, it will be filtered to reduce the publisher's resource cost and improve consumer experience. We implement *Win2* in a large-scale ad serving system and the evaluation results confirm its effectiveness.

***Index Terms*—Online Advertising, Publisher-side, Ad Request Filtering, Utility Optimization**

I. Introduction

Online advertising is the primary revenue model of Internet companies [1]. A typical ad serving procedure is shown in Figure 1. When a consumer uses an app or browses a webpage, the integrated ad unit[1] sends an ad request to the publisher's ad servers. This request carries such information as the ad unit ID and the consumer ID to identify itself. The ad servers take the request and query the data management platform for the consumer profile, e.g., demographic characteristics and browsing history. After that, the ad servers forward the ad request as well as the consumer profile to the ad exchange platform, where advertisers bid through a real-time bidding mechanism. The ad exchange platform determines the winner to maximize the expected revenue by considering both expected Click Through Rate (CTR) [2] and the bid. The winning

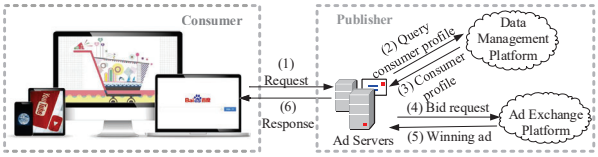


Fig. 1: The flow chart of an ad serving procedure.

advertisement is then returned and presented in the ad slot[2] on the consumer's device. The publisher usually makes money when the consumer takes a certain action, e.g., clicks on the advertisement.

Given the critical role online advertising plays, the publishers have made efforts to improve the advertising revenue by increasing the impressions [3] and the conversion rate [4]. To show more ads to more consumers with a reasonable latency, the publishers have to deploy large-scale Internet Data Centers (IDCs) to improve their service capabilities [5]. In addition, the publishers have developed intricate CTR prediction solutions [6]–[10] to improve the conversion rate by displaying ads that the consumers are interested in. The above measures, which are expected to maintain a steady growth of the advertising revenue, make the processing of ad requests on the publisher side especially expensive.

However, these measures do not achieve the desired results in real-world ad serving systems. A study from Yahoo! [11] shows that the CTRs of many ads are below 0.1%. Also, as given in Table I, *beneficial requests*[3] account for less than 0.5% on multiple platforms in Baidu's ad serving system. In other words, although a huge amount of ads are display every day, only a small portion of them are preferred and clicked by the consumers across all industries. It indicates

[1]An ad unit is a piece of code that integrated into the apps and webpages. It requests ads of specific sizes from the publisher and displays the returned ads at a specific position.

[2]Ad slots are the spaces in the apps or webpages where the ads are shown.

[3]We define *beneficial requests* as the ad requests that the returned ads are clicked by the consumers, and define *non-beneficial requests* as those do not result in clicks. Given that Cost-Per-Click (CPC) [12] is the dominant pricing model for online advertising, beneficial requests can bring revenue to the publishers and display ads that the consumers are interested in.

that the publishers have wasted a significant amount of IDCs resource to process the non-beneficial requests. To make the matter worse, increasing online ads have caused negative effects on the consumers' browsing experience. More and more consumers feel online ads annoying and intrusive, and apply adblockers [13], [14] to minimize annoyance.

TABLE I: Beneficial request ratio of representative ad units.

Ad unit	Platform	Beneficial request ratio
i	Android app	0.316%
ii	iOS app	0.306%
iii	Android WAP	0.454%
iv	iOS WAP	0.215%
v	WWW	0.074%

We attribute the above phenomenon to the fact that the existing ad serving systems prefer to handle all the incoming ad requests. The ad serving system is like a hard-working employee who has to distribute flyers to all random strangers who pass him/her on the street. Such a preference is based on the assumption that all the consumers are interested in some ads on the publishers' ad servers. However, this is not always the case. Doing a selective distribution is obviously more efficient.

In this paper, we aim to answer the following question: **Is there an effective approach to enhance the publishers' advertising income without harming consumers' browsing experience?** Rather than directly increasing the advertising revenue, we attempt to improve the publisher's income by reducing its resource cost. We for the first time explore the potential of serving the ad requests selectively and propose a publisher-side proactive ad request filtration solution, *Win2*. In detail, *Win2* predicts the clicking probability of each arrival ad request. If the clicking probability is larger than a dynamic threshold, the ad request is passed to the ad serving system where appropriate ads are matched and returned to the consumer; otherwise, the ad request is filtered and no advertisement will be returned. Intuitively, the performance of *Win2* heavily relies on the value of the threshold. To compute the optimal value, we formulate the threshold computation problem and design a dynamic threshold computation mechanism. Moreover, we also share the implementation of *Win2* on Baidu's large-scale ad serving system with extensive evaluation results.

It should be mentioned that, an alternative opportunity to show less ads that are not of interest to consumers is before the Step (5) in Figure 1: if the predicted CTRs of all the candidate ads are below a certain threshold, no advertisement will be returned and displayed. However, we notice that CTR prediction is almost at the end of the ad serving procedure, hence just not returning ads after CTR prediction is insufficient to lower the publishers' resource cost. *Win2* works before serving an ad request and attempts to serve less non-beneficial requests. Therefore, *Win2* can reduce the publishers' resource cost effectively and thus increase the advertising income.

The paper is structured as follows. Section II surveys the related work. Before proposing the design of *Win2*, we analyze a strawman approach in Section III to inspire our solution. Section IV introduces the architecture of *Win2*. Section V formulates the threshold computation problem and proposes the dynamic computation mechanism. The implementation of *Win2* is detailed in Section VI. We evaluate *Win2* in Section VII. Section VIII discusses the applicability of *Win2* in multiple pricing models. Section IX concludes the paper.

II. Related Work

AdBlock. The adblocker consumer market is growing steadily in recent years and there were 600 million devices running adblocking softwares globally as of 2016 [15]. The earliest adblocker was developed by Aasted Sorensen in 2002 [16]. The traditional adblockers are mostly utilizing manually curated rules to identify and block ads [17], [18]. To reduce the configuration overhead, the work from Kushmerick et al. [19] adopts machine-learning approaches in adblockers. Since then, an increasing number of machine-learning-based adblockers are proposed to detect the ads patterns in HTTP requests [20], HTML DOM [21] and JavaScript code [22].

It is known that the adblockers can improve the consumers' browsing experience and enhance their web engagement [23]. However, adblocking significantly reduces the publishers' revenue that could have been generated if the ads were displayed and clicked [24]. Based on the research from Adback [25], a single publisher's annual revenue loss due to adblockers can reach over one billion dollars. As a counter measure, publishers resort to anti-adblockers to detect and block the adblockers [26], [27], which unfortunately further motivate the birth of anti-"anti-adblockers" to block and fool anti-adblockers [28]. The arm race between the publishers and the consumers is a war that leaves no winners [29], [30].

Different from adblocking, the implementation of *Win2* is on the publisher side. Instead of blocking all the ads indiscriminately, *Win2* is designed to smartly recognize and filter the non-beneficial ad requests. As a result, *Win2* not only improves the consumers' browsing experience, but also reduces the overheads of the ad serving system without impacting advertising revenue.

Click Through Rate (CTR) Prediction. Nowadays, more and more publishers are aiming to attract the consumers' interests by pushing relevant ads to them. For example, the leading publishers such as Google [6], Facebook [7], Bing [8], Yahoo! [9] and Twitter [10] all highly rely on their CTR prediction systems for ads delivery. Recent years, it also attracts increasing attention in academia. The related studies develop a variety of machine-learning models for CTR prediction, including linear classification [31], collaborative filtering [32], tensor factorization [33], deep neural network [34], convolutional neural network [35], recurrent neural network [36], factorization machine [37], adversarial networks [38] and transfer learning [39], to name just a few. Besides, there are also studies on feature engineering [40], cold start [41] and feedback delay [42] in CTR prediction.

Win2 is orthogonal to CTR prediction. CTR prediction is a part of ad serving process, while *Win2* works before serving an arrival ad request. Moreover, CTR prediction solutions try to enhance advertising revenue by recommending relevant ads to the consumers, while *Win2* attempts to lower resource cost by serving less non-beneficial requests.

III. Strawman Solution and Inspirations

In this section, we start our discussion with the analysis a strawman solution for non-beneficial request filtration and explore its limitations to inspire our solution.

An intuitive approach to detect non-beneficial requests is to train a binary probabilistic classifier to predict the clicking probability of the ad requests, and the ad requests whose clicking probability are below the discrimination threshold[4] are classified as non-beneficial requests. To understand the basic performance of this intuitive approach, we adopt the XGBoost [43] classifier as a strawman solution, and validate its performance with the trace data of the ad serving system of Baidu. In the classification, the discrimination threshold of the classifier is set to its default value, i.e., 0.5.

Table II lists the datasets used in evaluation. *Trainset* contains 3 millions of ad requests that are received in 30 minutes, where the non-beneficial requests are down-sampled so that the non-beneficial requests have the same frequency as the beneficial requests. *Testset1* (T1) and *Testset2* (T2) respectively contains *all* the ad requests received in 3 minutes, where the non-beneficial requests are nearly 200 times of the beneficial ones. We use the trainset to train the XGBoost classifier, and use the trained classifier to classify the instances in the testsets. In the evaluation, we compare the strawman solution with a baseline, Random, which randomly assigns the ad requests into beneficial and non-beneficial categories.

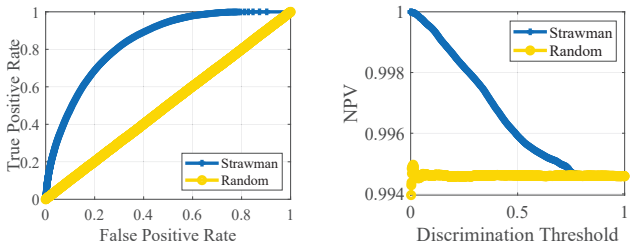
Predictive Performance. We first validate the correlation between the predicted clicking probability and the actual outcome of the ad requests (i.e., beneficial or non-beneficial). Figure 2(a) shows the Receiver Operating Characteristic (ROC) curves of the Strawman and Random. ROC, which is a graph of true positive rate against false positive rate for all possible discrimination thresholds, is commonly used to describe and compare the predictive performance of machine-learning models. The more convex the ROC curve, the better the predictive performance. Moreover, we compute the Area Under ROC Curve (AUC) of the two solutions. AUC ranges in value from 0 to 1, and a larger AUC generally means a better predictive performance [34], [44]. Particularly, an AUC of 0.5 means that the classifier has no predictive capacity. AUC of Strawman on T1 and T2 are 0.837 and 0.834, respectively, while that of Random on the two testsets are respectively 0.501 and 0.500.

The above results reveal that the outcome of the ad requests are predictable; however, the prediction of the XGBoost classifier is not 100% correct. In fact, it is easy to understand,

[4]Discrimination threshold is the probability at which the positive class is chosen over the negative class in classification. An ad request is classified as a beneficial request if its clicking probability is above the discrimination threshold, or it is classified as a non-beneficial request.

TABLE II: Details of datasets.

Dataset	Time range	Instances number
Trainset (down-sampled)	30 minutes	3 millions
Testset1 (T1)	3 minutes	9 millions
Testset2 (T2)	3 minutes	9 millions



(a) ROC (b) NPV

Fig. 2: Performance of Strawman and Random on T1. We omit the performance curves on T2 for clarify, since both solutions have a similar performance on the two testsets.

since we try to estimate the clicking probability based only on the ad request itself. However, the consumers' clicks depend not only on the information carried in the ad requests, but also on the consumer profile, consumer behavior, and advertising campaign [34], etc. Therefore, it will be inefficient to improve the predictive performance by improving the design of machine-learning models of the estimator.

Classification Performance. We next validate the classification performance, i.e., given the predictive performance of the classifier, can the default discrimination threshold (i.e., 0.5) classify the ad requests correctly? Since we focus on identifying and filtering non-beneficial requests, i.e., the negative instances, we compute the Negative Predictive Value (NPV) as the discrimination threshold shifts in Figure 2(b). NPV reflects how many true negatives are correctly classified. We can see that the NPV of Random is around always 0.995. This is because non-beneficial requests account for about 99.5% in the testsets. The NPV of Strawman is below 0.996 when the discrimination threshold is 0.5, which is only slightly better than Random.

This reveals that the typical fixed-value discrimination threshold is insufficient to accurately classify the non-beneficial requests. However, we note that the NPV of Strawman is relatively high when the discrimination threshold is small. For example, the NPV is above 0.9999 when the discrimination threshold is below 0.15. Thus, Strawman shows the great potential to identify the most likely negative instances by selecting a proper discrimination threshold.

Summary. Rather than attempting to classify all the ad requests correctly by improving the predictive performance of the classifier, we aim to recognize the most likely non-beneficial requests by computing the optimal discrimination threshold. Therefore, we will take two steps to identify and filter the non-beneficial requests in *Win2*. We first predict the

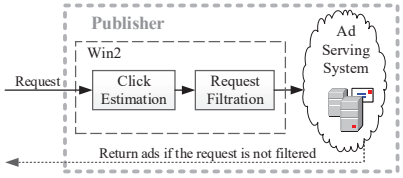


Fig. 3: The high-level architecture of *Win2*

clicking probability of the ad requests with a probabilistic estimator, and then computing the optimal discrimination threshold in classification. Moreover, we notice that the predictive performance of the estimator changes over time (although it is stable). It is understandable because the characteristics of the consumers and ads may change in different time slots. It motivates us dynamically update the discrimination threshold according to the predictive performance of the estimator to get the optimal result.

IV. Win2: Architecture

In this section, we describe the architecture of the publisher-side proactive ad request filtration solution *Win2*. As discussed in Section III, we filter non-beneficial requests in two steps. Thus, as shown in Figure 3, there are two main modules of *Win2*, namely Click Estimation and Request Filtration.

Click Estimation is a supervised probabilistic estimator that is trained based on the click log of the ad serving system. The entries in the click log are like $\{(r, click(r))\}$, where r is an ad request that consists of several key-value pairs, and $click(r) \in \{0, 1\}$ denotes the outcome of r with 0 representing non-click and 1 representing click. Features are extracted from the ad requests r to train the estimator. In operation, once receiving an ad request, the trained estimator takes the ad request as input, and computes the probability that the consumer will click if serving it. A higher probability indicates a higher likelihood that the consumer would click if an advertisement is matched and returned. Then, Click Estimation sends the ad request and its clicking probability to Request Filtration.

Request Filtration has two main functions, i.e., classification and filtering. Particularly, this module maintains the discrimination threshold. Upon receiving an ad request from Click Estimation, Request Filtration compares its clicking probability with the discrimination threshold. The request will be classified as a non-beneficial request if its clicking probability is less than the discrimination threshold, otherwise it will be classified as a beneficial request. After that, the predicted non-beneficial requests are dropped and thus no advertisement will be returned; the predicted beneficial requests are passed to the ad serving system, where proper ads are matched, returned and then displayed. In this way, the publisher is expected to serve less non-beneficial requests, while the consumers are expected to see less uninterested ads.

Obtaining a good discrimination threshold is the most critical and challenging issue in *Win2*. As shown in Section III, a small threshold filters less requests with a higher precision, while a large threshold filters more requests with a lower precision. Thus, we face a trade-off between the amount of ad requests filtered and the precision of classification, which makes the selection of discrimination threshold non-trivial. We will explorer the optimal discrimination threshold calculation in the next section.

V. Dynamic Threshold Computation

In this section, we formulate the threshold computation problem with utility-based optimization. Such a method has been shown to be a natural choice to explore the relationship between the publishers and consumers [45]. After that, we propose the dynamic computation mechanism to solve it. Table III summarizes the notations in this section.

A. Time-Slotted System

It is known that both publishers' and consumers' wellbeing are related to the number of ads displayed and clicked in a period of time. We, therefore, consider our model as a time-slotted system, which has been a common practice for utility optimization in digital marketing [46], [47].

Suppose a publisher runs a variety of products (i.e., apps and webpages) that contain a set of ad units $\mathcal{A}$. In every slot t, a set of consumers $\mathcal{M}^t$ use the publisher's products and the integrated ad units send ad requests to the publisher. The publisher applies *Win2* for ad request filtration, and an ad request will be served only when it is classified as a beneficial request. For simplicity, we assume the ad serving system returns one and only one piece of advertisement to serve an ad request. The consumers will click if they are interested in the ads displayed. Suppose the publisher adopts the CPC pricing model, i.e., it gains from the consumers' clicks.

Since *Win2* runs on the publisher side, to encourage the publishers to implement and deploy the system, we set the discrimination threshold to maximize publisher utility without compromising consumer utility in each time slot.

B. Consumer Utility

In this time-slotted system, each advertisement causes an annoyance to the consumer. However, clicking an interesting advertisement can satisfy a consumer's informational and emotional needs, which will benefit the consumer [48]. Inspired by the consumer utility modeling work in targeted advertising [45], we define the utility of a consumer as the wellbeing brought by the clicks minus the annoyance cost brought by the ads displayed.

In time slot t, the ad serving system serves $\sum_{a\in\mathcal{A}} r^t_{am}(\hat{p}^t)$ ad requests sent by consumer m, and $\sum_{a\in\mathcal{A}} r^t_{am}(\hat{p}^t)$ ads are displayed to consumer m accordingly. Among them, $\sum_{a\in\mathcal{A}} k^t_{am}(\hat{p}^t)$ ads are clicked. Hence, the utility of consumer m is expressed as $U^t_m(\hat{p}^t) = \sum_{a\in\mathcal{A}} (v_m \cdot k^t_{am}(\hat{p}^t) - c_m \cdot r^t_{am}(\hat{p}^t))$, where c_m denotes the average nuisance cost for consumer m to see an advertisement, and v_m denotes the average surplus in the utility of m brought by a click.

TABLE III: Summary of notations

Notations	Meaning
$\mathcal{M}^t$	Set of consumers in time slot t;
$\mathcal{A}$	Set of ad units;
$\hat{p}^t$	The value of the discrimination threshold in time slot t. Especially, $\hat{p}^t = 0$ means that all the ad requests are regarded as beneficial requests and served.
c_m	Average nuisance cost for consumer m to see an advertisement;
v_m	Average surplus in utility that a beneficial request (i.e., a click) brings to consumer m;
$r^t_{am}(\hat{p}^t)$	The number of ad requests of ad unit a sent by consumer m that will be served in time slot t if the discrimination threshold is equal to $\hat{p}^t$;
$k^t_{am}(\hat{p}^t)$	The number of beneficial requests of ad unit a sent by consumer m that will be served (i.e., the number of clicks) in time slot t if the discrimination threshold is equal to $\hat{p}^t$;
$U^t_C(\hat{p}^t)$	Consumer utility in time slot t if the discrimination threshold is equal to $\hat{p}^t$;
ϕ_a	Average revenue of a beneficial request (i.e., a click) of ad unit a;
θ_a	Average resource cost for the publisher to serve an ad request of ad unit a;
$V^t_{\mathcal{A}}(\hat{p}^t)$	Advertising revenue of the publisher in time slot t if the discrimination threshold is equal to $\hat{p}^t$;
$S^t_{\mathcal{A}}(\hat{p}^t)$	Resource cost of the publisher in time slot t if the discrimination threshold is equal to $\hat{p}^t$;
$U^t_P(\hat{p}^t)$	Publisher utility in time slot t if the discrimination threshold is equal to $\hat{p}^t$;

We define the consumer utility as the total utility of all the consumers in the time slot:

$$U^t_C(\hat{p}^t) = \sum\nolimits_{a\in\mathcal{A}, m\in\mathcal{M}^t} (v_m \cdot k^t_{am}(\hat{p}^t) - c_m \cdot r^t_{am}(\hat{p}^t)). \tag{1}$$

C. Publisher Utility

The publisher utilizes IDCs resources to offer ad services and gains from the consumers' clicks. Hence, we consider publisher utility as a trade-off between resource cost and advertising revenue.

In time slot t, $\sum_{m\in\mathcal{M}^t} r^t_{am}(\hat{p}^t)$ ad requests from ad unit a are served by the ad serving system, and thus $\sum_{m\in\mathcal{M}^t} r^t_{am}(\hat{p}^t)$ ads are displayed in the ad unit. Among them, $\sum_{m\in\mathcal{M}^t} k^t_{am}(\hat{p}^t)$ ads are clicked by the consumers.

The publisher's advertising revenue is equal to the product of the number of clicks and average revenue per click. Thus, the advertising revenue in time slot t is expressed as:

$$V^t_{\mathcal{A}}(\hat{p}^t) = \sum\nolimits_{a\in\mathcal{A}, m\in\mathcal{M}^t} (\phi_a \cdot k^t_{am}(\hat{p}^t)), \tag{2}$$

where ϕ_a denotes the average revenue of a click of ad unit a.

The resource cost for the publisher to provide ad service includes server cost, infrastructure cost and power consumption in the IDCs, .etc. We notice that these costs are generally in proportional to the IDCs resource needed to provide service [49], [50], and it is common sense that the resource demand in the production systems is approximately in proportional to the arrival rate of service requests [51]. Thus, we define the resource cost as a linear function of the number of ad requests served in the slot:

$$S^t_{\mathcal{A}}(\hat{p}^t) = \sum\nolimits_{a\in\mathcal{A}, m\in\mathcal{M}^t} (\theta_a \cdot r^t_{am}(\hat{p}^t)), \tag{3}$$

where θ_a represents the average resource cost to serve an ad request of ad unit a.

We present publisher utility as the difference between the advertising revenue and the resource cost:

$$U^t_P(\hat{p}^t) = V^t_{\mathcal{A}}(\hat{p}^t) - S^t_{\mathcal{A}}(\hat{p}^t). \tag{4}$$

D. Utility Optimization

In every time slot t, *Win2* determines a proper discrimination threshold $\hat{p}^t$ to maximize publisher utility while not harming consumer utility. We formulate the problem as:

$$\begin{aligned}
\underset{\hat{p}^t}{\text{maximize}} \quad & U^t_P(\hat{p}^t) = V^t_{\mathcal{A}}(\hat{p}^t) - S^t_{\mathcal{A}}(\hat{p}^t) \\
\text{subject to} \quad & U^t_C(\hat{p}^t) \geq U^t_C(0), \\
& V^t_{\mathcal{A}}(\hat{p}^t) = \sum\nolimits_{a\in\mathcal{A}, m\in\mathcal{M}^t} (\phi_a \cdot k^t_{am}(\hat{p}^t)), \\
& S^t_{\mathcal{A}}(\hat{p}^t) = \sum\nolimits_{a\in\mathcal{A}, m\in\mathcal{M}^t} (\theta_a \cdot r^t_{am}(\hat{p}^t)), \\
& U^t_C(\hat{p}^t) = \sum\nolimits_{a\in\mathcal{A}, m\in\mathcal{M}^t} (v_m \cdot k^t_{am}(\hat{p}^t) \\
& \qquad\qquad - c_m \cdot r^t_{am}(\hat{p}^t)), \\
& U^t_C(0) = \sum\nolimits_{a\in\mathcal{A}, m\in\mathcal{M}^t} (v_m \cdot k^t_{am}(0) \\
& \qquad\qquad - c_m \cdot r^t_{am}(0)).
\end{aligned}$$

In the formulation, $r^t_{am}(\hat{p}^t)$ and $k^t_{am}(\hat{p}^t)$ are related to the predictive performance of Click Estimation, and thus are difficult to be expressed as the formulae of $\hat{p}^t$. As a result, it is non-trivial to solve the problem via optimization techniques. To decide the optimal threshold in each time slot, we propose a dynamic threshold computation mechanism.

E. Design of Dynamic Threshold

The main challenge of threshold computation is to evaluate the precision of the classifier, i.e., estimate the value of $r^t_{am}(\hat{p}^t)$ and $k^t_{am}(\hat{p}^t)$ as $\hat{p}^t$ shifts. Notice that we can hardly tell whether a filtered ad request is a true negative in the online scenario, because no advertisement was displayed to the consumer once the request is dropped. To mitigate the challenge, we apply the following implementations to better estimate the predictive performance of Click Estimation and compute the optimal discrimination threshold accordingly. The detailed design of Request Filtration with dynamic threshold computation is given in Figure 4.

Traffic Splitting. As illustrated in Section IV, Click Estimation tags each ad request with its clicking probability and send the tagged requests to Request Filtration. In Request Filtration, we use Traffic Splitting to randomly assign the ad requests into two groups, namely treatment group and control group. The control group is passed to and served by the ad serving system. The treatment group is sent to Filter, which drops the ad requests that are classified as non-beneficial.

Threshold Computation. By analyzing the predicted clicking probability and the actual outcome of the ad requests in the control group, it is easy to infer the predictive performance of the estimator. In each time slot t, Threshold Computation computes a list of $(p, \sum_{m\in\mathcal{M}^{t-1}} \bar{r}^{t-1}_{am}(p), \sum_{m\in\mathcal{M}^{t-1}} \bar{k}^{t-1}_{am}(p) | p \in$

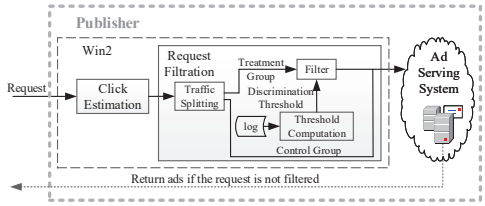


Fig. 4: Detailed design of Request Filtration

$(0, 1], a \in \mathcal{A}$) based on the click log of control group in the previous time slot, where $\bar{r}_{am}^{t-1}(p)$ and $\bar{k}_{am}^{t-1}(p)$ respectively denotes the number of ad requests and the number of beneficial requests of ad unit a from consumer m in the control group whose clicking probability are greater than p. After that, it decides the discrimination threshold according to Algorithm 1. In each iteration, we increase p by Δp, and compute both sides' expected utility if the discrimination threshold is set to p. Finally, we set $\hat{p}^t$ as the p that is expected to maximize publisher utility without lowering consumer utility and send $\hat{p}^t$ to Filter.

Filter. Filter stores the discrimination threshold and filters the ad requests according to the threshold. It updates the threshold when receiving $\hat{p}^t$ from Threshold Computation.

Algorithm 1: Dynamic Discrimination Threshold Computation in *Win2*

Input: $\{(p, \sum_{m \in \mathcal{M}^{t-1}} \bar{r}_{am}^{t-1}(p), \sum_{m \in \mathcal{M}^{t-1}} \bar{k}_{am}^{t-1}(p)) | p \in (0, 1], a \in \mathcal{A}\}$;

Output: Discrimination threshold $\hat{p}^t$ in time slot t;

$\bar{U}_{P\max}^{t-1} = \sum_{a \in \mathcal{A}, m \in \mathcal{M}^{t-1}} (\phi_a \cdot \bar{k}_{am}^{t-1}(0) - \theta_a \cdot \bar{r}_{am}^{t-1}(0))$;

$\bar{U}_{C\min}^{t-1} = \sum_{a \in \mathcal{A}, m \in \mathcal{M}^{t-1}} (v_m \cdot \bar{k}_{am}^{t-1}(0) - c_m \cdot \bar{r}_{am}^{t-1}(0))$;

$\hat{p}^t = 0$;

for $p = 0$ to 1 **do**

  $p = p + \Delta p$;

  $\bar{U}_P^{t-1} = \sum_{a \in \mathcal{A}, m \in \mathcal{M}^{t-1}} (\phi_a \cdot \bar{k}_{am}^{t-1}(p) - \theta_a \cdot \bar{r}_{am}^{t-1}(p))$;

  $\bar{U}_C^{t-1} = \sum_{a \in \mathcal{A}, m \in \mathcal{M}^{t-1}} (v_m \cdot \bar{k}_{am}^{t-1}(p) - c_m \cdot \bar{r}_{am}^{t-1}(p))$;

  if $\bar{U}_{P\max}^{t-1} < \bar{U}_P^{t-1}$ and $\bar{U}_{C\min}^{t-1} \leq \bar{U}_C^{t-1}$ **then**

    $\bar{U}_{P\max}^{t-1} = \bar{U}_P^{t-1}$;

    $\hat{p}^t = p$;

return $\hat{p}^t$;

VI. System Implementation

Win2 is implemented in 3000+ lines of C code. The components of *Win2* are instantiated as container groups [52] in the IDCs. The containers are managed by an orchestrator, which automatically scales the containers horizontally or vertically based on the arrival rate of ad requests. We implemented Remote Procedure Call (RPC) interfaces for the communication between the components of *Win2*.

Click Estimation. The main component of Click Estimator is the probabilistic estimator. It should be emphasized that any kind of probabilistic estimator can be applied in *Win2*. In fact, we face the trade-off between effectiveness and efficiency when designing the estimator. Notice that a large-scale ad serving system can receive tens of thousands ad requests in a second [5] and the feature vectors of ad requests could easily reach billions of dimensions after encoding [6]. Thus, we use XGBoost for efficiency and scalability consideration.

In operation, we train the estimator of Click Estimation incrementally with the help of a distributed machine learning platform of Baidu. More specifically, the classifier is trained every 24 hours, and we use the click log of the control group in the previous day as the trainset. To train the estimator, we apply the logistic regression as loss function and use the L2 regularization to penalize the complexity of the classifier.

The feature space can generally be divided into three categories, i.e., identification features, hardware features and software features. Identification features are used to identify an ad request, e.g., timestamp, ad request ID and ad unit ID, etc. Hardware features are related to the consumers' hardware, such as Collision resistant Unique IDentifier (CUID), International Mobile Equipment Identity (IMEI), IDentifier For Advertisers (IDFA) and network (e.g., cellular network or WiFi). Software features are related to the software platform, such as User Agent (UA), Operation System (OS), OS version, app name and app version. The feature space is one-hot encoded and the dimension reaches tens of millions.

Request Filtration. Request Filtration has three components, namely Traffic Splitting, Threshold Computation and Filter.

Traffic Splitting adopts hashing-based techniques [53] to split the ad requests. Particularly, Traffic Splitting applies a hash function with a subset of the fields in the ad requests, and then equally splits the requests into 100 bins according to the hashing value. The 100 bins are then mapped to two groups (i.e., treatment and control) according to the allocation table. The table specifies that which bin should be assigned to which group. The default split ratio is 90/10, where 90 bins of the ad requests are assigned to the treatment group, and 10 bins are assigned to the control group. Operators can tune the split ratio by adjusting the allocation table.

The click log of the ad serving system is stored in a distributed file system, and Threshold Computation accesses the log through C API. For efficiency consideration, Threshold Computation randomly samples the entries in the log to compute the optimal discrimination threshold. Filter simply compares the clicking probability field of the ad request with the current discrimination threshold, and sends the request to the ad serving system if it has a greater clicking probability.

VII. Performance Evaluation

We deployed and evaluated *Win2* in a large-scale in-feed[5] ad serving system of Baidu. The system runs in IDCs that consist

[5]In-feed ads are the ads placed inside a feed, i.e., a list of articles or news.

of tens of thousands of servers, and serves several billions ad requests from tens of ad units in a day.

In the evaluation, we seek to understand: 1) How much utility improvement can *Win2* bring to the publisher and consumers? 2) What are the performance of the main modules of *Win2*? 3) Given that we focus on publisher and consumer utility improvement, an interesting question is, what is the effect of *Win2* on advertisers' wellbeing?

Summary of Results:

- **Utility improvement.** Evaluation results show that *Win2* enhances publisher utility and consumer utility by up to 1.05% and 213.51%, respectively, which substantially outperforms alternative designs.
- **Performance of main modules.** The AUC of Click Estimation is always above 0.83, and the predictive performance is stable. Given the performance of Click Estimation, the optimality gap between our solution and the theoretical lower bound is below 2% and 6% for publisher utility and consumer utility, respectively. It reveals that Requests Filtration can always select the near optimal discrimination threshold.
- **Advertisers' wellbeing.** Evaluations show that *Win2* tends to increase the clicks of the advertisers' ads by showing the ads to more consumers who are interested in them, which will improve advertiser utility.

A. Experimental Setting

Baseline. As far as we know, there is no previous researches have focused on proactive ad request filtration on the publisher side. In the evaluation, we compare *Win2* with the rule-based ad request filtration solution that currently runs in this ad serving system, i.e., Ad unit Based Filtration (*ABF*). *ABF* reduces IDCs resource utilization by filtering the ad requests of the ad units with low beneficial request ratio.

The work flow of *ABF* is given in Figure 5. Upon receiving the ad requests from the consumers, Traffic Splitting randomly assigns the requests into treatment group and control group. The treatment group is sent to Filter and the control group is served by the ad serving system. In each time slot t, Filtration List Generation computes a pair list $\{(a, \overline{brr}_a^{t-1}) | a \in \mathcal{A}\}$ based on the click log, where $\overline{brr}_a^{t-1}$ denotes the beneficial request ratio of ad unit a in the control group in slot $t-1$. After that, it updates the filtration list according to Algorithm 2. In the algorithm, $\bar{r}_{am}^{t-1}(0)$ and $\bar{k}_{am}^{t-1}(0)$ respectively represents the number of ad requests and beneficial requests of ad unit a in the control group in slot $t-1$. Filtration List Generation adds the ad units into the filtration list $\mathcal{L}^t$ in order of lowest beneficial request ratio, until publisher utility is expected to be maximized without lowering consumer utility. Finally, it sends the filtration list $\mathcal{L}^t$ to Filter, which drops the ad requests of the ad units $a \in \mathcal{L}^t$ in the treatment group.

Parameters. We run both solutions for 30 minutes and each time slot is 3 minutes. The utility of consumer m is related to c_m and v_m in Equation (1), where c_m denotes the average nuisance cost for each ad and v_m represents the average increment in utility a click brings. Since there is no prior research focusing on the parameter settings of consumer utility in online advertising, we experimentally set $c_m = 1$, $v_m \in [1, 160]$ in the evaluation. We set v_m below 160 because consumer utility is positive when $v_m > 160$, which is inconsistent with the fact that most consumers have negative attitudes toward online advertising. The larger v_m, the greater the surplus in consumer utility brought by clicks. Publisher utility is related to the advertising revenue $V_{\mathcal{A}}^t(\hat{p}^t)$ and the resource cost $S_{\mathcal{A}}^t(\hat{p}^t)$. In the evaluation, we empirically set $V_{\mathcal{A}}^t(\hat{p}^t)$ as the advertising revenue in slot $t-1$, and $S_{\mathcal{A}}^t(\hat{p}^t)$ as the average power consumption and depreciation cost of the IDCs in a time slot.

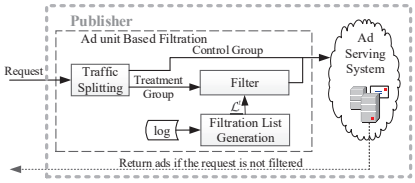


Fig. 5: The baseline solution *ABF*.

Algorithm 2: Filtration List Generation in *ABF*

Input: Pair list $\{(a, \overline{brr}_a^{t-1}) | a \in \mathcal{A}\}$;
Output: Filtration list $\mathcal{L}^t$ in time slot t;

$\mathcal{B} = \mathcal{A}$;
Sort the ad units in $\mathcal{B}$ by $\overline{brr}_a^{t-1}$ from low to high;
$\bar{U}_{P\max}^{t-1} = \sum_{a\in\mathcal{A}, m\in\mathcal{M}^{t-1}} (\phi_a \cdot \bar{k}_{am}^{t-1}(0) - \theta_a \cdot \bar{r}_{am}^{t-1}(0))$;
$\bar{U}_{C\min}^{t-1} = \sum_{a\in\mathcal{A}, m\in\mathcal{M}^{t-1}} (v_m \cdot \bar{k}_{am}^{t-1}(0) - c_m \cdot \bar{r}_{am}^{t-1}(0))$;
$\mathcal{L}^t = \{\}$;
for each ad unit $a \in \mathcal{B}$ **do**
  $\mathcal{B} = \mathcal{B} \backslash \{a\}$;
  $\bar{U}_P^{t-1} = \sum_{a\in\mathcal{B}, m\in\mathcal{M}^{t-1}} (\phi_a \cdot \bar{k}_{am}^{t-1}(0) - \theta_a \cdot \bar{r}_{am}^{t-1}(0))$;
  $\bar{U}_C^{t-1} = \sum_{a\in\mathcal{B}, m\in\mathcal{M}^{t-1}} (v_m \cdot \bar{k}_{am}^{t-1}(0) - c_m \cdot \bar{r}_{am}^{t-1}(0))$;
  if $\bar{U}_{P\max}^{t-1} < \bar{U}_P^{t-1}$ and $\bar{U}_{C\min}^{t-1} \leq \bar{U}_C^{t-1}$ **then**
    $\bar{U}_{P\max}^{t-1} = \bar{U}_P^{t-1}$;
    $\mathcal{L}^t = \mathcal{L}^t \cup \{a\}$;
  else
    break;
return $\mathcal{L}^t$;

B. Utility Improvement

Figures 6 and 7 respectively compares the utility improvement of publisher and consumers in *Win2* and *ABF*. For both publisher and consumers, the utility improvement is defined as the percentages by which their utility is enhanced by ad request filtration. When v_m is 160, *Win2* enhances publisher utility and consumer utility by up to 1.05% and 213.51%,

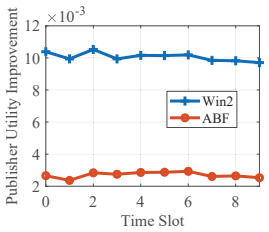


Fig. 6: Publisher utility improvement.

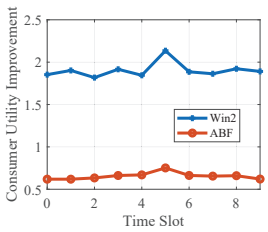


Fig. 7: Consumer utility improvement ($v_m = 160$).

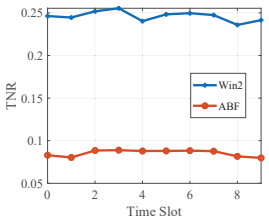


Fig. 8: True Negative Rate.

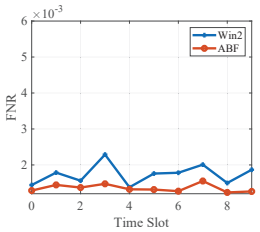


Fig. 9: False Negative Rate.

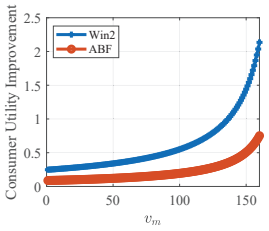


Fig. 10: Consumer utility improvement ($v_m \in [1, 160]$).

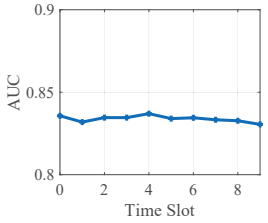


Fig. 11: AUC of Click Estimation.

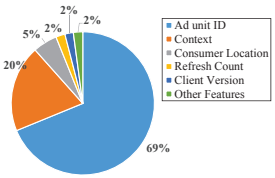


Fig. 12: Feature importance in Click Estimation.

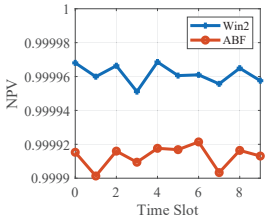


Fig. 13: Negative Predictive Value.

respectively, while *ABF* improves them respectively by up to 0.29% and 110.00%.

Figures 8 and 9 further analyse the True Negative Rate (TNR) and False Negative Rate (FNR) of the two solutions. TNR tells the proportion of non-beneficial requests that are correctly identified and filtered. The higher TNR, the more non-beneficial requests will be identified and filtered. FNR tells the percentage of beneficial requests that are predicted as non-beneficial and filtered. The lower FNR, the less beneficial requests will be filtered. *Win2* filters 24.60% of the non-beneficial requests and 0.17% of the beneficial requests in average, which significantly outperforms *ABF*. In other words, from publisher perspective, *Win2* saves up to 24.6% of the resource cost at the expense of 0.17% of advertising revenue loss, which finally results in the 1.05% improvement of publisher utility; from consumer perspective, *Win2* reduces up to 24.6% of the annoying ads at the expense of blocking 0.17% of the ads that the consumers are interested in, which leads to the 213.51% consumer utility improvement.

Moreover, to observe the impact of parameter settings on consumer utility, Figure 10 shows the consumer utility improvement in time slot 5 as v_m varies in [1,160]. In both *Win2* and *ABF*, consumer utility improvement increases with v_m, since a larger v_m means that consumers are more satisfied by viewing and clicking interested ads. *Win2* increases consumer utility by 213.51% and by 24.83% in the best case and worst case, respectively. It shows that *Win2* can effectively improve consumer utility independent of the value of v_m.

C. Performance of Win2 Modules

Click Estimation. To validate the predictive performance of the estimator, we compute the AUC in the ten time slots. As illustrated in Figure 11, the AUC is always above 0.83, which confirms that the outcome of the ad requests are predictable and the the estimator performs stably in operation.

Besides, a benefit of using gradient boosting algorithm is that it can tell the feature importance[6] in prediction, which helps to understand the opportunity to improve the ad serving system. Figure 12 shows the 5 features that are most correlated to the outcome of the ad requests, namely ad unit ID (which decides the position and size of the ads displayed), context (i.e., the topic that the consumer is browsing, e.g., sports, entertainment, technology, etc.), consumer's location information, refresh count (i.e., the number that the consumer refreshes the feed in the session, which indicates the dwell time of the consumer) and client version (i.e., version of the apps or web browsers). It reveals that, to increase clicks for the publishers, more emphasis should be given on the above factors in ads recommendation.

Request Filtration. To understand that whether Request Filtration can select proper discrimination threshold, Figure 13 shows the NPV after classification in the 10 time slots. The NPV of *Win2* is always above 0.99996, while that of *ABF* is mostly below 0.99992. It indicates that Request Filtration can effectively identify the most likely non-beneficial requests by selecting proper discrimination thresholds.

Moreover, given the predictive performance of Click Estimation, we compare *Win2* with the offline global optimal solution where the discrimination threshold is exactly set to the value that maximize publisher utility in that time slot. Figure 14 illustrates the optimality gap between *Win2* and the optimal solution. The gap of publisher utility is below 2%, while that of consumer utility is less than 6%. It also shows that Request Filtration can obtain the near optimal discrimination threshold.

[6]Feature importance quantifies the contribution of a feature on the predictive performance of the estimator. A higher feature importance score means a larger increment in predictive error if dropping this feature.

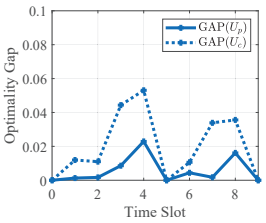


Fig. 14: Optimality gap of both sides' utility in *Win2*.

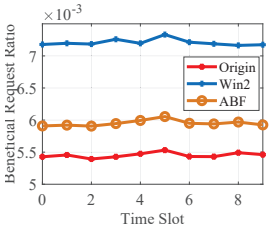


Fig. 15: Beneficial request ratio before and after filtration.

D. Advertisers' Wellbeing

In addition to publishers and consumers, the advertisers are also important participants in online advertising. An advertiser's wellbeing is in proportional to the number of its ads that clicked by consumers, because more clicks usually mean more interested consumers know about the advertiser's products.

Prior researches show that cumulative exposure to online ads can prompt consumers to filter out the excess ads [54], and reducing exposure to the ads increases consumer engagement with the web by 28% in average [23]. We notice that *Win2* reduces the consumers' exposure to uninterested ads. Hence, *Win2* tends to encourage consumers more engaged. Figure 15 further compares the ratio of beneficial requests before and after filtration. It can be seen that, *Win2* enhances this ratio by up to 30% by filtering out the non-beneficial requests.

Therefore, *Win2* shows the potential to increase ad clicks for the advertisers by encouraging consumers more engaged (thus increase the number of ad requests) as well as enhancing the ratio of beneficial requests. Quantifying the effect of *Win2* on advertiser utility is an important part of our future work.

VIII. Discussion

The pricing models in online advertising can be divided into two categories, namely performance-based pricing model and exposure-based pricing model [55].

Performance-based advertising, where the advertiser pays when an advertisement results in a consumer's action (e.g., a click), accounts for more than 60% of the global online advertising revenue [56]. In this paper, we assume the publisher adopts the CPC pricing model, which is the most popular performance-based pricing model. Besides CPC, an emerging performance-based pricing model is Cost Per Action (CPA) [57], where the publishers gain from clicks that subsequently see consumers complete specific actions, such as download, registration, or sign-up. We note that *Win2* is applicable to the CPA pricing model, since the ad requests filtered by *Win2* are unlikely result in clicks, not to mention further actions.

Exposure-based model is the earliest pricing model in online advertising, where the advertiser pays for each time an advertisement is exposed to the consumers. Publishers gain more by displaying more ads in such pricing models, which is inconsistent with our assumption in this paper. Thus, *Win2* is not fit for exposure-based pricing model, since filtering any ad requests can result in revenue loss to the publishers. Improving publisher utility without harming consumer utility in exposure-based pricing model is a part of our future work.

IX. Conclusion

Online advertising plays an important role in the Internet ecosystem. Solutions have been proposed to improve the publisher's advertising revenue, but the situation is not satisfactory.

In this paper, aiming to improve the publisher's advertising income as well as consumer experience, we propose a publisher-side proactive ad request filtration solution *Win2*. *Win2* estimates the clicking probability of ad requests with a probabilistic estimator, and then filter the ad requests that unlikely result in clicks. With the help of *Win2*, the publisher's income is increased by lowering resource cost and consumer experience is improved by viewing fewer uninterested ads. We implement and deploy *Win2* in a large-scale ad serving system. Evaluations show that *Win2* enhances publisher utility by up to 1.05% and consumer utility by up to 213.51%.

Acknowledgment

Ke Xu is the corresponding author. Ke Xu's work was in part supported by China National Funds for Distinguished Young Scientists with No. 61825204, NSFC Project with No. 61932016, Beijing Outstanding Young Scientist Program with No. BJJWZYJH01201910003011 and Beijing National Research Center for Information Science and Technology (BNRist) with No. BNR2019RC01011. Meng Shen's work was in part supported by NSFC Projects with No. 61602039 and No. 61972039, and Beijing Natural Science Foundation with No. 4192050.

References

[1] IAB, "Iab internet advertising revenue report," Interactive Advertising Bureau, Tech. Rep., 2018.

[2] G. A. Help, "Clickthrough rate (ctr)," https://support.google.com/adsense/answer/6157482?hl=en, 2019, [Online; accessed June-2019].

[3] ——, "Impression," https://support.google.com/google-ads/answer/6320?hl=en, 2019, [Online; accessed June-2019].

[4] ——, "Conversion rate," https://support.google.com/google-ads/answer/2684489?hl=en, 2019, [Online; accessed May-2019].

[5] A. Gupta and J. Shute, "High-availability at massive scale: Building google's data infrastructure for ads," *Proc. of BIRTE*, 2015.

[6] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, M. Wattenberg, A. M. Hrafnkelsson, T. Boulos, and J. Kubica, "Ad click prediction: A view from the trenches," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13, 2013, pp. 1222–1230.

[7] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers *et al.*, "Practical lessons from predicting clicks on ads at facebook," in *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*. ACM, 2014, pp. 1–9.

[8] T. Graepel, T. Borchert, and R. Herbrich, "Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine," in *International Conference on International Conference on Machine Learning*, 2010, pp. 13–20.

[9] Y. Tagami, S. Ono, K. Yamamoto, K. Tsukamoto, and A. Tajima, "Ctr prediction for contextual advertising: Learning-to-rank approach," in *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising*, ser. ADKDD '13, 2013, pp. 4:1–4:8.

[10] C. Li, Y. Lu, Q. Mei, D. Wang, and S. Pandey, "Click-through prediction for advertising in twitter timeline," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1959–1968.

[11] X. Wang, W. Li, Y. Cui, R. Zhang, and J. Mao, "Click-through rate estimation for rare events in online advertising," *Online Multimedia Advertising Techniques & Technologies*, 2011.

[12] G. A. Help, "Cost-per-click," https://support.google.com/google-ads/answer/116495?hl=en, 2019, [Online; accessed Feb-2019].

[13] I. Faizullabhoy and A. Korolova, "Facebook's advertising platform: New attack vectors and the need for interventions," *arXiv preprint arXiv:1803.10099*, 2018.

[14] J. Wielki and J. Grabara, "The impact of ad-blocking on the sustainable development of the digital advertising ecosystem," *Sustainability*, vol. 10, no. 11, p. 4039, 2018.

[15] S. Blanchfield, "The state of the blocked web," PageFair, Tech. Rep., 2017.

[16] Wikipedia, "Adblock plus," https://en.wikipedia.org/wiki/Adblock_Plus, 2019, [Online; accessed July-2019].

[17] A. Plus, "Surf the web with no annoying ads," https://adblockplus.org/, 2019, [Online; accessed April-2019].

[18] EasyList, "Easylist overview," https://easylist.to/, 2019, [Online; accessed April-2019].

[19] N. Kushmerick, "Learning to remove internet advertisements," in *Agents*. Citeseer, 1999, pp. 175–181.

[20] P. L. Szczepanski, A. Wi?niewski, and T. Gerszberg, "An automated framework with application to study url based online advertisements detection," *Journal of Applied Mathematics, Statistics and Informatics*, vol. 9, no. 1, pp. 47–60, 2013.

[21] V. Krammer, "An effective defense against intrusive web advertising," in *2008 Sixth Annual Conference on Privacy, Security and Trust*. IEEE, 2008, pp. 3–14.

[22] U. Iqbal, Z. Shafiq, P. Snyder, S. Zhu, Z. Qian, and B. Livshits, "Adgraph: A machine learning approach to automatic and effective adblocking," *arXiv preprint arXiv:1805.09155*, 2018.

[23] B. Miroglio, D. Zeber, J. Kaye, and R. Weiss, "The effect of ad blocking on user engagement with the web," *international world wide web conferences*, pp. 813–821, 2018.

[24] B. Shiller, J. Waldfogel, and J. Ryan, "The effect of ad blocking on website traffic and quality," *The RAND Journal of Economics*, vol. 49, no. 1, pp. 43–63, 2018.

[25] Adback, "Worldwide ranking of websites losing revenue due to adblockers," https://www.adback.co/revenue-loss-adblock-websites-ranking, 2019, [Online; accessed April-2019].

[26] IAB, "Ad blocking: Who blocks ads, why and how to win them back," Interactive Advertising Bureau, Tech. Rep., 2018.

[27] S. Zhu, X. Hu, Z. Qian, Z. Shafiq, and H. Yin, "Measuring and disrupting anti-adblockers using differential execution analysis," in *The Network and Distributed System Security Symposium (NDSS)*, 2018.

[28] S. Zhu, U. Iqbal, Z. Wang, Z. Qian, Z. Shafiq, and W. Chen, "Shadowblock: A lightweight and stealthy adblocking browser," in *The Web Conference*. Web4Good, 2019.

[29] S. Nichols, "Adblock plus blocks facebook block of adblock plus block of facebook block of adblock plus block of facebook ads," https://www.theregister.co.uk/2016/08/12/facebook_block_shock/, 2016, [Online; accessed April-2019].

[30] R. Nithyanand, S. Khattak, M. Javed, N. Vallinarodriguez, M. Falahrastegar, J. Powles, E. De Cristofaro, H. Haddadi, and S. J. Murdoch, "Adblocking and counter blocking: A slice of the arms race," *foundations of computational intelligence*, 2016.

[31] M. Richardson, E. Dominowska, and R. Ragno, "Predicting clicks:estimating the click-through rate for new ads," in *International Conference on World Wide Web*, 2007, pp. 521–530.

[32] A. K. Menon, K. P. Chitrapura, S. Garg, D. Agarwal, and N. Kota, "Response prediction using collaborative filtering with hierarchies and side-information," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011, pp. 141–149.

[33] L. Shan, L. Lei, C. Sun, and X. Wang, "Predicting ad click-through rates via feature-based fully coupled interaction tensor factorization," *Electronic Commerce Research & Applications*, vol. 16, no. C, pp. 30–42, 2016.

[34] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1059–1068.

[35] B. Edizel, A. Mantrach, and X. Bai, "Deep character-level click-through rate prediction for sponsored search," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '17, 2017, pp. 305–314.

[36] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, and T.-Y. Liu, "Sequential click prediction for sponsored search with recurrent neural networks." in *AAAI*, vol. 14, 2014, pp. 1369–1375.

[37] J. Pan, J. Xu, A. Lobos, W. Zhao, S. Pan, Y. Sun, and Q. Lu, "Field-weighted factorization machines for click-through rate prediction in display advertising," *the web conference*, pp. 1349–1357, 2018.

[38] Y. Deng, Y. Shen, and H. Jin, "Disguise adversarial networks for click-through rate prediction," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 2017, pp. 1589–1595.

[39] B. Dalessandro, D. Chen, T. Raeder, C. Perlich, M. H. Williams, and F. Provost, "Scalable hands-free transfer learning for online advertising," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 1573–1582.

[40] H. Cheng, R. v. Zwol, J. Azimi, E. Manavoglu, R. Zhang, Y. Zhou, and V. Navalpakkam, "Multimedia features for click prediction of new ads in display advertising," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 777–785.

[41] K. S. Dave and V. Varma, "Learning the click-through rate for rare/new ads from similar ads," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2010, pp. 897–898.

[42] O. Chapelle, "Modeling delayed feedback in display advertising," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 1097–1105.

[43] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016, pp. 785–794.

[44] Y. Zhao, M. Qiao, H. Wang, R. Zhang, D. Wang, K. Xu, and Q. Tan, "TDFI: Two-stage Deep Learning Framework for Friendship Inference via Multi-source Information," in *Proceedings of IEEE INFOCOM*, 2019, pp. 1981–1989.

[45] J. P. Johnson, "Targeted advertising and advertising avoidance," *The RAND Journal of Economics*, vol. 44, no. 1, pp. 128–144, 2013.

[46] Y. Zhao, H. Su, L. Zhang, D. Wang, and K. Xu, "Variety Matters: A New Model for the Wireless Data Market under Sponsored Data Plans," in *Proceedings of IEEE/ACM IWQoS*, 2019, pp. 23:1–23:10.

[47] Y. Zhao, H. Wang, H. Su, L. Zhang, R. Zhang, D. Wang, and K. Xu, "Understand Love of Variety in Wireless Data Market under Sponsored Data Plans," *IEEE Journal on Selected Areas in Communications (J-SAC)*, 2020.

[48] G. Brajnik and S. Gabrielli, "A review of online advertising effects on the user experience," *International Journal of Human-Computer Interaction*, vol. 26, no. 10, pp. 971–997, 2010.

[49] Z. Zhou, F. Liu, H. Jin, B. Li, B. Li, and H. Jiang, "On arbitrating the power-performance tradeoff in saas clouds," in *INFOCOM, 2013 Proceedings IEEE*, 2013, pp. 872–880.

[50] S. Chen, Y. Wang, and M. Pedram, "Resource allocation optimization in a data center with energy storage devices," in *Industrial Electronics Society, IECON 2014 - Conference of the IEEE*, 2015, pp. 393–399.

[51] M. Lin, A. Wierman, L. L. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *IEEE/ACM Transactions on Networking (TON)*, vol. 21, no. 5, pp. 1378–1391, 2013.

[52] L. Lv, Y. Zhang, Y. Li, K. Xu, D. Wang, W. Wang, M. Li, X. Cao, and Q. Liang, "Communication-aware container placement and reassignment in large-scale internet data centers," *IEEE Journal on Selected Areas in Communications (J-SAC)*, vol. 37, no. 3, pp. 540–555, 2019.

[53] Z. Wang, *Internet QoS: architectures and mechanisms for quality of service*. Morgan Kaufmann, 2001.

[54] J. D. Rumbo, "Consumer resistance in a world of advertising clutter: The case of adbusters," *Psychology & Marketing*, vol. 19, no. 2, pp. 127–148, 2002.

[55] S. Kumar, *Optimization Issues in Web and Mobile Advertising: Past and Future Trends*. Springer, 2015.

[56] IAB, "Iab internet advertising revenue report," Interactive Advertising Bureau, Tech. Rep., 2017.

[57] Wikipedia, "Cost per action," https://en.wikipedia.org/wiki/Cost_per_action, 2019, [Online; accessed Apr-2019].