Taking the Counterfactual Online: Efficient and Unbiased Online Evaluation for Ranking

Harrie Oosterhuis
University of Amsterdam
Amsterdam, The Netherlands
oosterhuis@uva.nl

Maarten de Rijke
University of Amsterdam & Ahold Delhaize
Amsterdam, The Netherlands
derijke@uva.nl

ABSTRACT

Counterfactual evaluation can estimate Click-Through-Rate (CTR) differences between ranking systems based on historical interaction data, while mitigating the effect of position bias and item-selection bias. We introduce the novel Logging-Policy Optimization Algorithm (LogOpt), which optimizes the policy for logging data so that the counterfactual estimate has minimal variance. As minimizing variance leads to faster convergence, LogOpt increases the data-efficiency of counterfactual estimation. LogOpt turns the counterfactual approach – which is indifferent to the logging policy – into an online approach, where the algorithm decides what rankings to display. We prove that, as an online evaluation method, LogOpt is unbiased w.r.t. position and item-selection bias, unlike existing interleaving methods. Furthermore, we perform large-scale experiments by simulating comparisons between *thousands* of rankers. Our results show that while interleaving methods make systematic errors, LogOpt is as efficient as interleaving without being biased.

ACM Reference Format:
Harrie Oosterhuis and Maarten de Rijke. 2020. Taking the Counterfactual Online: Efficient and Unbiased Online Evaluation for Ranking. In *Proceedings of the 2020 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '20), September 14–17, 2020, Virtual Event, Norway*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3409256.3409820

1 INTRODUCTION

Evaluation is essential for the development of search and recommendation systems [8, 14]. Before any ranking model is widely deployed it is important to first verify whether it is a true improvement over the currently-deployed model. A traditional way of evaluating relative differences between systems is through A/B testing, where part of the user population is exposed to the current system ("control") and the rest to the altered system ("treatment") during the same time period. Differences in behavior between these groups can then indicate if the alterations brought improvements, e.g. if the treatment group showed a higher Click-Through-Rate (CTR) or more revenue was made with this system [4].

ICTIR '20, September 14–17, 2020, Virtual Event, Norway

ACM ISBN 978-1-4503-8067-6/20/09…$15.00
https://doi.org/10.1145/3409256.3409820

Interleaving has been introduced in Information Retrieval (IR) as a more efficient alternative to A/B testing [11]. Interleaving algorithms take the rankings produced by two ranking systems, and for each query create an interleaved ranking by combining the rankings from both systems. Clicks on the interleaved rankings directly indicate relative differences. Repeating this process over a large number of queries and averaging the results, leads to an estimate of which ranker would receive the highest CTR [10]. Previous studies have found that interleaving requires fewer interactions than A/B testing, which enables them to make consistent comparisons in a much shorter timespan [4, 21].

More recently, counterfactual evaluation for rankings has been proposed by Joachims et al. [13] to evaluate a ranking model based on clicks gathered using a different model. By correcting for the position bias introduced during logging, the counterfactual approach can unbiasedly estimate the CTR of a new model on historical data. To achieve this, counterfactual evaluation makes use of Inverse-Propensity-Scoring (IPS), where clicks are weighted inversely to the probability that a user examined them during logging [22]. A big advantage compared to interleaving and A/B testing, is that counterfactual evaluation does not require online interventions.

In this paper, we show that no existing interleaving method is truly unbiased: they are not guaranteed to correctly predict which ranker has the highest CTR. On two different industry datasets, we simulate a total of 1,000 comparisons between 2,000 different rankers. In our setup, interleaving methods converge on the wrong answer for at least 2.2% of the comparisons on both datasets. A further analysis shows that existing interleaving methods are unable to reliably estimate CTR differences around 1% or lower. Therefore, in practice these systematic errors are expected to impact situations where rankers with a very similar CTR are compared.

We propose a novel online evaluation algorithm: Logging-Policy Optimization Algorithm (LogOpt). LogOpt extends the existing unbiased counterfactual approach, and turns it into an online approach. LogOpt estimates which rankings should be shown to the user, so that the variance of its CTR estimate is minimized. In other words, it attempts to learn the logging-policy that leads to the fastest possible convergence of the counterfactual estimation. Our experimental results indicate that our novel approach is as efficient as any interleaving method or A/B testing, without having a systematic error. As predicted by the theory, we see that the estimates of our approach converge on the true CTR difference between rankers. Therefore, we have introduced the first online evaluation method that combines high efficiency with unbiased estimation.

The main contributions of this work are:

(1) The first logging-policy optimization method for minimizing the variance in counterfactual CTR estimation.

(2) The first unbiased online evaluation method that is as efficient as state-of-the-art interleaving methods.
(3) A large-scale analysis of existing online evaluation methods that reveals a previously unreported bias in interleaving methods.

2 PRELIMINARIES: RANKER COMPARISONS

The overarching goal of ranker evaluation is to find the ranking model that provides the best rankings. For the purposes of this paper, we will define the quality of a ranker in terms of the number of clicks it is expected to receive. Let R indicate a ranking and let $\mathbb{E}[\text{CTR}(R)] \in \mathbb{R}_{\geq 0}$ be the expected number of clicks a ranking receives after being displayed to a user. We consider ranking R_1 to be better than R_2 if in expectation it receives more clicks: $\mathbb{E}[\text{CTR}(R_1)] > \mathbb{E}[\text{CTR}(R_2)]$. We will represent a ranking model by a policy π, with $\pi(R \mid q)$ as the probability that π displays R for a query q. With $P(q)$ as the probability of a query q being issued, the expected number of clicks received under a ranking model π is:

$$\mathbb{E}[\text{CTR}(\pi)] = \sum_q P(q) \sum_R \mathbb{E}[\text{CTR}(R)]\pi(R \mid q). \tag{1}$$

Our goal is to discover the $\mathbb{E}[\text{CTR}]$ difference between two policies:

$$\Delta(\pi_1,\pi_2) = \mathbb{E}[\text{CTR}(\pi_1)] - \mathbb{E}[\text{CTR}(\pi_2)]. \tag{2}$$

We recognize that to correctly identify if one policy is better than another, we merely need a corresponding binary indicator:

$$\Delta_{bin}(\pi_1,\pi_2) = \text{sign}\big(\Delta(\pi_1,\pi_2)\big). \tag{3}$$

However, in practice the magnitude of the differences can be very important, for instance, if one policy is much more computationally expensive while only having a slightly higher $\mathbb{E}[\text{CTR}]$, it may be preferable to use the other in production. Therefore, estimating the absolute $\mathbb{E}[\text{CTR}]$ difference is more desirable in practice.

2.1 User Behavior Assumptions

Any proof regarding estimators using user interactions must rely on assumptions about user behavior. In this paper, we assume that only two forms of interaction bias are at play: position bias and item-selection bias.

Users generally do not examine all items that are displayed in a ranking but only click on examined items [5]. As a result, a lower probability of examination for an item also makes it less likely to be clicked. Position bias assumes that only the rank determines the probability of examination [6]. Furthermore, we will assume that given an examination only the relevance of an item determines the click probability. Let $c(d) \in \{0,1\}$ indicate a click on item d and $o(d) \in \{0,1\}$ examination by the user. Then these assumptions result in the following assumed click probability:

$$\begin{aligned} P(c(d)=1 \mid R,q) &= P(o(d)=1 \mid R)P(c(d)=1 \mid o(d)=1,q) \\ &= \theta_{\text{rank}(d|R)}\zeta_{d,q}. \end{aligned} \tag{4}$$

Here $\text{rank}(d \mid R)$ indicates the rank of d in R; for brevity we use $\theta_{\text{rank}(d|R)}$ to denote the examination probability – $\theta_{\text{rank}(d|R)} = P(o(d) = 1 \mid R)$ – and $\zeta_{d,q}$ for the conditional click probability – $\zeta_{d,q} = P(c(d)=1 \mid o(d)=1,q)$.

We also assume that item-selection bias is present; this type of bias is an extreme form of position bias that results in zero examination probabilities for some items [16, 17]. This bias is unavoidable in top-k ranking settings, where only the $k \in \mathbb{N}_{>0}$ highest ranked items are displayed. Consequently, any item beyond rank k cannot be observed or examined by the user: $\forall r \in \mathbb{N}_{>0}(r > k \rightarrow \theta_r = 0)$. The distinction between item-selection bias and position bias is important because the original counterfactual evaluation method [13] is only able to correct for position bias when no item-selection bias is present [16, 17].

Based on these assumptions, we can now formulate the expected CTR of a ranking:

$$\mathbb{E}[\text{CTR}(R)] = \sum_{d \in R} P(c(d)=1 \mid R,q) = \sum_{d \in R} \theta_{\text{rank}(d|R)}\zeta_{d,q}. \tag{5}$$

While we assume this model of user behavior, its parameters are still assumed unknown. Therefore, the methods in this paper will have to estimate $\mathbb{E}[\text{CTR}]$ without prior knowledge of θ or ζ.

2.2 Goal: CTR-Estimator Properties

Recall that our goal is to estimate the CTR difference between rankers (Eq. 2); online evaluation methods do this based on user interactions. Let $\mathcal{I}$ be the set of available user interactions, it contains N tuples of a single (issued) query q_i, the corresponding displayed ranking R_i, and the observed user clicks c_i: $\mathcal{I} = \{(q_i, R_i, c_i)\}_{i=1}^{N}$. Each evaluation method has a different effect on what rankings will be displayed to users. Furthermore, each evaluation method converts each interaction into a single estimate using some function f: $x_i = f(q_i,R_i,c_i)$, the final estimate is simply the mean over these estimates: $\hat{\Delta}(\mathcal{I}) = \frac{1}{N}\sum_{i=1}^{N} x_i = \frac{1}{N}\sum_{i=1}^{N} f(q_i,R_i,c_i)$. This description fits all existing online and counterfactual evaluation methods for rankings. Every evaluation method uses a different function f to convert interactions into estimates; moreover, online evaluation methods also decide which rankings R to display when collecting $\mathcal{I}$. These two choices result in different estimators. Before we discuss the individual methods, we briefly introduce the three properties we desire of each estimator: consistency, unbiasedness and variance.

Consistency – an estimator is *consistent* if it converges as N increases. All existing evaluation methods are consistent as their final estimates are means of bounded values.

Unbiasedness – an estimator is *unbiased* if its estimate is equal to the true CTR difference in expectation:

$$\text{Unbiased}(\hat{\Delta}) \Leftrightarrow \mathbb{E}\big[\hat{\Delta}(\mathcal{I})\big] = \Delta(\pi_1,\pi_2). \tag{6}$$

If an estimator is both consistent and unbiased it is guaranteed to converge on the true $\mathbb{E}[\text{CTR}]$ difference.

Variance – the *variance* of an estimator is the expected squared deviation between a single estimate x and the mean $\hat{\Delta}(X)$:

$$\text{Var}(\hat{\Delta}) = \mathbb{E}\Big[\big(x - \mathbb{E}[\hat{\Delta}(\mathcal{I})]\big)^2\Big]. \tag{7}$$

Variance affects the rate of convergence of an estimator; for fast convergence it should be as low as possible.

In summary, our goal is to find an estimator, for the CTR difference between two ranking models, that is consistent, unbiased and has minimal variance.

3 EXISTING ONLINE AND COUNTERFACTUAL EVALUATION METHODS

We describe three families of online and counterfactual evaluation methods for ranking.

3.1 A/B Testing

A/B testing is a well established form of online evaluation to compare a system A with a system B [14]. Users are randomly split into two groups and during the same time period each group is exposed to only one of the systems. In expectation, the only factor that differs between the groups is the exposure to the different systems. Therefore, by comparing the behavior of each user group, the relative effect each system has can be evaluated.

We will briefly show that A/B testing is unbiased for $\mathbb{E}[\text{CTR}]$ difference estimation. For each interaction either π_1 or π_2 determines the ranking, let $A_i \in \{1,2\}$ indicate the assignment and $A_i \sim P(A)$. Thus, if $A_i = 1$, then $R_i \sim \pi_1(R \mid q)$ and if $A_i = 2$, then $R_i \sim \pi_2(R \mid q)$. Each interaction i is converted into a single estimate x_i by $f_{\text{A/B}}$:

$$x_i = f_{\text{A/B}}(q_i, R_i, c_i) = \left(\frac{\mathbb{1}[A_i = 1]}{P(A=1)} - \frac{\mathbb{1}[A_i = 2]}{P(A=2)} \right) \sum_{d \in R_i} c_i(d). \tag{8}$$

Abbreviating $f_{\text{A/B}}(q_i, R_i, c_i)$ as $f_{\text{A/B}}(\ldots)$, we can prove that A/B testing is unbiased, since in expectation each individual estimate is equal to the CTR difference:

$$\begin{aligned}
\mathbb{E}[f_{\text{A/B}}(\ldots)] &= \sum_q P(q) \left(\frac{P(A=1) \sum_R \pi_1(R \mid q) \mathbb{E}[\text{CTR}(R)]}{P(A=1)} \right. \\
&\qquad \left. - \frac{P(A=2) \sum_R \pi_2(R \mid q) \mathbb{E}[\text{CTR}(R)]}{P(A=2)} \right) \\
&= \sum_q P(q) \sum_R \mathbb{E}[\text{CTR}(R)] \big(\pi_1(R \mid q) - \pi_2(R \mid q) \big) \\
&= \mathbb{E}[\text{CTR}(\pi_1)] - \mathbb{E}[\text{CTR}(\pi_2)] = \Delta(\pi_1, \pi_2).
\end{aligned} \tag{9}$$

Variance is harder to evaluate without knowledge of π_1 and π_2. Unless $\Delta(\pi_1, \pi_2) = 0$, some variance is unavoidable since A/B testing alternates between estimating $\text{CTR}(\pi_1)$ and $\text{CTR}(\pi_2)$.

3.2 Interleaving

Interleaving methods were introduced specifically for evaluation in ranking, as a more efficient alternative to A/B testing [11]. After a query is issued, they take the rankings of two competing ranking systems and combine them into a single interleaved ranking. Any clicks on the interleaved ranking can be interpreted as a preference signal between either ranking system. Thus, unlike A/B testing, interleaving does not estimate the CTR of individual systems but a relative preference; the idea is that this allows it to be more efficient than A/B testing.

Each interleaving method attempts to use randomization to counter position bias, without deviating too much from the original rankings so as to maintain the user experience [11]. *Team-draft interleaving* (TDI) randomly selects one ranker to place their top document first, then the other ranker places their top (unplaced) document next [20]. Then it randomly decides the next two documents, and this process is repeated until all documents are placed in the interleaved ranking. Clicks on the documents are attributed to the ranker that placed them. The ranker with the most attributed clicks is inferred to be preferred by the user. *Probabilistic interleaving* (PI) treats each ranking as a probability distribution over documents; at each rank a distribution is randomly selected and a document is drawn from it [9]. After clicks have been received, probabilistic interleaving computes the expected number of clicks documents per ranking system to infer preferences. *Optimized interleaving* (OI) casts the randomization as an optimization problem, and displays rankings so that if all documents are equally relevant no preferences are found [19].

While every interleaving method attempts to deal with position bias, none is unbiased according to our definition (Section 2.2). This may be confusing because previous work on interleaving makes claims of unbiasedness [9, 10, 19]. However, they use different definitions of the term. More precisely, TDI, PI, and OI provably converge on the correct outcome if all documents are equally relevant [9, 10, 19, 20]. Moreover, if one assumes binary relevance and π_1 ranks all relevant documents equal to or higher than π_2, the binary outcome of PI and OI is proven to be correct in expectation [10, 19]. However, beyond the confines of these unambiguous cases, we can prove that these methods do not meet our definition of unbiasedness: for every method one can construct an example where it converges on the incorrect outcome. The rankers π_1, π_2 and position bias parameters θ can be chosen so that in expectation the wrong (binary) outcome is estimated; see Appendix A for a proof for each of the three interleaving methods. Thus, while more efficient than A/B testing, interleaving methods make systematic errors in certain circumstances and thus should not be considered to be unbiased w.r.t. CTR differences.

We note that the magnitude of the bias should also be considered. If the systematic error of an interleaving method is minuscule while the efficiency gains are very high, it may still be very useful in practice. Our experimental results (Section 6.2) reveal that the systematic error of all interleaving methods becomes very high when comparing systems with a CTR difference of 1% or smaller.

3.3 Counterfactual Evaluation

Counterfactual evaluation is based on the idea that if certain biases can be estimated well, they can also be adjusted [12, 22]. While estimating relevance is considered the core difficulty of ranking evaluation, estimating the position bias terms θ is very doable. By randomizing rankings, e.g., by swapping pairs of documents [12] or exploiting data logged during A/B testing [1], differences in CTR for the same item on different positions can be observed directly. Alternatively, using Expectation Maximization (EM) optimization [23] or a dual learning objective [2], position bias can be estimated from logged data as well. Once the bias terms θ have been estimated, logged clicks can be weighted so as to correct for the position bias during logging. Hence, counterfactual evaluation can work with historically logged data. Existing counterfactual evaluation algorithms do not dictate which rankings should be displayed during logging: they do not perform interventions and thus we do not consider them to be online methods.

Counterfactual evaluation assumes that the position bias θ and the logging policy π_0 are known, in order to correct for both position bias and item-selection bias. Clicks are gathered with π_0 which decides which rankings are displayed to the user. We follow Oosterhuis and de Rijke [16] and use as propensity scores the probability of observance in expectation over the displayed rankings:

$$\begin{aligned}
\rho(d \mid q) &= \mathbb{E}_R \big[P(o(d) = 1 \mid R) \mid \pi_0 \big] \\
&= \textstyle\sum_R \pi_0(R \mid q) P(o(d) = 1 \mid R).
\end{aligned} \tag{10}$$

Then we use $\lambda(d \mid \pi_1, \pi_2)$ to indicate the difference in observance probability under π_1 or π_2:

$$\lambda(d \mid \pi_1, \pi_2) = \mathbb{E}_R \big[P(o(d) = 1 \mid R) \mid \pi_1 \big] - \mathbb{E}_R \big[P(o(d) = 1 \mid R) \mid \pi_2 \big]$$

$$= \sum_R \theta_{\text{rank}(d|R)} \big(\pi_1(R \mid q_i) - \pi_2(R \mid q_i)\big). \tag{11}$$

Then, the IPS estimate function is formulated as:

$$x_i = f_{\text{IPS}}(q_i, R_i, c_i) = \sum_{d:\rho(d|q_i)>0} \frac{c_i(d)}{\rho(d|q_i)} \lambda(d \mid \pi_1, \pi_2). \tag{12}$$

Each click is weighted inversely to its examination probability, but items with a zero probability: $\rho(d|q_i) = 0$ are excluded. We note that these items can never be clicked: $\forall q, d(\rho(d|q) = 0 \rightarrow c(d) = 0)$. Before we prove unbiasedness, we note that given $\rho(d|q_i) > 0$:

$$\begin{aligned} \mathbb{E}\left[\frac{c(d)}{\rho(d|q)}\right] &= \frac{\sum_R \pi_0(R \mid q)\theta_{\text{rank}(d|R)}\zeta_{d,q}}{\rho(d|q_i)} \\ &= \frac{\sum_R \pi_0(R \mid q)\theta_{\text{rank}(d|R)}}{\sum_{R'} \pi_0(R' \mid q)\theta_{\text{rank}(d|R')}} \zeta_{d,q} = \zeta_{d,q}. \end{aligned} \tag{13}$$

This, in turn, can be used to prove unbiasedness:

$$\begin{aligned} \mathbb{E}[f_{\text{IPS}}(\ldots)] &= \sum_q P(q) \sum_{d:\rho(d|q_i)>0} \zeta_{d,q} \lambda(d \mid \pi_1, \pi_2) \\ &= \mathbb{E}[\text{CTR}(\pi_1)] - \mathbb{E}[\text{CTR}(\pi_2)] = \Delta(\pi_1, \pi_2). \end{aligned} \tag{14}$$

This proof is only valid under the following requirement:

$$\forall d, q(\zeta_{d,q}\lambda(d \mid \pi_1, \pi_2) > 0 \rightarrow \rho(d|q) > 0). \tag{15}$$

In practice, this means that the items in the top-k of either π_1 or π_2 need to have a non-zero examination probability under π_0, i.e., they must have a chance to appear in the top-k under π_0.

Besides Requirement 15 the existing counterfactual method [12, 22] is completely indifferent to π_0 and hence we do not consider it to be an online method. In the next section, we will introduce an algorithm for choosing and updating π_0 during logging to minimize the variance of the estimator. By doing so we turn counterfactual evaluation into an online method.

4 LOGGING POLICY OPTIMIZATION FOR VARIANCE MINIMIZATION

Next, we introduce a method aimed at finding a logging policy for the counterfactual estimator that minimizes its variance.

4.1 Minimizing Variance

In Section 3.3, we have discussed counterfactual evaluation and established that it is unbiased as long as θ is known and the logging policy meets Requirement 15. The variance of Δ_{IPS} depends on the position bias θ, the conditional click probabilities ζ, and the logging policy π_0. In contrast with the user-dependent θ and ζ, the way data is logged by π_0 is something one can have control over. The goal of our method is to find the optimal policy that minimizes variance while still meeting Requirement 15:

$$\pi_0^* = \underset{\pi_0:\pi_0 \text{ meets Req. 15}}{\text{argmin}} \text{Var}\big(\hat{\Delta}_{IPS}^{\pi_0}\big), \tag{16}$$

where $\hat{\Delta}_{IPS}^{\pi_0}$ is the counterfactual estimator based on data logged using π_0.

To formulate the variance, we first note that it is an expectation over queries:

$$\text{Var}(\hat{\Delta}) = \sum_q P(q) \text{Var}(\hat{\Delta} \mid q). \tag{17}$$

To keep notation short, for the remainder of this section we will write: $\Delta = \Delta(\pi_1, \pi_2)$; $\theta_{d,R} = \theta_{\text{rank}}(d \mid R)$; $\zeta_d = \zeta_{d,q}$; $\lambda_d = \lambda(d \mid \pi_1, \pi_2)$; and $\rho_d = \rho(d \mid q, \pi_0)$. Next, we consider the probability of a click pattern c, this is simply a possible combination of clicked documents $c(d) = 1$ and not-clicked documents $c(d) = 0$:

$$\begin{aligned} P(c \mid q) &= \sum_R \pi_0(R \mid q) \prod_{d:c(d)=1} \theta_{d,R}\zeta_d \prod_{d:c(d)=0} (1 - \theta_{d,R}\zeta_d) \\ &= \sum_R \pi_0(R \mid q) P(c \mid R). \end{aligned} \tag{18}$$

Here, π_0 has some control over this probability: by deciding the distribution of displayed rankings it can make certain click patterns more or less frequent. The variance added per query is the squared error of every possible click pattern weighted by the probability of each pattern. Let $\sum_c$ sum over every possible click pattern:

$$\text{Var}(\hat{\Delta}_{IPS}^{\pi_0} \mid q) = \sum_c P(c \mid q) \left(\Delta - \sum_{d:c(d)=1} \frac{\lambda_d}{\rho_d}\right)^2. \tag{19}$$

It is unknown whether there is a closed-form solution for π_0^*. However, the variance function is differentiable. Taking the derivative reveals a trade-off between two potentially conflicting goals:

$$\begin{aligned} \frac{\delta}{\delta \pi_0} \text{Var}(\hat{\Delta}_{IPS}^{\pi_0} \mid q) = \sum_c \overbrace{\left[\frac{\delta}{\delta \pi_0} P(c \mid q)\right] \left(\Delta - \sum_{d:c(d)=1} \frac{\lambda_d}{\rho_d}\right)^2}^{\text{minimize frequency of high-error click patterns}} \\ \underbrace{+ P(c \mid q) \left[\frac{\delta}{\delta \pi_0} \left(\Delta - \sum_{d:c(d)=1} \frac{\lambda_d}{\rho_d}\right)^2\right]}_{\text{minimize error of frequent click patterns}}. \end{aligned} \tag{20}$$

On the one hand, the derivative reduces the frequency of click patterns that result in high error samples, i.e., by updating π_0 so that these are less likely to occur. On the other hand, changing π_0 also affects the propensities ρ_d, i.e., if π_0 makes an item d less likely to be examined, its corresponding value λ_d / ρ_d becomes larger, which can lead to a higher error for related click patterns. The optimal policy has to balance: (i) avoiding showing rankings that lead to high-error click patterns; and (ii) avoiding minimizing propensity scores, which increases the errors of corresponding click patterns.

Our method applies stochastic gradient descent to optimize the logging policy w.r.t. the variance. There are two main difficulties with this approach: (i) the parameters θ and ζ are unknown a priori; and (ii) the gradients include summations over all possible rankings and all possible click patterns, both of which are computationally infeasible. In the following sections, we will detail how LogOpt solves both of these problems.

4.2 Bias & Relevance Estimation

In order to compute the gradient in Eq. 20, the parameters θ and ζ have to be known. LogOpt is based on the assumption that accurate estimates of θ and ζ suffice to find a near-optimal logging policy. We note that the counterfactual estimator only requires θ to be known for unbiasedness (see Section 3.3). Our approach is as follows, at given intervals during evaluation we use the available clicks to estimate θ and ζ. Then we use the estimated $\hat{\theta}$ to get the current estimate

$\hat{\Delta}_{IPS}(\mathcal{I},\hat{\theta})$ (Eq. 12) and optimize w.r.t. the estimated variance (Eq. 19) based on $\hat{\theta}$, $\hat{\zeta}$, and $\hat{\Delta}_{IPS}(\mathcal{I},\hat{\theta})$.

For estimating θ and ζ we use the existing EM approach by Wang et al. [23], because it works well in situations where few interactions are available and does not require randomization. We note that previous work has found randomization-based approaches to be more accurate for estimating θ [1, 7, 23]. However, they require multiple interactions per query and specific types of randomization in their results, by choosing the EM approach we do avoid having these requirements.

4.3 Monte-Carlo-Based Derivatives

Both the variance (Eq. 19) and its gradient (Eq. 20), include a sum over all possible click patterns. Moreover, they also include the probability of a specific pattern $P(c \mid q)$ that is based on a sum over all possible rankings (Eq. 18). Clearly, these equations are infeasible to compute under any realistic time constraints. To solve this issue, we introduce gradient estimation based on Monte-Carlo sampling. Our approach is similar to that of Ma et al. [15], however, we are estimating gradients of variance instead of general performance.

First, we assume that policies place the documents in order of rank and the probability of placing an individual document at rank x only depends on the previously placed documents. Let $R_{1:x-1}$ indicate the (incomplete) ranking from rank 1 up to rank x, then $\pi_0(d \mid R_{1:x-1},q)$ indicates the probability that document d is placed at rank x given that the ranking up to x is $R_{1:x-1}$. The probability of a ranking R up to rank k is thus:

$$\pi_0(R_{1:k} \mid q) = \prod_{x=1}^{k} \pi_0(R_x \mid R_{1:x-1},q). \tag{21}$$

Let K be the length of a complete ranking R, the gradient of the probability of a ranking w.r.t. a policy is:

$$\frac{\delta \pi_0(R \mid q)}{\delta \pi_0} = \sum_{x=1}^{K} \frac{\pi_0(R \mid q)}{\pi_0(R_x \mid R_{1:x},q)} \left[\frac{\delta \pi_0(R_x \mid R_{1:x-1},q)}{\delta \pi_0} \right]. \tag{22}$$

The gradient of the propensity w.r.t. the policy (cf. Eq. 10) is:

$$\frac{\delta \rho(d|q)}{\delta \pi_0} = \sum_{k=1}^{K} \theta_k \sum_{R} \pi_0(R_{1:k-1}|q) \left(\left[\frac{\delta \pi_0(d|R_{1:k-1},q)}{\delta \pi_0} \right] + \sum_{x=1}^{k-1} \frac{\pi_0(d|R_{1:k-1},q)}{\pi_0(R_x|R_{1:x-1},q)} \left[\frac{\delta \pi_0(R_x|R_{1:x-1},q)}{\delta \pi_0} \right] \right). \tag{23}$$

To avoid iterating over all rankings in the $\sum_R$ sum, we sample M rankings: $R^m \sim \pi_0(R \mid q)$, and a click pattern on each ranking: $c^m \sim P(c \mid R^m)$. This enables us to make the following approximation:

$$\widehat{\rho\text{-grad}}(d) = \frac{1}{M} \sum_{m=1}^{M} \sum_{k=1}^{K} \theta_k \left(\left[\frac{\delta \pi_0(d|R^m_{1:k-1},q)}{\delta \pi_0} \right] + \sum_{x=1}^{k-1} \frac{\pi_0(d|R^m_{1:k-1},q)}{\pi_0(R^m_x|R^m_{1:x-1},q)} \left[\frac{\delta \pi_0(R^m_x|R^m_{1:x-1},q)}{\delta \pi_0} \right] \right), \tag{24}$$

since $\frac{\delta \rho(d|q)}{\delta \pi_0} \approx \widehat{\rho\text{-grad}}(d,q)$. In turn, we can use this to approximate the second part of Eq. 20:

$$\widehat{\text{error-grad}}(c) = 2 \left(\Delta - \sum_{d:c(d)=1} \frac{\lambda_d}{\rho_d} \right) \sum_{d:c(d)=1} \frac{\lambda_d}{\rho_d^2} \widehat{\rho\text{-grad}}(d), \tag{25}$$

we approximate the first part of Eq. 20 with:

$$\widehat{\text{freq-grad}}(R,c) = \left(\Delta - \sum_{d:c(d)=1} \frac{\lambda_d}{\rho_d} \right)^2 \sum_{x=1}^{K} \frac{1}{\pi_0(R_x \mid R_{1:x-1},q)} \left[\frac{\delta \pi_0(R_x \mid R_{1:x-1},q)}{\delta \pi_0} \right]. \tag{26}$$

Together, they approximate the complete gradient (cf. Eq. 20):

$$\frac{\delta \text{Var}(\hat{\Delta}^{\pi_0}_{IPS} \mid q)}{\delta \pi_0} \approx \frac{1}{M} \sum_{m=1}^{M} \widehat{\text{freq-grad}}(R^m,c^m) + \widehat{\text{error-grad}}(c^m). \tag{27}$$

Therefore, we can approximate the gradient of the variance w.r.t. a logging policy π_0, based on rankings sampled from π_0 and our current estimated click model $\hat{\theta}$, $\hat{\zeta}$, while staying computationally feasible.[1]

4.4 Summary

We have summarized the LogOpt method in Algorithm 1. The algorithm requires a set of historical interactions $\mathcal{I}$ and two rankers π_1 and π_2 to compare. Then by fitting a click model on $\mathcal{I}$ using an EM-procedure (Line 2) an estimate of observation bias $\hat{\theta}$ and document relevance $\hat{\zeta}$ is obtained. Using $\hat{\theta}$, an estimate of the difference in observation probabilities $\hat{\lambda}$ is computed (Line 3 and cf. Eq 11), and an estimate of the CTR difference $\hat{\Delta}(\pi_1,\pi_2)$ (Line 4 and cf. Eq 12). Then the optimization of a new logging policy π_0 begins: A query is sampled from $\mathcal{I}$ (Line 7), and for that query M rankings are sampled from the current π_0 (Line 8), then for each ranking a click pattern is sampled using $\hat{\theta}$ and $\hat{\zeta}$ (Line 9). Finally, using the sampled rankings and clicks, $\hat{\theta}$, $\hat{\lambda}$, and $\hat{\Delta}(\pi_1,\pi_2)$, the gradient is now approximated using Eq. 27 (Line 10) and the policy π_0 is updated accordingly (Line 11). This process can be repeated for a fixed number of steps, or until the policy has converged.

This concludes our introduction of LogOpt: the first method that optimizes the logging policy for faster convergence in counterfactual evaluation. We argue that LogOpt turns counterfactual evaluation into online evaluation, because it instructs which rankings should be displayed for the most efficient evaluation. The ability to make interventions like this is the defining characteristic of an online evaluation method.

5 EXPERIMENTAL SETUP

We ran semi-synthetic experiments that are prevalent in online and counterfactual evaluation [9, 13, 16]. User-issued queries are simulated by sampling from learning to rank datasets; each dataset contains a preselected set of documents per query. We use Yahoo! Webscope [3] and MSLR-WEB30k [18]; they both contain 5-grade relevance judgements for all preselected query-document pairs. For each sampled query, we let the evaluation method decide which ranking to display and then simulate clicks on them using probabilistic click models.

[1] For a more detailed description see Appendix B in the supplementary material.

Algorithm 1 Logging-Policy Optimization Algorithm (LogOpt)

Input: Historical interactions: $\mathcal{I}$; rankers to compare π_1, π_2.
$\hat{\theta}, \hat{\zeta} \leftarrow$ infer_click_model($\mathcal{I}$) *// estimate bias using EM*
$\hat{\lambda} \leftarrow$ estimated_observance($\hat{\theta}, \pi_1, \pi_2$) *// estimate λ cf. Eq 11*
$\hat{\Delta}(\pi_1, \pi_2) \leftarrow$ estimated_CTR($\mathcal{I}, \hat{\lambda}, \hat{\theta}$) *// CTR diff. cf. Eq 12*
$\pi_0 \leftarrow$ init_policy() *// initialize logging policy*
for $j \in \{1,2,...\}$ **do**
  $q \sim P(q \mid \mathcal{I})$ *// sample a query from interactions*
  $\mathcal{R} \leftarrow \{R^1, R^2, ..., R^M\} \sim \pi_0(R \mid q)$ *// sample M rankings*
  $\mathcal{C} \leftarrow \{c^1, c^2, ..., c^M\} \sim P(c \mid \mathcal{R})$ *// sample M click patterns*
  $\hat{\delta} \leftarrow$ approx_grad($\mathcal{R}, \mathcal{C}, \hat{\lambda}, \hat{\theta}, \hat{\Delta}(\pi_1, \pi_2)$) *// using Eq. 27*
  $\pi_0 \leftarrow$ update($\pi_0, \hat{\delta}$) *// update using approx. gradient*
return π_0

To simulate position bias, we use the rank-based probabilities of Joachims et al. [13]:

$$P(o(d) = 1 \mid R, q) = \frac{1}{\text{rank}(d \mid R)}. \tag{28}$$

If observed, the click probability is determined by the relevance label of the dataset (ranging from 0 to 4). More relevant items are more likely to be clicked, yet non-relevant documents still have a non-zero click probability:

$$P(c(d) = 1 \mid o(d) = 1, q) = 0.225 \cdot \text{relevance\_label}(q, d) + 0.1. \tag{29}$$

Spread over both datasets, we generated 2,000 rankers and created 1,000 ranker-pairs. We aimed to generate rankers that are likely to be compared in real-world scenarios; unfortunately, no simple distribution of such rankers is available. Therefore, we tried to generate rankers that have (at least) a decent CTR and that span a variety of ranking behaviors. Each ranker was optimized using LambdaLoss [24] based on the labelled data of 100 sampled queries; each ranker is based on a linear model that only uses a random sample of 50% of the dataset features. Figure 1 displays the resulting CTR distribution; it appears to follow a normal distribution.

For each ranker-pair and method, we sample $3 \cdot 10^6$ queries and calculate their CTR estimates for different numbers of queries. We considered three metrics: (i) The binary error: whether the estimate correctly predicts which ranker should be preferred. (ii) The absolute error: the absolute difference between the estimate and the true $\mathbb{E}[\text{CTR}]$ difference:

$$\text{absolute-error} = |\Delta(\pi_1, \pi_2) - \hat{\Delta}(\mathcal{I})|. \tag{30}$$

And (iii) the mean squared error: the squared error *per sample* (not the final estimate); if the estimator is unbiased this is equivalent to the variance:

$$\text{mean-squared-error} = \frac{1}{N} \sum_{i=1}^{N} (\Delta(\pi_1, \pi_2) - x_i)^2. \tag{31}$$

We compare LogOpt with the following baselines: (i) A/B testing (with equal probabilities for each ranker), (ii) Team-Draft Interleaving, (iii) Probabilistic Interleaving (with $\tau = 4$), and (iv) Optimized Interleaving (with the inverse rank scoring function). Furthermore, we compare LogOpt with other choices of logging policies: (i) uniform sampling, (ii) A/B testing: showing either the ranking of A or B with equal probability, and (iii) an Oracle logging policy: applying LogOpt to the true relevances ζ and position bias θ. We also consider

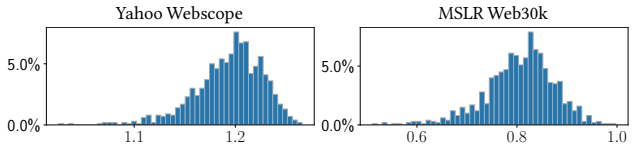

Figure 1: The CTR distribution of the 2000 generated rankers, 1000 were generated per dataset.

LogOpt both in the case where θ is known *a priori*, or where it has to be estimated still. Because estimating θ and optimizing the logging policy π_0 is time-consuming, we only update $\hat{\theta}$ and π_0 after 10^3, 10^4, 10^5 and 10^6 queries. The policy LogOpt optimizes uses a neural network with 2 hidden layers consisting of 32 units each. The network computes a score for every document, then a softmax is applied to the scores to create a distribution over documents.

6 RESULTS

Our results are displayed in Figures 2, 3, and 4. Figure 2 shows the results comparing LogOpt with other online evaluation methods; Figure 3 compares LogOpt with counterfactual evaluation using other logging policies; and finally, Figure 4 shows the distribution of binary errors for each method after $3 \cdot 10^6$ sampled queries.

6.1 Performance of LogOpt

In Figure 2 we see that, unlike interleaving methods, counterfactual evaluation with LogOpt continues to decrease both its binary error and its absolute error as the number of queries increases. While interleaving methods converge at a binary error of at least 2.2% and an absolute error greater than 0.01, LogOpt appears to converge towards zero errors for both. This is expected as LogOpt is proven to be unbiased when the position bias is known. Interestingly, we see similar behavior from LogOpt with estimated position bias. Both when bias is known or estimated, LogOpt has a lower error than the interleaving methods after $2 \cdot 10^3$ queries. Thus we conclude that interleaving methods converge faster and have an initial period where their error is lower, but are biased. In contrast, by being unbiased, LogOpt converges on a lower error eventually.

If we use Figure 2 to compare LogOpt with A/B testing, we see that on both datasets LogOpt has a considerably smaller mean squared error. Since both methods are unbiased, this means that LogOpt has a much lower variance and thus is expected to converge faster. On the Yahoo dataset we observe this behavior, both in terms of binary error and absolute error and regardless of whether the bias is estimated, LogOpt requires half as much data as A/B testing to reach the same level or error. Thus, on Yahoo LogOpt is roughly twice as data-efficient than A/B testing. On the MSLR dataset it is less clear whether LogOpt is noticeably more efficient: after 10^4 queries the absolute error of LogOpt is twice as high, but after 10^5 queries it has a lower error than A/B testing. We suspect that the relative drop in performance around 10^4 queries is due to LogOpt overfitting on incorrect $\hat{\zeta}$ values, however, we were unable to confirm this. Hence, LogOpt is just as efficient as, or even more efficient than, A/B testing, depending on the circumstances.

Finally, when we use Figure 3 to compare LogOpt with other logging policy choices, we see that LogOpt mostly approximates the optimal Oracle logging policy. In contrast, the uniform logging policy

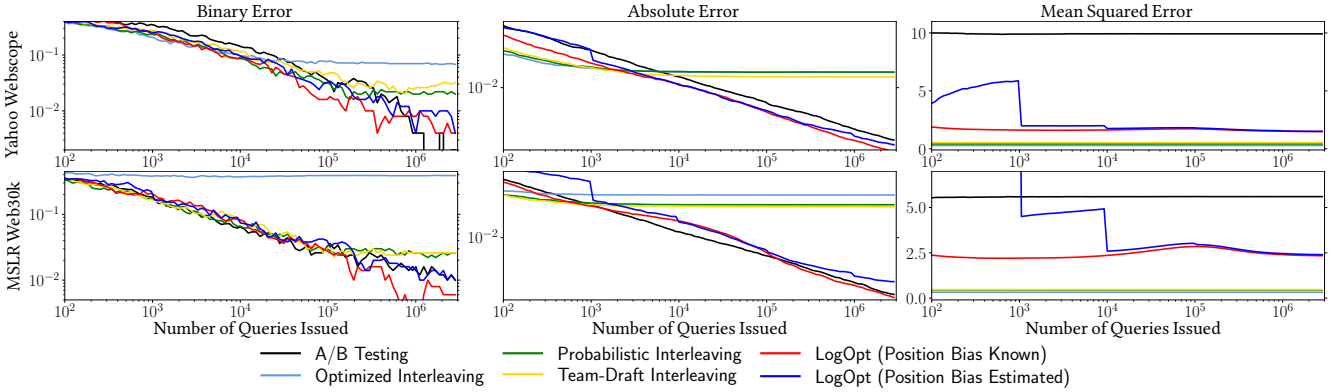

Figure 2: Comparison of LogOpt with other online methods; displayed results are an average over 500 comparisons.

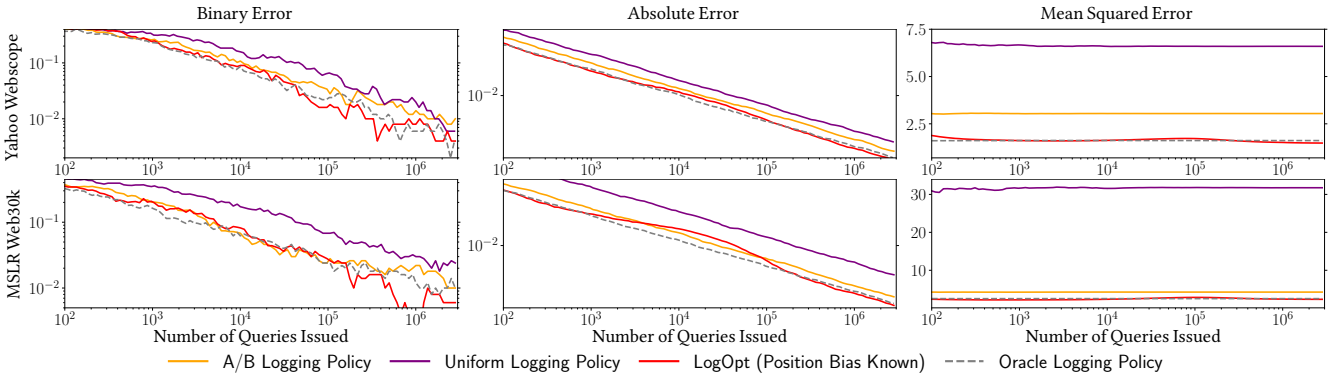

Figure 3: Comparison of logging policies for counterfactual evaluation; displayed results are an average over 500 comparisons.

is very data-inefficient on both datasets it requires around ten times the number of queries to reach the same level or error as LogOpt. The A/B logging policy is a better choice than the uniform logging policy, but apart from the dip in performance on the MSLR dataset, it appears to require twice as many queries as LogOpt. Interestingly, the performance of LogOpt is already near the Oracle when only 10^2 queries have been issued. With such a small number of interactions, accurately estimating the relevances ζ should not be possible, thus it appears that in order for LogOpt to find an efficient logging policy the relevances ζ are not important. This must mean that only the differences in behavior between the rankers (i.e. λ) have to be known for LogOpt to be efficient. Overall, these results show that LogOpt can greatly increase the efficiency of counterfactual estimation.

6.2 Bias of Interleaving

Our results in Figure 2 clearly illustrate the bias of interleaving methods: each of them systematically infers incorrect preferences in (at least) 2.2% of the ranker-pairs. These errors are systematic since increasing the number of queries from 10^5 to $3{\cdot}10^6$ does not remove any of them. Additionally, the combination of the lowest mean-squared-error with a worse absolute error than A/B testing after 10^4 queries, indicates that interleaving results in a low variance at the cost of bias. To better understand when these systematic errors occur, we show the distribution of binary errors w.r.t. the CTR differences of the associated ranker-pairs in Figure 4. Here we see that most errors occur on ranker-pairs where the CTR difference is smaller than 1%, and that of all comparisons the percentage of errors greatly increases as the CTR difference decreases below 1%. This suggests that interleaving methods are unreliable to detect preferences when differences are 1% CTR or less.

It is hard to judge the impact this bias may have in practice. On the one hand, a 1% CTR difference is far from negligible: generally a 1% increase in CTR is considered an impactful improvement in the industry. On the other hand, our results are based on a single click model with specific values for position bias and conditional click probabilities. While our results strongly prove interleaving is biased, we should be careful not to generalize the size of the observed systematic error to all other ranking settings.

Previous work has performed empirical studies to evaluate various interleaving methods with real users. Chapelle et al. [4] applied interleaving methods to compare ranking systems for three different search engines, and found team-draft interleaving to highly correlate with absolute measures such as CTR. However, we note that in this study no more than six rankers were compared, thus such a study would likely miss a systematic error of 2.2%. In fact, Chapelle et al. [4] note themselves that they cannot confidently claim team-draft interleaving is completely unbiased. Schuth et al. [21] performed a larger comparison involving 38 ranking systems, but again, too small to reliably detect a small systematic error.

It appears that the field is missing a large scale comparison that involves a large enough number of rankers to observe small systematic errors. If such an error is found, the next step is to identify if certain types of ranking behavior are erroneously and systematically disfavored. While these questions remain unanswered, we are

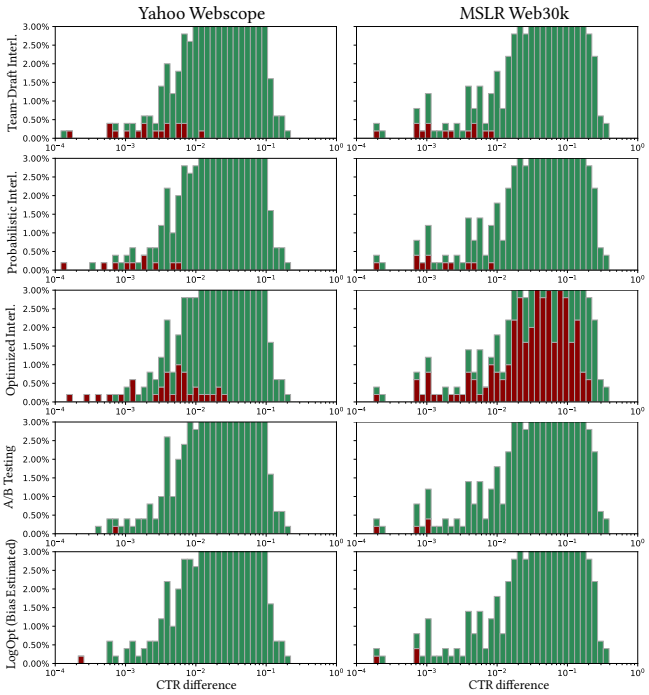

Figure 4: Distribution of errors over the CTR differences of the rankers in the comparison; red indicates a binary error; green indicates a correctly inferred binary preference; results are on estimates based on $3 \cdot 10^6$ sampled queries.

concerned that the claims of unbiasedness in previous interleaving work (see Section 3.2) give practitioners an unwarranted sense of reliability in interleaving.

7 CONCLUSION

In this paper, we have introduced the Logging-Policy Optimization Algorithm (LogOpt): the first method that optimizes a logging policy for minimal variance counterfactual evaluation. Counterfactual evaluation is proven to be unbiased w.r.t. position bias and item-selection bias under a wide range of logging policies. With the introduction of LogOpt, we now have an algorithm that can decide which rankings should be displayed for the fastest convergence. Therefore, we argue that LogOpt turns the existing counterfactual evaluation approach – which is indifferent to the logging policy – into an online approach – which instructs the logging policy.

Our experimental results show that LogOpt can lead to a better data-efficiency than A/B testing, without introducing the bias of interleaving. While our findings are mostly theoretical, they do suggest that future work should further investigate the bias in interleaving methods. Our results suggest that all interleaving methods make systematic errors, in particular when rankers with a similar CTR are compared. Furthermore, to the best of our knowledge, no empirical studies have been performed that could measure such a bias, our findings strongly show that such a study would be highly valuable to the field. Finally, LogOpt shows that in theory an evaluation method that is both unbiased and efficient is possible, if future work finds that these theoretical findings match empirical results with real users, this could be the start of a new line of theoretically-justified online evaluation methods.

Acknowledgements

We want to thank the anonymous reviewers for their feedback. This research was partially supported by the Netherlands Organisation for Scientific Research (NWO) under project nr 612.001.551 and by the Innovation Center for AI (ICAI). All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

Reproducibility

Our experimental implementation is publicly available at https://github.com/HarrieO/2020ictir-evaluation.

REFERENCES

[1] Aman Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Li, Marc Najork, and Thorsten Joachims. 2019. Estimating Position Bias without Intrusive Interventions. In *WSDM*. ACM, 474–482.
[2] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W Bruce Croft. 2018. Unbiased Learning to Rank with Unbiased Propensity Estimation. In *SIGIR*. ACM, 385–394.
[3] Olivier Chapelle and Yi Chang. 2011. Yahoo! Learning to Rank Challenge Overview. *Journal of Machine Learning Research* 14 (2011), 1–24.
[4] Olivier Chapelle, Thorsten Joachims, Filip Radlinski, and Yisong Yue. 2012. Large-scale Validation and Analysis of Interleaved Search Evaluation. *ACM Transactions on Information Systems (TOIS)* 30, 1 (2012), 1–41.
[5] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2015. *Click Models for Web Search*. Morgan & Claypool Publishers.
[6] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An Experimental Comparison of Click Position-bias Models. In *WSDM*. 87–94.
[7] Zhichong Fang, Aman Agarwal, and Thorsten Joachims. 2019. Intervention Harvesting for Context-dependent Examination-bias Estimation. In *SIGIR*. 825–834.
[8] Katja Hofmann, Lihong Li, and Filip Radlinski. 2016. Online Evaluation for Information Retrieval. *Foundations and Trends in Information Retrieval* 10, 1 (2016), 1–117.
[9] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. 2011. A Probabilistic Method for Inferring Preferences from Clicks. In *CIKM*. ACM, 249–258.
[10] Katja Hofmann, Shimon Whiteson, and Maarten De Rijke. 2013. Fidelity, Soundness, and Efficiency of Interleaved Comparison Methods. *ACM Transactions on Information Systems (TOIS)* 31, 4 (2013), 1–43.
[11] Thorsten Joachims. 2003. Evaluating Retrieval Performance Using Clickthrough Data. In *Text Mining*, J. Franke, G. Nakhaeizadeh, and I. Renz (Eds.). Physica Verlag.
[12] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately Interpreting Clickthrough Data as Implicit Feedback. In *SIGIR*. ACM, 154–161.
[13] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased Learning-to-Rank with Biased Feedback. In *WSDM*. ACM, 781–789.
[14] Ron Kohavi and Roger Longbotham. 2017. Online Controlled Experiments and A/B Testing. *Encyclopedia of Machine Learning and Data Mining* 7, 8 (2017), 922–929.
[15] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Ji Yang, Minmin Chen, Jiaxi Tang, Lichan Hong, and Ed H Chi. 2020. Off-policy Learning in Two-stage Recommender Systems. In *The Web Conference 2020*. ACM, 463–473.
[16] Harrie Oosterhuis and Maarten de Rijke. 2020. Policy-Aware Unbiased Learning to Rank for Top-k Rankings. In *SIGIR*. ACM.
[17] Zohreh Ovaisi, Ragib Ahsan, Yifan Zhang, Kathryn Vasilaky, and Elena Zheleva. 2020. Correcting for Selection Bias in Learning-to-rank Systems. In *WWW*. 1863–1873.
[18] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 datasets. *arXiv preprint arXiv:1306.2597* (2013).
[19] Filip Radlinski and Nick Craswell. 2013. Optimized Interleaving for Online Retrieval Evaluation. In *WSDM*. ACM, 245–254.
[20] Filip Radlinski, Madhu Kurup, and Thorsten Joachims. 2008. How Does Clickthrough Data Reflect Retrieval Quality?. In *CIKM*. ACM, 43–52.
[21] Anne Schuth, Katja Hofmann, and Filip Radlinski. 2015. Predicting Search Satisfaction Metrics with Interleaved Comparisons. In *SIGIR*. 463–472.
[22] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to Rank with Selection Bias in Personal Search. In *SIGIR*. ACM, 115–124.
[23] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position Bias Estimation for Unbiased Learning to Rank in Personal Search. In *WSDM*. ACM, 610–618.
[24] Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2018. The LambdaLoss Framework for Ranking Metric Optimization. In *CIKM*. ACM, 1313–1322.