To Model or to Intervene: A Comparison of Counterfactual and Online Learning to Rank from User Interactions

Rolf Jagerman
University of Amsterdam
Amsterdam, The Netherlands
rolf.jagerman@uva.nl

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

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

ABSTRACT

Learning to Rank (LTR) from user interactions is challenging as user feedback often contains high levels of bias and noise. At the moment, two methodologies for dealing with bias prevail in the field of LTR: *counterfactual* methods that learn from historical data and model user behavior to deal with biases; and *online* methods that perform interventions to deal with bias but use no explicit user models. For practitioners the decision between either methodology is very important because of its direct impact on end users. Nevertheless, there has never been a direct comparison between these two approaches to unbiased LTR. In this study we provide the first benchmarking of both counterfactual and online LTR methods under different experimental conditions. Our results show that the choice between the methodologies is consequential and depends on the presence of selection bias, and the degree of position bias and interaction noise. In settings with little bias or noise counterfactual methods can obtain the highest ranking performance; however, in other circumstances their optimization can be detrimental to the user experience. Conversely, online methods are very robust to bias and noise but require control over the displayed rankings. Our findings confirm and contradict existing expectations on the impact of model-based and intervention-based methods in LTR, and allow practitioners to make an informed decision between the two methodologies.

CCS CONCEPTS

• **Information systems → Learning to rank**.

KEYWORDS

Learning to rank; Online learning; Counterfactual learning

ACM Reference Format:
Rolf Jagerman, Harrie Oosterhuis, and Maarten de Rijke. 2019. To Model or to Intervene: A Comparison of Counterfactual and Online Learning to Rank from User Interactions. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3331184.3331269

SIGIR '19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331269

1 INTRODUCTION

Interest in Learning to Rank (LTR) approaches that learn from user interactions has increased recently [4, 8, 16, 37]. Compared to learning from annotated datasets [19], implicit feedback obtained through user interactions matches user preferences more closely [14]. Furthermore, gathering interactions is much less costly than expert annotations [5, 20]. Additionally, unlike LTR from annotated datasets, LTR from user interactions can respect privacy-sensitive settings [4]. However, a big disadvantage of user interactions is that they often contain different types of bias and noise. Hence, LTR methods that learn from user interactions mainly focus on removing bias from the learning process [4, 16, 24].

There are two main families of algorithms for *unbiased* LTR from user interactions:

(1) Counterfactual Learning to Rank (CLTR) [16]: These algorithms learn a ranking model from a historical interaction log, often collected using a production system. They usually treat clicks as absolute relevance indicators and employ a form of re-weighing in order to debias interaction data. Counterfactual methods have no experimental control; they avoid the risks associated with online interventions where untested rankings may be displayed. A disadvantage is that they cannot explore and are limited to rankings displayed by the production system.
(2) Online Learning to Rank (OLTR) [37]: This class of algorithms interactively optimize and update a ranking model after every interaction. They combat bias by *interventions*, i.e., by displaying slightly modified rankings. This type of experimental control allows the learner to assess and learn novel rankings. Clearly, experimental control comes with a risk: untested rankings may hurt the user experience.

For practitioners the decision whether to use counterfactual or online LTR is important for practical deployment and user satisfaction with their ranking system. E.g., if there are situations where CLTR and OLTR methods provide the same performance, the risks of interventions can be avoided. However, if under some conditions CLTR methods are unable to reach the same performance as online interventions promise to bring, an OLTR method may be preferred. Currently, there has not been a study comparing methods across the two methodologies. As a result, it is currently unclear when which methods may be preferred, what benefits either methodology provides, and the scope of these benefits. Direct comparisons between CLTR and OLTR are required to help advance the field of LTR and inform its uptake.

A direct and fair comparison of counterfactual and online LTR algorithms is non-trivial for several reasons. First, CLTR methods do not affect the user experience as they learn from historical data; in

contrast, the user experience is a vital part of the evaluation of OLTR methods. Second, unlike OLTR methods, CLTR methods assume there is no selection bias, and proofs of their unbiasedness depend on this assumption. Finally, the optimization problems for OLTR and CLTR methods are formulated differently – therefore they may not be optimizing the same metrics and observed differences could be a consequence of this difference.

To the best of our knowledge, this is the first study to provide a direct comparison of CLTR and OLTR methods. Our main goal in this work is to answer the following question:

> *How should LTR practitioners choose which method to apply from either* counterfactual *or* online *LTR methodologies?*

In order to enable informed answers to this question, we address multiple aspects that are important to practitioners of both large-scale and small-scale LTR systems. First, we evaluate whether both approaches converge at the same level of performance, in other words, whether both approaches capture the true user preferences equally well. Furthermore, we investigate how the learning outcomes are affected by different levels of selection bias, position bias and interaction noise. Second, we evaluate how well the user experience is maintained during learning, since OLTR methods could potentially deter users with inappropriate interventions. Thirdly, we investigate the effect of interventions by allowing counterfactual methods to execute periodic deployments; this simulates multiple steps of optimization and deployment as one would see in practice.

The research questions we address are:

RQ1 Do state-of-the-art counterfactual and online LTR methods converge to the same level of performance?

RQ2 Is the user experience the same for counterfactual methods as for online methods?

RQ3 When do online interventions help the learning to rank algorithm?

In this work we present the first direct comparison between CLTR and OLTR methods. Our comparison leads to valuable insights as it reveals that, depending on the experimental conditions, a different methodology should be preferred. In particular, our results show that OLTR methods are more robust to selection bias, position bias and interaction noise. However, under low levels of bias and noise CLTR methods can obtain a significantly higher performance. Furthermore, to our surprise we find that some properties asserted to pertain to CLTR or OLTR methods in previously published work appear to be lacking when tested. For instance, in contrast with previously published expectations [24] OLTR is not substantially faster at learning than CLTR, and while always assumed to be safe [36], CLTR may be detrimental to the user experience when deployed under high-levels of noise.

Our findings reveal areas where future LTR work could make important advances, and moreover, allow practitioners to make an informed decision on which LTR methodology to apply.

2 COUNTERFACTUAL LEARNING TO RANK

Counterfactual Learning to Rank (CLTR) [1, 2, 16] aims to learn a ranking model offline from historical interaction data. Employing an offline approach has many benefits compared to an online one. First, it is possible to try and iterate many different learning algorithms without needing to deploy them online. Furthermore, it avoids the pitfalls and engineering overhead of having to deploy an online learning system. Finally, models that are learned offline can be tested before actually being deployed online, alleviating some of the safety concerns surrounding OLTR, such as the aggressive exploration of online methods that may place irrelevant items at high ranked positions [17, 36].

Table 1: Notation used throughout the paper.

Notation	Description
d	document
D	set of documents
R	ranked list
R_i	document placed at rank i in ranked list R
$f_\theta(\cdot)$	ranking model with parameters θ
c_i	1 if document at rank i was clicked, 0 otherwise
p_i	The propensity score at rank i

A straightforward approach to LTR from historical user interactions is to collect clicks and treat them as signals of relevance [13]. This is referred to as *partial information feedback* because it only conveys information about the documents that the user *has seen* and clicked on, but not other documents that the user *could have* seen and clicked on. Traditional supervised learning algorithms expect data to be in a "full information" form, where it is exactly known which documents are relevant and which ones are not. This is never the case in user interactions due to biases and noise. As a solution, CLTR provides a way to deal with partial information feedback.

2.1 Unbiased LTR with biased feedback

Joachims et al. [16] introduce a method to utilize interaction data in LTR, by casting the problem as a counterfactual learning problem [31]. In [16], it is assumed that the user does not examine all documents in a ranked list, and is more likely to observe documents at the top of the list than at the bottom; this is referred to as *position bias*. After a document is observed, a user will either judge it as relevant resulting in a click, or judge it as non-relevant. More formally, the user observes document d_i at rank i with some probability p_i, called the *propensity*.[1]

If the propensity is known, it is possible to modify an existing learning algorithm and simply re-weigh interaction data according to the propensity scores using *Inverse Propensity Scoring* (IPS). Joachims et al. [16] take the SVMRank algorithm and modify it to optimize a re-weighted objective, where each click is re-weighted according to whether the click appeared at the top of the ranked list (thus with high propensity) or lower in the ranked list (thus with lower propensity). Samples with high propensity are weighted less than samples with low propensity and vice versa. We will assume the propensities are known a priori and discuss related work dealing with propensity estimation in Section 2.3

To formalize CLTR, we consider a ranking function $f_{\text{production}}$ that produces a ranked list R to be shown to the user in response to a query q. When a user clicks on a document in this ranking, they are revealing to us that this document is relevant. We denote

[1] The notation we use in the paper is listed in Table 1.

Algorithm 1 Data collection for Counterfactual Learning to Rank.

1: **Input**: production ranker: $f_{\text{production}}$; log size n;
2: **Output**: a session log $\mathcal{L}$
3: $\mathcal{L} = \emptyset$
4: **for** $t \leftarrow 1 \dots n$ **do**
5: $q^{(t)} \leftarrow receive\_query(t)$
6: $D^{(t)} \leftarrow preselect\_documents(q^{(t)})$
7: $R^{(t)} \leftarrow rank\_documents(f_{\text{production}}, D^{(t)})$
8: $c^{(t)} \leftarrow receive\_clicks(R^{(t)})$
9: $\mathcal{L} \leftarrow \mathcal{L} \cup \{(R^{(t)}, c^{(t)})\}$

a user's clicks by a 0/1 vector c:

$$c_i = \begin{cases} 1 & \text{if document } d_i \text{ was observed and judged relevant,} \\ 0 & \text{otherwise.} \end{cases}$$

Note that it is possible for a user to click on more than one document during a session or click on no documents. Since a user is more likely to observe top-ranked documents than lower-ranked ones, we are more likely to observe relevance signals of the top-ranked documents. We denote the probability that a user observes the document at rank i with p_i; this is usually called the *propensity* of the observation.

We record a click log $\mathcal{L} = \{(R^{(j)}, c^{(j)})\}_{j=1}^{n}$, containing rankings of documents R and clicks c according to the procedure in Algorithm 1. For brevity we drop the superscript notation $\cdot^{(j)}$ for the session identifier. We now derive the learning objective of [16], a modified version of the SVMRank training objective that minimizes the average rank of relevant results, weighted by the inverse propensity scores:

$$\hat{\theta} = \operatorname*{argmin}_{\theta} \frac{1}{|\mathcal{L}|} \sum_{(R,c)\in\mathcal{L}} \sum_{\{i:c_i=1\}} \frac{\text{rank}(R_i \mid f_\theta)}{p_i}. \tag{1}$$

It can be shown that the above training objective is unbiased and can be solved via a hinge loss formulation [16]. We will refer to this method as *Counterfactual SVMRank* (CF-RANK).

2.2 Unbiased LTR with additive metrics

The counterfactual learning framework described in the previous section can be adapted to optimize additive metrics [1]. For example, we can modify the training objective so it optimizes DCG [10], a common metric in evaluation rankings:

$$\hat{\theta} = \operatorname*{argmin}_{\theta} \frac{1}{|\mathcal{L}|} \sum_{(R,c)\in\mathcal{L}} \sum_{\{i:c_i=1\}} \frac{\lambda(\text{rank}(R_i \mid f_\theta))}{p_i}, \tag{2}$$

where $\lambda(r) = \frac{-1}{\log(1+r)}$. This objective is both continuous and sub-differentiable, making it possible to solve using existing gradient descent techniques. We will refer to the DCG-optimizing counterfactual method as *Counterfactual DCGRank* (CF-DCG).

The counterfactual LTR algorithm is described in Algorithm 2. As input, the algorithm takes a set of weights (typically initialized to 0), a scoring function f, a learning rate μ and a click log $\mathcal{L}$. The algorithm runs for a fixed number of epochs which trades off computation time for convergence. The gradient is calculated on line 7 where a clicked document is compared against every other document. The gradient is computed as a λ-modified hinge loss: with $\lambda(r) = r$ this is the Counterfactual SVMRank (CF-RANK) method, which attempts to minimize the rank of relevant results; and with $\lambda(r) = \frac{-1}{\log(1+r)}$ we obtain the Counterfactual DCGRank (CF-DCG) method, which attempts to maximize DCG. Finally, the ranking model is updated via stochastic gradient descent on line 9.

Algorithm 2 Counterfactual Learning to Rank (CLTR).

1: **Input**: initial weights: θ; scoring function: f; learning rate μ; click log $\mathcal{L}$; number of epochs: E.
2: **for** $e \leftarrow 1 \dots E$ **do**
3: **for** $(R, c) \in \mathcal{L}$ **do**
4: **for** $R_i : c_i = 1$ **do**
5: $\nabla f_\theta \leftarrow \mathbf{0}$ *// initialize gradient*
6: **for** $R_j : R_j \neq R_i$ **do**
7: $\nabla f_\theta \leftarrow \nabla f_\theta + \nabla \left[\lambda(hinge(f_\theta(R_i) - f_\theta(R_j)))\right]$ *// (modified) hinge-loss gradient*
8: $\nabla f_\theta \leftarrow \frac{\nabla f_\theta}{p_i}$
9: $\theta \leftarrow \theta + \mu \nabla f_{\theta_t}$ *// update the ranking model*

2.3 Propensity estimation methods

Recent work in CLTR has focused on estimating propensities from data [2, 35, 36]. As the aim of our work is to compare counterfactual and online LTR approaches, we consider propensity estimation beyond the scope of this paper and assume the propensity scores are known a priori. This is a reasonable assumption, as practitioners typically first perform a randomization experiment to measure the observation probabilities before applying a counterfactual learning algorithm [4].

Our experimental setup allows us to measure the difference of counterfactual methods and online methods without confounding our results by the accuracy of the propensity estimator.

3 ONLINE LEARNING TO RANK

Online Learning to Rank (OLTR) [8, 24, 29, 37] aims to learn by directly interacting with users. OLTR algorithms affect the data gathered during the learning process because they have control over what is displayed to users. These interventions potentially allow for more efficient learning by requiring less user interactions to reach a certain level of performance. Yet, an OLTR algorithm has to simultaneously provide good results to the user and learn from their interactions with the displayed results [22]. Thus besides unbiasedly learning from user interactions, the user experience during learning should also be maintained. A great advantage of the online approach is that learned behavior is immediately applied. However, this high level of responsiveness also means that an unreliable OLTR method can decrease the user experience immediately, making it a potential risk. Therefore, it is important for OLTR methods to be reliable, i.e. unbiased and robust to noise.

In contrast to CLTR, OLTR methods do not explicitly model user behavior, i.e., they do not estimate observance probabilities. Instead, they use stochasticity in the displayed results to handle selection and position biases. In addition, their properties are only based on simple assumptions about user behavior [24, 37], e.g., *a relevant document is more likely to be clicked than a non-relevant document.* Thus, in cases where users are hard to model OLTR

may have an advantage over CLTR. In other areas of machine learning [30], online (or active) approaches tend to be more efficient than algorithms without control over data gathering w.r.t. data requirements. However, CLTR and OLTR methods have never been compared directly, thus currently we do not know if this advantage also generalizes to LTR problems.

3.1 Dueling bandit gradient descent

Dueling Bandit Gradient Descent (DBGD) [37] is the earliest OLTR method and is based on interleaving: an unbiased online evaluation method. Interleaving methods unbiasedly compare two ranking systems in the online setting [9, 12, 27]. Therefore, interleaving can be used to recognize an improvement to a ranking system. At each iteration DBGD compares its current model with a sampled variation using interleaving. If a preference towards the variation is inferred, the current model is updated in its direction. Over time this process estimates a gradient descent and the model should oscillate towards an optimum w.r.t. user preference. Most OLTR methods published to date are extensions of DBGD. This includes, for instance, Multileave Gradient Descent [28], which compares multiple variations per iteration using multileaving [23, 29]; other methods reuse historical interactions to guide exploration [8, 38]. All of these extensions improve on the initial learning speed of DBGD. However, no extension has been shown to improve long-term performance [24, 26, 28]. Moreover, even under ideal circumstances and with a very large number of iterations DBGD is unable to reach levels of performance comparable to LTR from labeled datasets [22, 24]. Despite these shortcomings, DBGD is a key method in the field of OLTR.

3.2 Pairwise differentiable gradient descent

In reaction to the shortcomings of DBGD, recent work has introduced the Pairwise Differentiable Gradient Descent (PDGD) algorithm [24]. In contrast to DBGD, PDGD does not depend on sampling model variations or online evaluation methods. Instead, PDGD constructs a pairwise gradient at each interaction, from inferred user preferences between documents. Algorithm 3 describes the PDGD method in formal detail. At each iteration, the algorithm waits until a query is issued by the user (line 3). Then PDGD creates a probability distribution over documents by applying a Plackett-Luce model with parameter $\tau \in \mathbb{R}_{>0}$ to the scoring function:

$$P(d|D,\theta) = \frac{e^{\tau f_\theta(d)}}{\sum_{d' \in D} e^{\tau f_\theta(d')}}. \tag{3}$$

We introduce the τ parameter to control the *sharpness* of the initial distribution, which indicates the confidence we have in the initial model. Previous work only considered cold-start situations thus did not require this parameter [24, 25]. From this distribution a result list is sampled (Line 5) and displayed to the user. Then PDGD infers preferences between the clicked documents and the first unclicked document and every unclicked document preceding a clicked document, a longstanding pairwise assumption [11]. With $d_i \succ_\mathbf{c} d_j$ denoting an inferred preference of d_i over d_j, PDGD estimates the model gradient as:

$$\nabla f_\theta \approx \sum_{d_i \succ_\mathbf{c} d_j} \rho(d_i, d_j, R, \theta) \nabla P(d_i \succ d_j|\theta), \tag{4}$$

where ρ is a weighing function used to deal with biases (line 8). Finally, the scoring function is updated according to the estimated gradient (line 10), and the process repeats with the updated model.

The ρ function makes use of the so-called reverse pair ranking function $R^*(d_i, d_j, R)$, which returns the same ranking as R with the position of document d_i and d_j swapped. Then, the value of ρ is determined by the ratio between the probabilities of the two rankings:

$$\rho(d_i, d_j, R, \theta) = \frac{P(R^*(d_i, d_j, R)|\theta)}{P(R|\theta) + P(R^*(d_i, d_j, R)|\theta)}. \tag{5}$$

PDGD assumes that if a user considers both d_i and d_j equally relevant, then inferring the preference in $d_i \succ_\mathbf{c} d_j$ in R is equally probable as inferring the reverse preference $d_j \succ_\mathbf{c} d_i$ in $R^*(d_i, d_j, R)$. Furthermore, if a user prefers one of the documents, inferring the corresponding preference is more likely than the reverse. These assumptions can formulated as:

$$\begin{aligned} &sign(relevance(d_i) - relevance(d_j)) = \\ &\quad sign\left(P(d_i \succ_\mathbf{c} d_j|R) - P(d_j \succ_\mathbf{c} d_i|R^*(d_i, d_j, R))\right). \end{aligned} \tag{6}$$

Intuitively, this means that relative relevance differences can be inferred by swapping document pairs without changing the rest of the ranking. A similar approach is used by counterfactual methods to estimate propensities [16], conversely, PDGD uses it to directly optimize its ranking model. In the original paper Oosterhuis and de Rijke prove that the gradient estimation of PDGD is unbiased w.r.t. document pair preferences. This means that the expected gradient of PDGD can be written as a sum over all document pairs:

$$E[\Delta f(\cdot, \theta)] = \sum_{(d_i, d_j) \in D} \alpha_{ij} \left(f'(d_i, \theta) - f'(d_j, \theta)\right), \tag{7}$$

where α_{ij} is a unique weight for every document pair in the collection. PDGD is unbiased in the sense that the sign of α_{ij} matches the user preferences between d_i and d_j:

$$sign(\alpha_{ij}) = sign\left(relevance(d_i) - relevance(d_j)\right). \tag{8}$$

Thus, in expectation PDGD will perform an unbiased update towards the pairwise preferences of the user.

Recent work has extensively compared DBGD with PDGD [24, 25]; PDGD performs considerably better in terms of final convergence, user experience during optimization, and learning speed. These findings generalize from settings with no to moderate levels of position bias and noise [24] to circumstances with extreme levels of bias and noise [25]. PDGD is the new state-of-the-art for OLTR, and we will therefore not consider DBGD in our comparison.

4 EXPECTATIONS FROM PREVIOUS WORK

This section will discuss several expectations about the qualitative differences between CLTR and OLTR based on previous work. Subsequently Section 5 describes the experiments that have been run to test these expectations and Section 6 their outcomes. By discussing existing expectations here we can later contrast them with our observations. Whether and how our results match our expectations can reveal how well our understanding of LTR from user interactions is.

Expectation 1 – The performance at convergence. As described in Section 2 it has been proven that CLTR can unbiasedly optimize

Algorithm 3 Pairwise Differentiable Gradient Descent (PDGD).

Input: initial weights: θ_1; scoring function: f; learning rate μ.
for $t \leftarrow 1 \ldots \infty$ **do**
  $q^{(t)} \leftarrow$ *receive_query*(t) // *obtain a query from a user*
  $D^{(t)} \leftarrow$ *preselect_documents*$(q^{(t)})$ // *preselect doc. for query*
  $\mathbf{R}^{(t)} \leftarrow$ *sample_list*$(f_\theta, D^{(t)})$ // *sample list according to Eq. 3*
  $\mathbf{c}^{(t)} \leftarrow$ *receive_clicks*$(\mathbf{R}^{(t)})$ // *show result list to the user*
  $\nabla f_\theta \leftarrow \mathbf{0}$ // *initialize gradient*
  for $d_i >_\mathbf{c} d_j \in \mathbf{c}^{(t)}$ **do**
    $\nabla f_\theta \leftarrow \nabla f_\theta + \rho(d_i, d_j, R)\nabla P(d_i \succ d_j|\theta)$
  $\theta \leftarrow \theta + \mu \nabla f_\theta$ // *update the ranking model*

additive metrics [1], for instance using CF-DCG, when the observation probabilities of the user are correctly known. Conversely, for PDGD there is no known proof that it optimizes any metric unbiasedly. Therefore, we expect CLTR methods like CF-DCG to reach a higher level of performance than PDGD if the propensities are known, since CLTR can guarantee that the performance metric is optimized, while for PDGD it is unclear whether its pairwise gradient will optimize the metric precisely.

Expectation 2 – The user experience during learning. The field of OLTR has long claimed that their methods provide the most responsive experience [8, 24, 28] because OLTR methods apply their learned model instantly. However, noise may cause a method to decrease model quality (temporarily) and exploration adds stochasticity to the results, thus risking a worsened user experience. As a result, we expect an OLTR method to provide an experience worse than the initial ranker at the start, but as learning continues the user experience should eventually exceed that of the initial model. In contrast, CLTR methods do not affect the user experience during learning as they work with historical data, and therefore, also cannot improve it. Nevertheless, this approach completely avoids the risks of degrading the user experience. Therefore, we expect OLTR to provide a worse experience than under click gathering for CLTR initially, yet eventually the experience under OLTR should exceed that of CLTR. The question is whether the long-term improvements of OLTR outweigh the initial decrease.

Expectation 3 – The effect of interventions. Interventions are expected to greatly reduce the data requirements for learning [30], as they allow algorithms to gather data that is more useful for their current state. Correspondingly, OLTR methods are expected to learn faster [8, 28], in other words, they should require less user interactions to reach a decent level of performance than CLTR methods [24]. Similarly, allowing CLTR methods to intervene, e.g., by deploying a current model should make them more efficient as well.

This concludes the key expectations regarding the performance differences between CLTR and OLTR methods. While these expectations are based on previously published literature on CLTR and OLTR [1, 24, 30], they have never directly been tested. To the best of our knowledge, our study is the first to confirm or challenge them with hard experimental facts.

Table 2: Click probabilities after observing a document in the result list for different user models.

	$P(\text{click} = 1 \mid \text{observed} = 1, \text{rel}(d))$				
rel(d)	0	1	2	3	4
Perfect	0.00	0.20	0.40	0.80	1.00
Binarized	0.10	0.10	0.10	1.00	1.00
Near-Random	0.40	0.45	0.50	0.55	0.60

5 EXPERIMENTS

Our experiments evaluate the user experience of several methods at different time-steps and a multitude of conditions with varying levels of interaction noise, position bias, and selection bias. Due to the scope of this comparison and the varying requirements, we rely on a synthetic setting based on an existing LTR dataset and simulated user behavior. Our setup is an extension of the synthetic experiments common in both OLTR [8, 24, 28] and CLTR [2, 16].

5.1 Optimization setup

We use the *Yahoo! Webscope* dataset [5]; it contains a set of queries with a unique set of preselected documents for each query. The dataset provides a train, validation and test split. We use the train partition during optimization of the methods, the validation set for tuning hyperparameters and the test partition to report our results. Each query-document pair is represented by a feature vector and a relevance label, the relevance labels are in a five-degree scale ranging from *not relevant* (0) to *perfectly relevant* (4).

A baseline ranker is trained to serve as a logging ranker for the CLTR methods, and an initial ranker to warm-start the OLTR method. To create the baseline ranker, we follow the setup of Joachims et al. [16] and train an SVMRank ranker on 1% of the queries in the training data. This setup is chosen as it reflects a common real-world scenario: it is possible to manually annotate a small amount of data to learn an initial ranker, and then use a large amount of logged interaction data, either online or counterfactually, to further improve this ranker.

Finally, the gathering of click-data is simulated using the following steps: First, a user-issued query is simulated by uniformly sampling a query from the training partition of the dataset. Then, the corresponding documents are ranked according to the applied LTR method, i.e., by the logging policy for CLTR methods or by the algorithm itself in OLTR. Subsequently, the ranked results are displayed to a simulated user who then clicks on any number of documents (including none); Section 5.2 details the behavior models we applied. Lastly, the resulting clicks are presented to the LTR method, which may now use the interaction for optimization.

5.2 Simulating user behavior

We simulate user behavior by modelling three aspects of user behavior in search: interaction noise, position bias and selection bias.

First, *interaction noise* affects the probability of a user clicking on a document after observing it. The probability of clicking is conditioned on the relevance label of a document, as users are more likely to click on more relevant documents. Table 2 provides the click probabilities for three different click behavior models: *Perfect* click behavior has probabilities proportional to the relevance and

never clicks on a non-relevant document, simulating an *ideal* user. *Binarized* click behavior acts on only two levels of relevance and is affected by position-bias; this simulated behavior has been used in previous work on CLTR [1, 2, 16]. And *Near-Random* behavior clicks very often, and only slightly more frequently on more relevant documents than on less relevant documents; this behavior simulates very high levels of click noise.

Second, *position bias* is modelled by observation probabilities; for a document at rank i the probability of being observed is determined by the parameter η and formula:

$$P(observed = 1 \mid i) = \left(\frac{1}{i}\right)^{\eta}. \tag{9}$$

Again this follows previous work on CLTR [1, 2, 16]. We apply this position bias to the *Binarized* and *Near-Random* user models; the *Perfect* user observes all documents every time and thus has no position bias. In our experiments we use $\eta = 1$ and $\eta = 2$ to model different levels of position bias.

Thirdly, we simulate *selection bias*, which occurs when not all documents can be displayed and thus also not observed. In practice it also occurs because users never look past certain positions, for instance, users rarely look beyond the initial page of many multi-page result displays. We model selection bias by giving a zero observance probability to documents beyond rank 10. This is common practice in OLTR experiments [8, 24, 28]; in contrast, CLTR methods assume that no selection bias is present. To investigate the effect of selection bias, our experiments both contain simulations with and without it.

In conclusion, we can apply selection bias, have two levels of position bias, and three levels of interaction noise. In total, we apply ten different types of user behavior: *Perfect* click behavior with and without selection bias, the *Binarized* and *Near-Random* click behaviors with two levels of position bias, with and without selection bias. To the best of our knowledge this is the most extensive set of types of behavior used for evaluating CLTR and OLTR methods, in addition to being the first comparison between the two methodologies.

5.3 Evaluation

To measure the performance of a ranker at any time step, we evaluate it on held-out annotated test data using the *nDCG*@10 metric [10]. We use the learned models without any exploration when evaluating performance at convergence. To evaluate the user experience during learning for OLTR we apply the algorithm with exploration to the held-out dataset, this measures the performance for a previously unseen query that appears during optimization.

To test for statistical significance we average the results over multiple runs and apply a two-tailed t-test. In Section 6, we indicate for each comparison, whether the observed difference is statistically significant with $p < 0.01$.

5.4 Methods and interventions

Our comparisons concern one OLTR method and several CLTR methods, in addition to CLTR methods with periodic deployments during the optimization process.

The OLTR method in the comparison is PDGD (Section 3); the parameters ($\mu = 0.01$, $\tau = 10$) were tuned on the validation set.

The CLTR methods we apply are CF-RANK (Section 2.1) and CF-DCG (Section 2.2); the former is the original CLTR method while the latter optimizes DCG corresponding to our evaluation metric. For each run, the CLTR methods are given the propensity scores of the actual user models applied; this guarantees that the CLTR methods optimize unbiasedly. Furthermore, to investigate the effect of interventions we also run these methods with *periodic deployment*. For these runs we replaced the logging policy with the (then) current ranking model after every 200,000 iterations, thus simulating a situation where the learned model is deployed periodically. The parameters for the CLTR were optimized for every instance of user behavior and number of interactions on the validation set, thus results at different time-steps may use different parameter settings. The complete set of hyper parameter settings that were used is released alongside our source code; see Section 8 for details.

6 RESULTS AND ANALYSIS

This section will answer the research questions posed in Section 1, using our experimental results displayed in Figure 1, 2, 3, and 4.

6.1 Ranking performance

We investigate the ranking performance of the OLTR and CLTR methods under different experimental conditions to answer **RQ1**:

> *Do state-of-the-art online and counterfactual LTR methods converge at the same level of performance?*

As discussed in Section 4 we expect CF-DCG to have better performance when the assumptions for unbiased CLTR are met, as unlike PDGD, CF-DCG is then proven to optimize the DCG metric. Figure 1 displays the performance of the counterfactual and online LTR methods over 1,000,000 sessions, where a session consists of a query, a ranking of documents and all corresponding clicks generated by the click simulation. These results are grouped by the user behavior under which they were optimized, which varies in the amount of selection bias, position bias, and interaction noise.

First, we consider the results without selection bias displayed in the top row of Figure 1. Under *Perfect* and *Binarized* click behavior the performance of the CLTR methods and PDGD are quite comparable, with CF-DCG performing better than PDGD in the *Binarized* case ($p < 0.01$). In contrast, performance of the CLTR methods drops below the production ranker under *Near-Random* click behavior, and does not exceed it within 1,000,000 iterations ($p < 0.01$). This goes against our expectations, as the *Near-Random* case does not violate any assumptions necessary for unbiased CLTR as described in [16]. PDGD, on the other hand, is much less affected and reaches much higher levels of performance. Because the *Binarized* and *Near-Random* behaviors have the same position bias, the difference in performance must be due to the increased interaction noise in the latter. Thus, it appears that the CLTR methods are much less robust to noise than PDGD, yet with low levels of noise CLTR methods outperform the OLTR method.

Second, we look at the results with selection bias displayed in the bottom row in Figure 1. For each user model the performance of the CLTR methods is worse than without selection bias. Except under *Near-Random* click behavior where CF-DCG now performs slightly better than the production ranker. Unbiased CLTR does not

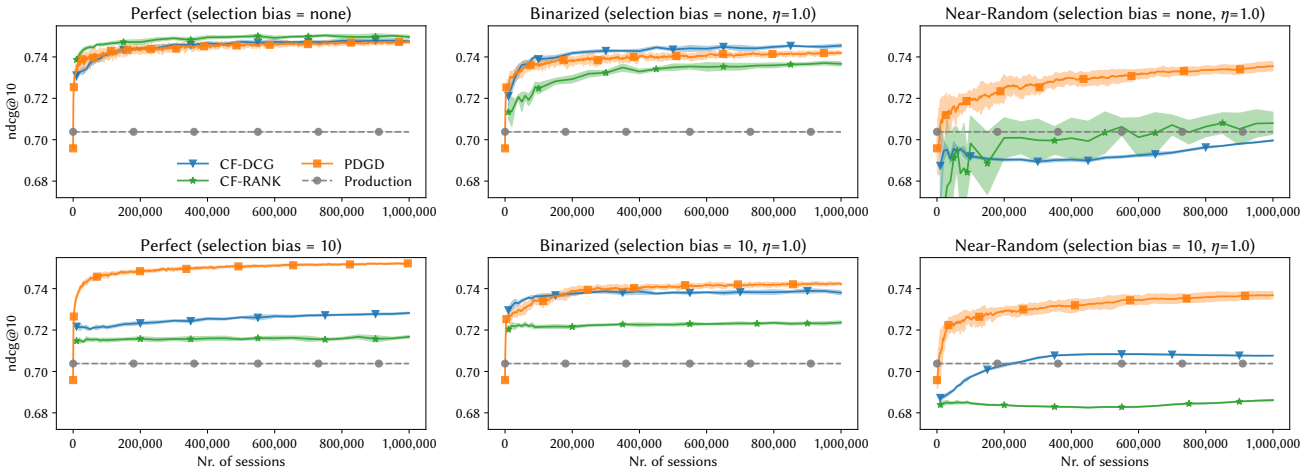

Figure 1: Performance of online and counterfactual methods under perfect, binarized, and near-random user models. In the top row no selection bias is present; in the bottom row, a selection of 10 is used.

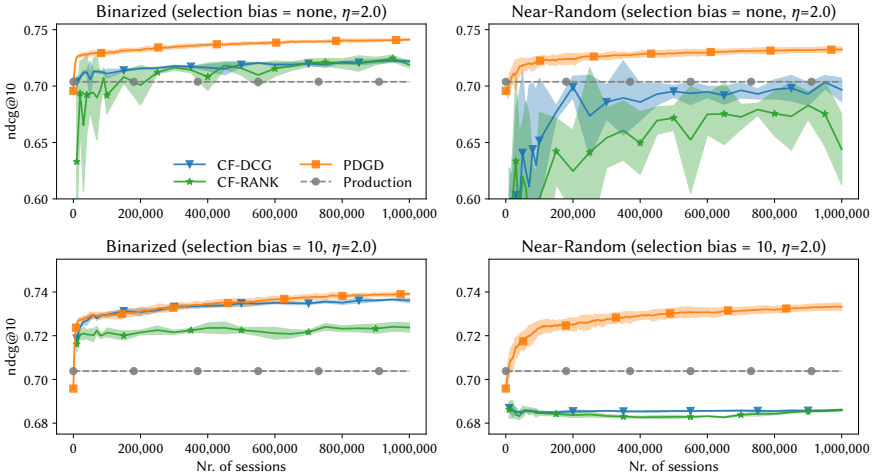

Figure 2: Performance of online and counterfactual methods under very strong position bias ($\eta = 2$). The scale of the y-axis of the plots in the top row has been modified to be able to show the large variance. In the top row no selection bias is present; in the bottom row, a selection of 10 is used.

consider selection bias [16] which could explain this unexpected result. In contrast, the performance of PDGD is affected very little in comparison and is now better than both CLTR methods under all click behaviors ($p < 0.01$). Thus, it appears that PDGD is preferable when selection bias is present.

Third, to understand the effect of position bias we look at the results in Figure 2, where strong position bias is simulated with $\eta = 2$. It is clear that all methods are negatively affected by strong position bias. Unexpectedly, PDGD now outperforms the CLTR methods in all cases ($p < 0.01$), even though the *Binarized* click behavior without selection bias provides the exact circumstances for which CLTR was designed [16]. Therefore, we attribute the negative effect on CLTR to high variance since the methods are still proven to be unbiased in this case. This may further explain why selection bias has a positive effect on the CLTR methods under the *Binarized* click behavior: it removes documents with low propensities that lead to high variance. Clearly, we see that OLTR is better at handling high levels of position bias than CLTR.

In conclusion, we answer **RQ1** negatively: online and counterfactual methods do not converge to the same level of performance. However, which method reaches the best performance depends heavily on the conditions under which they are deployed. In the presence of selection bias or under high levels of position bias or interaction noise OLTR reaches the highest performance. However, when there is no selection bias, little position bias and little interaction noise, then CLTR reaches a level of performance that OLTR is unable to obtain. Counter to our expectations, even when the assumptions of CLTR are true, the CLTR methods are still not robust to noise. Thus, to be able to make the best decision between the CLTR and OLTR methodologies, a practitioner should first measure the severity of different types of bias and noise in their search scenario.

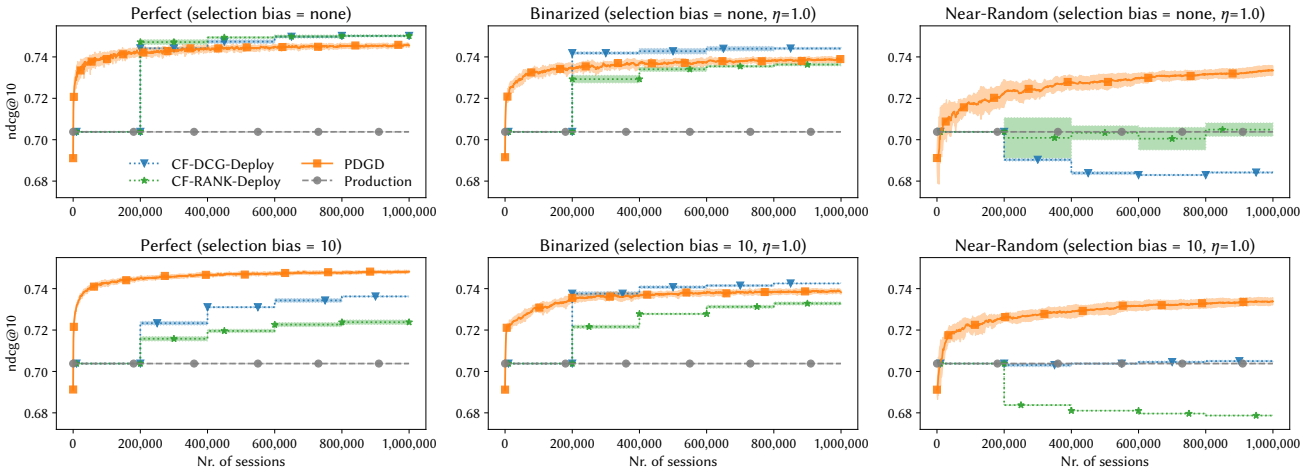

Figure 3: Display performance during training, indicating user experience. In the top row no selection bias is present; in the bottom row, a selection of 10 is used.

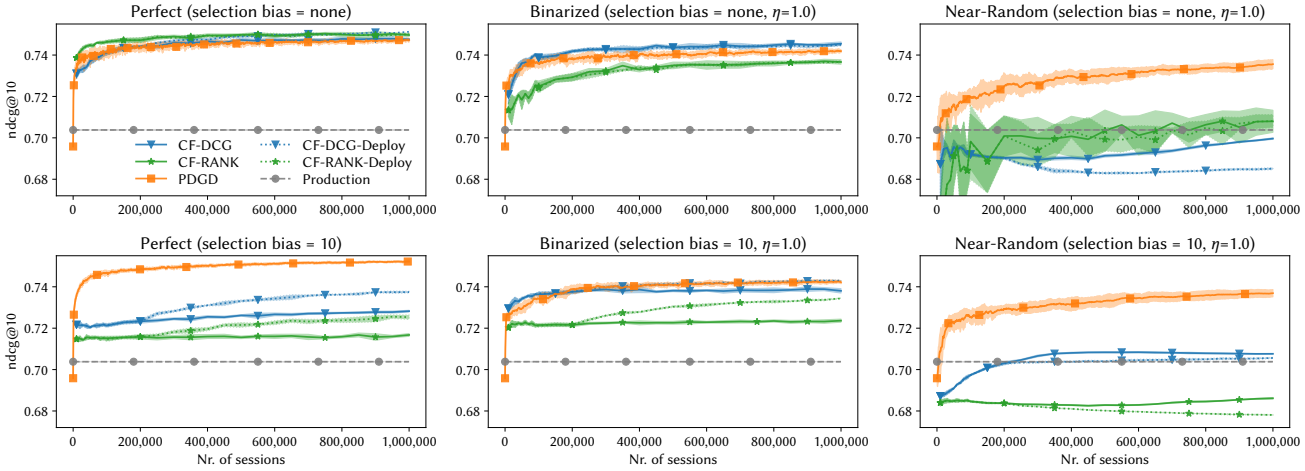

Figure 4: Performance of counterfactual methods with a deployment every 200,000 sessions. In the top row no selection bias is present; in the bottom row, a selection of 10 is used.

6.2 User experience

In this section, we examine the quality of displayed rankings in order to answer **RQ2**:

> *Is the user experience the same for online methods as for counterfactual methods?*

Figure 3 shows the quality of rankings displayed by the PDGD method during optimization and of the *Production* ranker used to gather click-logs for the CLTR methods. For clarity: we are not discussing the *CF-DCG-Deploy* and *CF-RANK-Deploy* results for this research question, they will be discussed in Section 6.3.

In Section 4 we stated the expectation that OLTR methods start with a user experience worse than the production ranker due to exploration. However, OLTR is expected to overtake the production ranker as it continues to learn from interactions. The results in Figure 3 confirm this expectation. Across all types of user behavior, we see that the initially displayed performance is substantially worse than the production ranker ($p < 0.01$). PDGD provides considerably better rankings than the production ranker within 1,000, 2,000 and 21,000 sessions for *Perfect*, *Binarized* and *Near-Random* click behavior, respectively ($p < 0.01$). Thus, we conclude that PDGD provides a better user experience than CLTR methods overall, with a decrease in quality for a limited initial period.

Therefore, we answer **RQ2** negatively: OLTR does not provide the same user experience as CLTR. Besides a limited initial period, OLTR provides a more responsive user experience during optimization than CLTR. However, it is up to practitioners to decide whether the initial worse period is worth it, or whether they prefer the constant user experience in the click gathering for CLTR.

6.3 The power of interventions

In this section we investigate whether performing interventions helps the learning performance, so as to answer **RQ3**:

> *When do online interventions help the learning algorithm?*

To answer this question we consider the performance of the optimized models in Figure 4, and the user experience during click gathering in Figure 3.

In Section 4 we stated the expectation that interventions significantly speed up the learning process. In Figure 4 the performance of CLTR methods diverge at the first moment of deployment: after 200,000 sessions. We see that only in cases with high interaction noise, i.e., *Near-Random* click behavior, periodic deployment leads to worse performance than without ($p < 0.01$). For *Perfect* and *Binarized* click behavior, periodic deployment has no negative effects, moreover, when selection bias is present it substantially increases performance ($p < 0.01$). Thus it appears that interventions cause CLTR methods to reach higher levels of performance and especially help in dealing with selection bias.

Then we examine Figure 3, which displays the user experience during click gathering. Here we see that interventions allow users to benefit from improvements earlier, or suffer from deteriorations sooner. The same trend appears: a worse experience under high interaction noise, a better experience with little noise. Furthermore, CF-DCG with periodic deployment is capable of providing a better user experience than PDGD when little noise is present ($p < 0.01$). Unlike PDGD, CF-DCG does not perform exploration which seems to be a crucial advantage in these cases.

Lastly, we discuss the expectation that interventions speed up learning, in particular that OLTR methods require significantly less data. None of our results indicate that OLTR learns faster than CLTR methods. While in many cases OLTR reaches higher performance levels than CLTR, when they reach comparable levels they do so after similar numbers of interactions. We suspect the reason to be that PDGD does not reiterate over previous interactions, where the CLTR methods perform numerous epochs. Nonetheless, despite expectations in previous work our results do not indicate that the interventions of OLTR reduce data requirements.

To answer **RQ3**: interventions help CLTR methods in circumstances where they already improve over the production ranker. Moreover, their effect is substantial when dealing with selection bias. Unfortunately, deployment in difficult circumstances, i.e. with high levels of noise, can decrease performance even further and negatively affect the user experience considerably. Thus, practitioners should realize that a repeated cycle of optimization and deployment with CLTR can be quite harmful to the user experience. Counterfactual evaluation [18, 32, 34] could estimate whether the deployment of a model improves the experience, before deployment. The question is whether this evaluation is reliable and sensitive enough to prevent harmful changes.

7 RELATED WORK

In this section we discuss previous work that concerns large-scale comparisons of LTR methods.

Liu [19] provides a comprehensive overview of the (then) state-of-the-art in (mostly) LTR from labeled data but does not include a large-scale empirical comparison of methods. Tax et al. [33] do compare LTR algorithms that use manually annotated training data in a large-scale cross-benchmark comparison. They show that there is no single optimal LTR algorithm and provide a selection of supervised LTR methods that are pareto-optimal. In this paper we compare two different families of LTR algorithms: *online* and *counterfactual* LTR methods, neither of which learn from manually annotated data; both types of method utilize user interactions. As such, the algorithms we compare are not supervised in the traditional sense [16].

A systematic comparison of CLTR methods appears to be lacking at this moment in time. Joachims and Swaminathan [15] seem to have provided the first comprehensive overview of counterfactual methods for LTR aimed at the information retrieval community, but the authors do not include a large-scale experimental comparison. More recently, Ai et al. [3] provide an overview of existing approaches to CLTR and describe both the theory and detailed instructions on how to deploy CLTR in practice. Furthermore, their work also contrasts CLTR with click models [6] but it does not contrast CLTR and OLTR methods.

Similarly, a systematic comparison of OLTR methods appears to be lacking too. The comprehensive survey due to [7] is several years old; it does not provide a large-scale experimental comparison nor does it contrast CLTR and OLTR methods; modern OLTR algorithms such as PDGD are also not included. In a more recent tutorial on OLTR, Oosterhuis [21] does provide a theoretical and experimental comparison of OLTR methods based on Dueling Bandit Gradient Descent and PDGD.

Our aim in this study is to gain an understanding in what situations counterfactual and online LTR approaches are appropriate to be used. To the best of our knowledge, there is no prior work that systematically compares counterfactual and online LTR approaches, or answers this question.

8 CONCLUSION

The goal of this study was to answer the question:

> *How should LTR practitioners choose which method to apply from either* counterfactual *or* online *LTR methodologies?*

The choice between OLTR and CLTR is important as there are large differences between the results obtained by the two methodologies. We recognize three factors that determine which approach should be preferred: selection bias, position bias, and interaction noise. CLTR reaches a higher level of performance than OLTR in the absence of selection bias, and when there is little position bias or interaction noise. In contrast, OLTR outperforms CLTR in the presence of selection bias, high position bias, high interaction noise, or any combination of these three. Surprisingly, CLTR methods can decrease performance w.r.t. the production ranker when high levels of noise are present, even in situations where they are proven to be unbiased. We conclude that OLTR is more robust to different types of bias and noise than CLTR. Therefore, practitioners should be well aware of the levels of bias and noise present in their search setting to choose between the two methodologies.

Unlike OLTR, CLTR does not need to intervene and can use data collected by the production ranker, which prevents a negative impact on the user experience. OLTR initially provides an experience worse than the production ranker but very quickly substantially improves over it. We have not observed OLTR having large negative effects on the user experience, even under high levels of interaction noise. However, practitioners should consider whether they are willing to risk the initial worsened user experience in order to get long term gains.

We observed that cycles of optimization and deployment with CLTR methods can have harmful effects on performance and user

experience. High levels of interaction noise can severely worsen model quality for CLTR; if, subsequently, such a model is deployed, it can worsen the next model even further. Thus, practitioners should realize that CLTR brings risks to the user experience and evaluate models before deploying them, for instance using offline or counterfactual evaluation [18, 32, 34] .

Our comparison is not without limitations: In our experiments, the CLTR methods were provided with the exact propensities; in realistic settings these values are not known and have to be estimated [2]. Thus we do not consider how errors in propensity estimation affect the comparison. Additionally, we have not applied any heuristic methods such as propensity clipping; these methods reduce variance but make CLTR biased. Such heuristics could help in situations with high position bias, though they would not help combat interaction noise. Finally, our comparative study considers only a single metric on a single dataset. Although the dataset and metric we use are widely accepted as a benchmark in both OLTR and CLTR, we would like to extend our study to multiple datasets, measuring across various dimensions and utilizing real user behavior from deployed systems in future work.

Our findings also reveal the importance of safety in LTR. Future work could further look at methods that evaluate when it is safe to deploy models. Moreover, a method that could estimate which CLTR or OLTR algorithm would perform best, could provide the best user experience with little effort from practitioners.

Code and data

To facilitate reproducibility of the results in this paper, we are sharing all resources used in this paper at http://github.com/rjagerman/sigir2019-user-interactions.

ACKNOWLEDGMENTS

This research was partially supported by Ahold Delhaize, the Association of Universities in the Netherlands (VSNU), the Innovation Center for Artificial Intelligence (ICAI), and the Netherlands Organisation for Scientific Research (NWO) under project nr 612.001.551. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Aman Agarwal, Ivan Zaitsev, and Thorsten Joachims. 2018. Counterfactual Learning-to-Rank for Additive Metrics and Deep Models. *arXiv preprint arXiv:1805.00065* (2018).
[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] Qingyao Ai, Jiaxin Mao, Yiqun Liu, and W Bruce Croft. 2018. Unbiased Learning to Rank: Theory and Practice. In *CIKM*. ACM, 2305–2306.
[4] Michael Bendersky, Xuanhui Wang, Donald Metzler, and Marc Najork. 2017. Learning from User Interactions in Personal Search via Attribute Parameterization. In *WSDM*. ACM, 791–799.
[5] Olivier Chapelle and Yi Chang. 2011. Yahoo! Learning to Rank Challenge Overview. In *Proceedings of the Learning to Rank Challenge*. 1–24.
[6] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2015. *Click Models for Web Search*. Morgan & Claypool Publishers.
[7] Artem Grotov and Maarten de Rijke. 2016. Online Learning to Rank for Information Retrieval: SIGIR 2016 Tutorial. In *SIGIR*. ACM, 1215–1218.
[8] Katja Hofmann, Anne Schuth, Shimon Whiteson, and Maarten de Rijke. 2013. Reusing Historical Interaction Data for Faster Online Learning to Rank for IR. In *WSDM*. ACM, 183–192.
[9] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. 2011. A Probabilistic Method for Inferring Preferences from Clicks. In *CIKM*. ACM, 249–258.
[10] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.
[11] Thorsten Joachims. 2002. Optimizing Search Engines using Clickthrough Data. In *SIGKDD*. ACM, 133–142.
[12] Thorsten Joachims. 2003. Evaluating Retrieval Performance using Clickthrough Data. In *Text Mining*. Physica/Springer.
[13] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately Interpreting Clickthrough Data as Implicit Feedback. In *SIGIR*. ACM, 154–161.
[14] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, Filip Radlinski, and Geri Gay. 2007. Evaluating the Accuracy of Implicit Feedback from Clicks and Query Reformulations in Web Search. *ACM Transactions on Information Systems (TOIS)* 25, 2 (2007), 7.
[15] Thorsten Joachims and Adith Swaminathan. 2016. Counterfactual Evaluation and Learning for Search, Recommendation and Ad Placement. In *SIGIR*. ACM, New York, NY, USA, 1199–1201.
[16] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased Learning-to-Rank with Biased Feedback. In *WSDM*. ACM, 781–789.
[17] Branislav Kveton, Chang Li, Tor Lattimore, Ilya Markov, Maarten de Rijke, Csaba Szepesvari, and Masrour Zoghi. 2018. BubbleRank: Safe Online Learning to Rerank. *arXiv preprint arXiv:1806.05819* (2018).
[18] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. 2011. Unbiased Offline Evaluation of Contextual-Bandit-Based News Article Recommendation Algorithms. In *WSDM*. ACM, 297–306.
[19] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (2009), 225–331.
[20] Tie-Yan Liu, Jun Xu, Tao Qin, Wenying Xiong, and Hang Li. 2007. Letor: Benchmark Dataset for Research on Learning to Rank for Information Retrieval. In *Proceedings of SIGIR 2007 workshop on learning to rank for information retrieval*, Vol. 310. ACM Amsterdam, The Netherlands.
[21] Harrie Oosterhuis. 2018. Learning to Rank and Evaluation in the Online Setting. 12th Russian Summer School in Information Retrieval (RuSSIR 2018). (2018).
[22] Harrie Oosterhuis and Maarten de Rijke. 2017. Balancing Speed and Quality in Online Learning to Rank for Information Retrieval. In *CIKM*. ACM, 277–286.
[23] Harrie Oosterhuis and Maarten de Rijke. 2017. Sensitive and Scalable Online Evaluation with Theoretical Guarantees. In *CIKM*. ACM, 77–86.
[24] Harrie Oosterhuis and Maarten de Rijke. 2018. Differentiable Unbiased Online Learning to Rank. In *CIKM*. ACM, 1293–1302.
[25] Harrie Oosterhuis and Maarten de Rijke. 2019. Optimizing Ranking Models in the Online Setting. In *ECIR*. Springer, 382–396.
[26] Harrie Oosterhuis, Anne Schuth, and Maarten de Rijke. 2016. Probabilistic Multileave Gradient Descent. In *ECIR*. Springer, 661–668.
[27] Filip Radlinski and Nick Craswell. 2013. Optimized Interleaving for Online Retrieval Evaluation. In *WSDM*. ACM, 245–254.
[28] Anne Schuth, Harrie Oosterhuis, Shimon Whiteson, and Maarten de Rijke. 2016. Multileave Gradient Descent for Fast Online Learning to Rank. In *WSDM*. ACM, 457–466.
[29] Anne Schuth, Floor Sietsma, Shimon Whiteson, Damien Lefortier, and Maarten de Rijke. 2014. Multileaved Comparisons for Fast Online Evaluation. In *CIKM*. ACM, 71–80.
[30] Burr Settles. 2012. Active Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6, 1 (2012), 1–114.
[31] Adith Swaminathan and Thorsten Joachims. 2015. Counterfactual Risk Minimization: Learning from Logged Bandit Feedback. In *ICML*. PMLR, 814–823.
[32] Adith Swaminathan, Akshay Krishnamurthy, Alekh Agarwal, Miro Dudik, John Langford, Damien Jose, and Imed Zitouni. 2017. Off-policy Evaluation for Slate Recommendation. In *NIPS*. 3632–3642.
[33] Niek Tax, Sander Bockting, and Djoerd Hiemstra. 2015. A Cross-benchmark Comparison of 87 Learning to Rank Methods. *Information Processing & Management* 51, 6 (2015), 757–772.
[34] Philip S Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. 2015. High-Confidence Off-Policy Evaluation. In *AAAI*. 3000–3006.
[35] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to Rank with Selection Bias in Personal Search. In *SIGIR*. ACM, 115–124.
[36] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position Bias Estimation for Unbiased Learning to Rank in Personal Search. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. ACM, 610–618.
[37] Yisong Yue and Thorsten Joachims. 2009. Interactively Optimizing Information Retrieval Systems as a Dueling Bandits Problem. In *ICML*. ACM, 1201–1208.
[38] Tong Zhao and Irwin King. 2016. Constructing Reliable Gradient Exploration for Online Learning to Rank. In *CIKM*. ACM, 1643–1652.