Towards Two-Stage Counterfactual Learning to Rank

Shashank Gupta
University of Amsterdam
Amsterdam, The Netherlands
s.gupta2@uva.nl

Yiming Liao
Meta AI
New York, United States of America
yimingliao@meta.com

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

Abstract

Counterfactual learning to rank (CLTR) aims to learn a ranking policy from user interactions while correcting for the inherent biases in interaction data, such as position bias. Existing CLTR methods assume a single ranking policy that selects top-K ranking from the entire document candidate set. In real-world applications, the candidate document set is on the order of millions, making a single-stage ranking policy impractical. In order to scale to millions of documents, real-world ranking systems are designed in a two-stage fashion, with a candidate generator followed by a ranker. The existing CLTR method for a two-stage offline ranking system only considers the top-1 ranking set-up and only focuses on training the candidate generator, with the ranker fixed. A CLTR method for training both the ranker and candidate generator jointly is missing from the existing literature.

In this paper, we propose a two-stage CLTR estimator that considers the interaction between the two stages and estimates the joint value of the two policies offline. In addition, we propose a novel joint optimization method to train the candidate and ranker policies, respectively. To the best of our knowledge, we are the first to propose a CLTR estimator and learning method for two-stage ranking. Experimental results on a semi-synthetic benchmark demonstrate the effectiveness of the proposed joint CLTR method over baselines.

CCS Concepts

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

Keywords

Learning to rank; Counterfactual learning to rank

ACM Reference Format:
Shashank Gupta, Yiming Liao, and Maarten de Rijke. 2025. Towards Two-Stage Counterfactual Learning to Rank. In *Proceedings of the 2025 International ACM SIGIR Conference onInnovative Concepts and Theories in Information Retrieval (ICTIR) (ICTIR '25), July 18, 2025, Padua, Italy*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3731120.3744583

1 Introduction

A learning to rank (LTR) method aims to optimize a ranking policy to maximize a given information retrieval (IR) metric [21]. Traditionally, LTR policies were trained using manually curated queries and document relevance judgments. It is well known now that manually curating relevance judgments is time consuming, not scalable, and does not always translate to user preferences [3, 31]. As an alternative, LTR from user interactions is scalable because user interactions are cheaper to collect on scale, and user interactions are generally aligned with user interests [19]. However, user interactions are noisy and biased indicators of true user preferences, subject to biases such as position bias, and trust bias in search engines [2, 18].

Counterfactual learning to rank (CLTR) aims to learn a ranking policy while correcting for biases in user interaction data [8, 9, 18, 28]. Inverse propensity scoring (IPS) is the most common choice of estimator for CLTR [12, 25]. IPS weights each user interaction on a document with the inverse of the document's exposure probability, also known as document exposure propensity. Thus, documents with historically lower exposure propensity receive a higher weight and vice versa. In expectation, this procedure optimizes for unbiased document relevance [2, 16, 18].

Traditionally, CLTR methods assume a single re-ranking policy, i.e., for a given query, the re-ranking policy selects top-K items from the *candidate document set*, generated via a (lightweight) candidate generator. In a typical CLTR pipeline, the candidate generator is fixed, and only the re-ranker is updated with the CLTR method [9]. To the best of our knowledge, there is no CLTR method that jointly updates both candidate generator and the re-ranker.

In the context of offline contextual bandit learning, Ma et al. [22] introduced the first two-stage off-policy correction method. The proposed estimator considered the interaction between the candidate generator and the contextual bandit policy. Although effective, the method has the following limitations: (i) it is only designed for the contextual bandit setup (top-1 ranking) and extending it to the CLTR setup is non-trivial since existing CLTR estimators do no consider the interaction between different policies in a cascading setup [1, 18, 25]; and (ii) it only trains the candidate generator using the two-stage estimator while keeping the second-stage bandit policy fixed, typically pre-trained.

We argue that this strategy is not optimal. Initially trained using logged data generated from the production candidate generator, the re-ranker faces potential performance issues when a new candidate generator is deployed with the new candidate policy. This causes a distribution shift in the input data distribution for the re-ranker, potentially leading to an overall degradation in the re-ranker performance when deployed online. To mitigate this, we propose a joint optimization method to optimize the re-ranker and candidate generator via an alternating optimization fashion. Experimental results on a semi-synthetic benchmark demonstrate that the proposed joint optimization method outperforms the two-staged baseline method, where the re-ranker and candidate generator are trained independently. To the best of our knowledge, this work is the first to develop a CLTR estimator and learning method for the two-stage LTR system. We hope our contributions will enable LTR

ICTIR '25, Padua, Italy

ACM ISBN 979-8-4007-1861-8/2025/07
https://doi.org/10.1145/3731120.3744583

practitioners to apply CLTR methods at a real-world scale.

2 Related Work

2.1 Counterfactual learning to rank

In the context of LTR, Joachims et al. [18] introduced the first counterfactual learning method to correct for position bias in search engines [15]. They applied the IPS weighting technique, common in the offline contextual bandit literature [30]. For each user interaction with a document displayed by the ranking system, a click is weighted with the inverse of the document examination probability, a.k.a. document propensity. With the IPS weighting scheme, the effect of the position bias is removed in expectation. As an extension, Oosterhuis and de Rijke [25] introduced a policy-aware CLTR method for the top-K ranking setup. In a top-K ranking setup, any document ranked beyond the position K gets zero exposure. In this work, we deal with the top-K setting. For a more recent overview of CLTR, we refer to [8].

2.2 Offline policy learning for contextual bandits

In the context of offline contextual bandits, IPS is commonly used to correct for selection bias over actions introduced by the previously deployed behavior policy [10, 11, 17, 32–34, 37, 38]. Briefly, for each (action, context) pair, the corresponding reward is weighted by the ratio of corresponding action probabilities under the new policy and the behavior policy. In expectation, this weighting scheme removes the effect of selection bias introduced by the behavior policy.

2.3 Two-stage offline policy learning

Most work on offline policy learning for contextual bandits and LTR consider a single policy, typically the re-ranking policy, that re-ranks all the candidate documents for a given query context, generated via a candidate generation policy [8, 30]. Typically, the candidate generation policy is fixed during the off-policy update of the re-ranker, and the interaction between the two policies is generally ignored.

In the context of LTR from manual relevance judgments, Dang et al. [6] introduced the first joint two-stage ranking method, which considers the interaction between the re-ranker and candidate policy and jointly updates both. Wang et al. [35] introduced an efficient boosting-based method for joint two-staged ranking.

In offline contextual bandits, Ma et al. [22] proposed a two-stage off-policy learning objective that considers the interaction between the first and second-stage contextual bandit policies. They propose a policy learning method to train the candidate generator considering the response of the re-ranker on the candidate generated at the first stage.

3 Background

3.1 Counterfactual learning to rank

For a given query q ($q \sim P(Q)$), a ranked list $y = \{d_1, d_2, \dots, d_k\}$ generated by a ranker π, we define the utility of the list y as:

$$U(y \mid q, \pi) = \sum_{d \in D} \alpha_{k(d)} P(R = 1 \mid q, d) = \sum_{d \in D} \alpha_{k(d)} R_{q,d}, \quad (1)$$

where $k(d)$ is the rank of the document d in the list y, and $\alpha_{k(d)}$ is the weight for the document d at the rank $k(d)$, and $P(R = 1 \mid q, d) = R_{q,d}$ is the probability of relevance for the query, document pair (q, d). Given the ranking utility, the goal of a LTR algorithm is to train a ranking policy π to optimize the following overall utility function:

$$\begin{aligned} U(\pi) &= \mathbb{E}_{q \sim P(Q)} \mathbb{E}_{y \sim \pi(\cdot \mid q)} \left[U(y \mid q, \pi) \right] \\ &= \mathbb{E}_q \left[\sum_{d \in D} \rho(d \mid q, \pi) R_{q,d} \right], \end{aligned} \quad (2)$$

where the choice of the document weight $\rho(d \mid q, \pi) = \mathbb{E}_{y \sim \pi}\left[\alpha_{k(d)}\right]$ (Eq. 1) defines the IR metric that is being optimized as a result. For example, the choice of $\alpha_{k(d)} = (\log_2(\text{rank}(d \mid y) + 1))^{-1}$ means that we are optimizing for the NDCG metric [21].

Traditional LTR methods assume access to the *true* manually graded document relevance scores: $R_{q,d}$ (Eq. 1). However, manually curating relevance judgments in practice is not scalable, and relevance judgments are generally not aligned with user interests [3, 29]. As an alternative, CLTR considers the user interactions directly. CLTR assumes access to the click log dataset $\mathcal{D}$ from a logging policy (π_0), $\mathcal{D} = \left\{q_i, y_i, c_i\right\}_{i=1}^{N}$, with user issued queries q_i, the ranked list from the ranking policy y_i, and, finally, user clicks ($c_i \in 0, 1$) on the list y_i indicate whether the document was clicked or not. In this work, we assume that user clicks follow the examination hypothesis [4, 18]:

$$P(C = 1 \mid d, q, k) = P(E = 1 \mid k) P(R = 1 \mid d, q), \quad (3)$$

which states that for a given query q, the probability of clicking on a document d depends on the probability of examination at a given rank k of the document and its relevance.

Given the click model (Eq. 3), to define the IPS-based CLTR objective, following Oosterhuis and de Rijke [25], we define the document propensity under a policy π as:

$$\rho(d \mid q, \pi) = \mathbb{E}_{y \sim \pi} \left[P(E = 1 \mid k(d)) \right]. \quad (4)$$

A similar propensity definition was used in other CLTR work [12, 26, 39]. The choice of rank-based examination probability as the propensity sets the overall utility function as the expected number of clicks (Eq. 2). Finally, the CLTR utility function is defined as:

$$\hat{U}(\pi) = \frac{1}{N} \sum_{i=1}^{N} \sum_{d \in D} \frac{\rho(d)}{\rho_0(d)} c_i(d), \quad (5)$$

with $\rho_0(d)$ as the propensity score (Eq. 4) for the logging policy π_0.

3.2 Two-stage off-policy learning

In the context of offline contextual bandits, Ma et al. [22] proposed a two-stage off-policy learning method, where the objective function is defined with respect to the bandit feedback dataset:

$$\mathcal{D} = \left\{s_i, a_i, r_i\right\}_{i=1}^{N}, \quad (6)$$

where s_i is the context/state, $a_i \sim \pi(\cdot \mid s_i)$ is the action sampled from the policy π, and r_i is the reward for the (state, action) pair. The policy π is defined as the mixture of the candidate policy which scores a size k list: A^k, $p(A^k \mid s)$ and the re-ranker $q(a \mid A^k, s)$:

$\pi(a \mid s) = \sum_{A^k} p(A^k \mid s) q(a \mid A^k, s)$. Given this, the two-stage off-policy objective is defined as:

$$\hat{U}(\pi) = \frac{1}{N} \sum_{i=1}^{N} \frac{\pi(a_i \mid s_i)}{\pi_0(a_i \mid s_i)} r_i. \tag{7}$$

In their proposed method, *only* the candidate policy $p(A^k \mid s)$ is updated via a stochastic gradient method; the re-ranker $q(a \mid A^k, s)$ is kept fixed.

4 Method: Two-stage Counterfactual Learning to Rank

This section introduces our main contribution: a novel two-stage CLTR objective and a joint learning method.

4.1 Two-stage LTR

We first define the two-stage LTR objective with relevance labels. Given a candidate generator policy π_c, the candidate list $y_c \sim \pi_c$ of length K_2, the re-ranker policy π_r, and the final ranked list displayed to the user $y_r \sim \pi_r$ of length K, similar to the utility function for a given list in a single-stage setup $U(y \mid q, \pi)$ (Eq. 1), the two-stage LTR objective is given by:

$$\begin{aligned} U(\pi_c, \pi_r) &= \mathbb{E}_{q \sim P(Q)} \mathbb{E}_{y_c \sim \pi_c(\cdot \mid q)} \mathbb{E}_{y_r \sim \pi_r(\cdot \mid y_c, q)} \left[U(y_r \mid q, \pi) \right] \\ &= \mathbb{E}_{q \sim P(Q)} \mathbb{E}_{y_c \sim \pi_c(\cdot \mid q)} \mathbb{E}_{y_r \sim \pi_r(\cdot \mid y_c, q)} \left[\sum_{d \in D} \alpha_{k(d)} R_{q,d} \right] \\ &= \mathbb{E}_{q \sim P(Q)} \left[\sum_{d \in D} \rho_{c,r}(d) R_{q,d} \right], \end{aligned} \tag{8}$$

where $\rho_{c,r}(d) = \mathbb{E}_{y_c \sim \pi_c(\cdot \mid q)} \mathbb{E}_{y_r \sim \pi_r(\cdot \mid y_c, q)} \left[\alpha_{k(d)} \right]$ is the new document weight, equivalent to the single-stage case (Eq. 2).

4.2 Method: A novel two-stage CLTR

Given the true joint utility function for the two-stage ranking with the true relevance (Eq. 8), we now define the CLTR objective with the logged data $\mathcal{D}$ (Eq. 6):

$$\hat{U}(\pi_r, \pi_c) = \frac{1}{N} \sum_{i=1}^{N} \sum_{d \in D} \frac{\rho_{r,c}(d)}{\rho_0(d)} c_i(d). \tag{9}$$

Similarly to the single-stage CLTR objective (Eq. 5), we set the weight $\alpha_{k(d)} = P(E = 1 \mid k(d))$. As a result, the overall objective optimizes for expected clicks on relevant documents [24]. To the best of our knowledge, we are the first to introduce a two-staged CLTR objective for real-world ranking systems.

Theorem 4.1. *The counterfactual objective for the two stage CLTR method (Eq. 9) is unbiased in expectation, i.e.,*

$$\mathbb{E}_{q, y \sim \pi_0, c} \left[\hat{U}(\pi_r, \pi_c) \right] = U(\pi_r, \pi_c). \tag{10}$$

Proof. The expected value of the two-stage objective is given by:

$$\begin{aligned} \mathbb{E}_{q, y \sim \pi_0, c} \left[\hat{U}(\pi_r, \pi_c) \right] &= \mathbb{E}_q \left[\sum_{d \in D} \frac{\rho_{r,c}(d)}{\rho_0(d)} \mathbb{E}_{y \sim \pi_0, c} \left[c(d) \right] \right] \\ &= \mathbb{E}_q \left[\sum_{d \in D} \rho_{r,c}(d) R_{q,d} \right], \end{aligned} \tag{11}$$

where we apply the click model (Eq. 3) in the first step. □

Intuitively, to maximize the objective in case of a click on a given document d, the optimizer will push the expected document weight under both policies $\rho_{r,c}(d)$ to the maximum. To maximize the expected weight, the candidate generator will push the document higher in the candidate list, which gives the re-ranker a chance to finally display the document to the user at a higher rank.

The candidate generator can be selected as a lightweight model for efficiency, generating the candidate list quickly. The re-ranker can be a complex model that incorporates additional features to improve the precision of the results. However, for simplicity, we use the same model for both candidate generator and re-ranker and leave the effects of different architecture choices for future work.

4.3 Joint optimization for two-stage CLTR

For optimization, we use stochastic gradient descent. To estimate the gradient with respect to the policy π, in this work, we make use of the general log-derivate trick of the REINFORCE algorithm [36], with the gradient of the metric weight term following:

$$\nabla_\pi \rho(d) = \mathbb{E}_{y \sim \pi} \left[P(E = 1 \mid k) \nabla_\pi \log \pi(y \mid q) \right]. \tag{12}$$

Ma et al. [22] proposed to optimize the candidate generator policy while keeping the re-ranking policy constant during the optimization (the re-ranker is pre-trained on the same logged data). For the CLTR context, this translates into estimating the gradient:

$$\nabla_{\pi_c} \hat{U}(\pi_r, \pi_c) = \frac{1}{N} \sum_{i=1}^{N} \sum_{d \in D} \frac{\nabla_{\pi_c} \rho_{r,c}(d)}{\rho_0(d)} c_i(d). \tag{13}$$

While this objective considers the feedback from the re-ranker for training the candidate generator, this strategy is overall sub-optimal. The re-ranker policy is pre-trained on the log data generated by the logging policy π_0. This means that it is optimized with respect to candidates generated by the logging policy during training. At inference time, if the candidate generator is switched to a different one, its performance might suffer because of the distribution shift.

For two-stage optimization, if the re-ranker is kept fixed and the candidate is updated during training, then at test time, the distribution of the candidates will change, and the re-ranker might underperform. To remedy this problem, we propose a joint optimization method, where both the candidate and re-ranker are updated during training. We follow an alternating optimization method, wherein for each minibatch, the candidate generator and the re-ranker are updated alternatingly. This ensures that the re-ranker considers the candidate generator's feedback during training.

Gradient calculation. As the choice of the ranking policies in the candidate generator and re-ranker, we choose a Plackett-Luce model, similar to previous work in CLTR [12, 23, 24]. For optimization, we use stochastic gradient descent. For the gradient of the document weight with respect to the candidate policy, we apply

Table 1: NDCG@10 performance of different two-stage CLTR methods with varying candidate list sizes (K_2) and varying amounts of click logs (N). The numbers reported are averages over 25 independent runs.

	$K_2 = 500$			$K_2 = 1,000$			$K_2 = 1,500$		
Method	$N = 0.1M$	$N = 0.32M$	$N = 1M$	$N = 0.1M$	$N = 0.32M$	$N = 1M$	$N = 0.1M$	$N = 0.32M$	$N = 1M$
Baseline	0.404 (0.000)	0.437 (0.001)	0.470 (0.001)	0.401 (0.001)	0.440 (0.001)	0.469 (0.001)	0.400 (0.001)	0.441 (0.001)	0.474 (0.002)
Indepen.	0.410 (0.001)	0.454 (0.001)	0.489 (0.001)	0.416 (0.001)	0.465 (0.015)	0.496 (0.001)	0.416 (0.001)	0.455 (0.001)	0.496 (0.001)
Joint opt.	**0.420 (0.001)**	**0.470 (0.001)**	**0.504 (0.001)**	**0.426 (0.001)**	**0.479 (0.001)**	**0.504 (0.001)**	**0.424 (0.001)**	**0.472 (0.001)**	**0.510 (0.001)**

the log-derivative/REINFORCE trick [12, 36, 39]. The gradient expression is given as:

$$\nabla_{\pi_c} \rho_{r,c}(d) = \mathbb{E}_{y_c \sim \pi_c}\left[\mathbb{E}_{y_r \sim \pi_r(\cdot \mid y_c, q)} \left[P(E = 1 \mid k)\right] \nabla_\pi \log \pi_c(y_c \mid q)\right]. \tag{14}$$

Similarly, the gradient of the document weight with respect to the re-ranker policy can be expressed as:

$$\nabla_{\pi_r} \rho_{r,c}(d) = \mathbb{E}_{y_c \sim \pi_c}\left[\mathbb{E}_{y_r \sim \pi_r(\cdot \mid y_c, q)} \left[P(E = 1 \mid k) \nabla_\pi \log \pi_r(y_r \mid y_c, q)\right]\right]. \tag{15}$$

5 Experimental Setup

For our experiments, we follow the semi-synthetic experimental setup, common in the CLTR literature [12, 13, 18, 25]. The standard LTR datasets involve a query and a pre-selected document list with the corresponding relevance judgments, for example, the MSLR30K dataset [29]. The logging policy is trained on 3% of the queries to simulate a production ranker, which is used to generate the logged dataset following a position-based user model [5, 18, 25]. In the LTR datasets, documents are already prefiltered from the entire document pool and in the order of $\mathbf{O}(100)$, which is not reflective of a real-world setup, where the candidate document set is of much larger order. As a result, the LTR datasets are unsuitable for a realistic large-scale simulation of a two-staged CLTR system. Instead, we rely on the MovieLens-1M recommender systems dataset for the simulation in this work. The dataset consists of user ratings of items. Following previous work [40–42], we assume that the ratings reflect the *true* user intent and are not biased by the recommendations to the user. We assume a rating value of > 3 as a positive reward and a zero otherwise. We randomly sample 10% of users with the *true* rating as the ground truth for policy evaluation. This simulates a real-world setting where a production policy is trained on a small fraction of the *ideal* relevance labels, then deployed to collect click signals [1, 18]. We simulate a top-K ranking setup [12, 25] where any item beyond the top-K slot gets no exposure. The clicks are simulated via a position-based click model [5, 18], where we define the examination probability as follows:

$$P(E = 1 \mid q, d, y) = \begin{cases} \text{rank}(d|y)^{-1} & \text{if rank}(d \mid y) \leq 10, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

For our experiments, we define the query q as the user context, and an item is a document d. Following previous work [12, 24, 26, 27], we use frequency estimates for the propensity $\rho_0(d)$ (Eq. 13). We use a simple matrix factorization model for the re-ranker and candidate generator policy, initialized with user and item embeddings of dimension 50, generated via applying SVD on the initial user-item rating matrix. Exploring the effect of different model architectures is out of scope and will be in future work. For estimating the policy gradient expectations (Eq. 14, 15), we use 300 Monte-Carlo samples [12, 24, 39]. For optimization, we use the Adam optimizer [20] with a learning rate of 0.01 for all methods. Finally, the following methods are included in our comparisons:

(1) *Baseline.* As a baseline, we follow the setup from previous work [22], wherein we pre-train a re-ranker on the click data and only train the candidate generator using the two-stage objective (Eq. 9) and evaluate via the two-stage LTR objective (Eq. 8).
(2) *Independent.* We train the candidate generator and reranker independently and evaluate via the two-stage objective (Eq. 8).
(3) *Joint optimization.* Our proposed method (Sect. 4.3) is where we jointly optimize the candidate generator and re-ranker in an alternating optimization fashion.

6 Results

Table 1 reports the NDCG@10 results for different methods. We report the results with varying the candidate list sizes $K_2 \in \{500, 1000, 1500\}$ for realistic real-world evaluation. The table shows that the joint optimization method (Sect. 4.3) performs the best compared to the other methods.

The baseline method from the two-stage bandit work [22], where the re-ranker is pretrained from the data generated from the logging candidate policy, performs worst among all methods, confirming our initial hypothesis that changing the candidate ranking distribution from training to inference can result in an overall performance drop for the re-ranker policy. The proposed joint optimization method for the candidate generator and the re-ranker performs the best across different candidate list sizes and logged data. This shows that the joint optimization strategy is optimal for the two-stage CLTR methods. The performance of each method improves with increasing logged data size ($N = 0.1M, 0.32M, 1M$), consistent with previous work [12, 14, 18, 25, 27].

Note that in this work, we use the same architecture for the candidate generator and the re-ranker, i.e., a matrix factorization model. Exploring the effects of different neural architectures for the candidate generator and re-ranker is beyond the scope of this work, as our focus is on introducing joint optimization for the two-stage CLTR. We will study it in future work.

7 Conclusion

In this work, we have studied two-stage ranking systems from a CLTR perspective. This is an important problem that has not been adequately discussed in the contemporary literature of CLTR; in this work, we address this gap. First, we propose a novel single LTR two-stage objective function (Eq. 8) for optimizing LTR systems with access to the *ideal* relevance judgments. Next, we propose a two-stage CLTR objective (Eq. 9) that accounts for both the candidate generator and re-ranker jointly. To optimize the overall system, we propose a joint optimization strategy. We also provide the gradients for first-order optimization methods.

To the best of our knowledge, we are the first to tackle the problem of two-stage LTR systems and a counterfactual learning and evaluation method for two-staged systems. As part of future work, we wish to extend the formulation from a contextual bandit setup to a reinforcement learning setup [7].

Acknowledgments

This research was supported by the Dutch Research Council (NWO), under project numbers 024.004.022, NWA.1389.20.183, and KICH3.-LTP.20.006, and the European Union's Horizon Europe program under grant agreement No 101070212. 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, Kenta Takatsu, Ivan Zaitsev, and Thorsten Joachims. 2019. A General Framework for Counterfactual Learning-to-rank. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 5–14.

[2] Aman Agarwal, Xuanhui Wang, Cheng Li, Michael Bendersky, and Marc Najork. 2019. Addressing Trust Bias for Unbiased Learning-to-rank. In *The World Wide Web Conference*. 4–14.

[3] Olivier Chapelle and Yi Chang. 2011. Yahoo! Learning to Rank Challenge Overview. In *Proceedings of the learning to rank challenge*. PMLR, 1–24.

[4] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2015. *Click Models for Web Search*. Morgan & Claypool Publishers.

[5] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An Experimental Comparison of Click Position-bias Models. In *Proceedings of the 2008 international conference on web search and data mining*. 87–94.

[6] Van Dang, Michael Bendersky, and W. Bruce Croft. 2013. Two-stage Learning to Rank for Information Retrieval. In *Advances in Information Retrieval: 35th European Conference on IR Research, ECIR 2013, Moscow, Russia, March 24-27, 2013. Proceedings 35*. Springer, 423–434.

[7] Chongming Gao, Kexin Huang, Jiawei Chen, Yuan Zhang, Biao Li, Peng Jiang, Shiqi Wang, Zhong Zhang, and Xiangnan He. 2023. Alleviating Matthew Effect of Offline Reinforcement Learning in Interactive Recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 238–248.

[8] Shashank Gupta, Philipp Hager, Jin Huang, Ali Vardasbi, and Harrie Oosterhuis. 2023. Recent Advances in the Foundations and Applications of Unbiased Learning to Rank. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[9] Shashank Gupta, Philipp Hager, Jin Huang, Ali Vardasbi, and Harrie Oosterhuis. 2024. Unbiased Learning to Rank: On Recent Advances and Practical Applications. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 1118–1121.

[10] Shashank Gupta, Olivier Jeunen, Harrie Oosterhuis, and Maarten de Rijke. 2024. Optimal Baseline Corrections for Off-Policy Contextual Bandits. In *Proceedings of the 18th ACM Conference on Recommender Systems*. 722–732.

[11] Shashank Gupta, Harrie Oosterhuis, and Maarten de Rijke. 2023. A Deep Generative Recommendation Method for Unbiased Learning from Implicit Feedback. In *Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval*. 87–93.

[12] Shashank Gupta, Harrie Oosterhuis, and Maarten de Rijke. 2023. Safe Deployment for Counterfactual Learning to Rank with Exposure-Based Risk Minimization. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[13] Shashank Gupta, Harrie Oosterhuis, and Maarten de Rijke. 2024. Practical and Robust Safety Guarantees for Advanced Counterfactual Learning to Rank. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 737–747.

[14] Rolf Jagerman, Ilya Markov, and Maarten de Rijke. 2020. Safe Exploration for Optimizing Contextual Bandits. *ACM Transactions on Information Systems (TOIS)* 38, 3 (2020), 1–23.

[15] Thorsten Joachims. 2002. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 133–142.

[16] Thorsten Joachims and Adith Swaminathan. 2016. Counterfactual Evaluation and Learning for Search, Recommendation and Ad Placement. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 1199–1201.

[17] Thorsten Joachims, Adith Swaminathan, and Maarten de Rijke. 2018. Deep learning with Logged Bandit Feedback. In *International Conference on Learning Representations*.

[18] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased Learning-to-rank with Biased Feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. 781–789.

[19] Alexandros Karatzoglou, Linas Baltrunas, and Yue Shi. 2013. Learning to Rank for Recommender Systems. In *Proceedings of the 7th ACM Conference on Recommender Systems*. 493–494.

[20] Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* (2014).

[21] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (2009), 225–331.

[22] 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 *Proceedings of The Web Conference 2020*. 463–473.

[23] Harrie Oosterhuis. 2021. Computationally Efficient Optimization of Plackett-Luce Ranking Models for Relevance and Fairness. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1023–1032.

[24] Harrie Oosterhuis. 2023. Doubly Robust Estimation for Correcting Position Bias in Click Feedback for Unbiased Learning to Rank. *ACM Transactions on Information Systems* 41, 3 (2023), 1–33.

[25] Harrie Oosterhuis and Maarten de Rijke. 2020. Policy-aware Unbiased Learning to Rank for Top-k Rankings. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 489–498.

[26] Harrie Oosterhuis and Maarten de Rijke. 2021. Unifying Online and Counterfactual Learning to Rank: A Novel Counterfactual Estimator that Effectively Utilizes Online Interventions. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 463–471.

[27] Harrie Oosterhuis and Maarten de de Rijke. 2021. Robust Generalization and Safe Query-Specialization in Counterfactual Learning to Rank. In *Proceedings of the Web Conference 2021*. 158–170.

[28] Harrie Oosterhuis, Rolf Jagerman, and Maarten de Rijke. 2020. Unbiased Learning to Rank: Counterfactual and Online Approaches. In *Companion Proceedings of the Web Conference 2020*. 299–300.

[29] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2010. LETOR: A Benchmark Collection for Research on Learning to Rank for Information Retrieval. *Information Retrieval* 13, 4 (2010), 346–374.

[30] Yuta Saito and Thorsten Joachims. 2021. Counterfactual Learning and Evaluation for Recommender Systems: Foundations, Implementations, and Recent Advances. In *Fifteenth ACM Conference on Recommender Systems*. 828–830.

[31] Mark Sanderson. 2010. Test Collection based Evaluation of Information Retrieval Systems. *Foundations and Trends in Information Retrieval* 4, 4 (2010), 247–375.

[32] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as Treatments: Debiasing Learning and Evaluation. In *international conference on machine learning*. PMLR, 1670–1679.

[33] Adith Swaminathan and Thorsten Joachims. 2015. Batch Learning from Logged Bandit Feedback through Counterfactual Risk Minimization. *The Journal of Machine Learning Research* 16, 1 (2015), 1731–1755.

[34] Adith Swaminathan, Akshay Krishnamurthy, Alekh Agarwal, Miro Dudik, John Langford, Damien Jose, and Imed Zitouni. 2017. Off-Policy Evaluation for Slate Recommendation. *Advances in Neural Information Processing Systems* 30 (2017).

[35] Lidan Wang, Jimmy Lin, and Donald Metzler. 2011. A Cascade Ranking Model for Efficient Ranked Retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 105–114.

[36] Ronald J Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine learning* 8, 3 (1992), 229–256.

[37] Hang Wu and May Wang. 2018. Variance Regularized Counterfactual Risk Minimization via Variational Divergence Minimization. In *International Conference on Machine Learning*. PMLR, 5353–5362.

[38] Yuan Xie, Boyi Liu, Qiang Liu, Zhaoran Wang, Yuan Zhou, and Jian Peng. 2018. Off-Policy Evaluation and Learning from Logged Bandit Feedback: Error Reduction via Surrogate Policy. In *International Conference on Learning Representations*.

[39] Himank Yadav, Zhengxiao Du, and Thorsten Joachims. 2021. Policy-Gradient Training of Fair and Unbiased Ranking Functions. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1044–1053.

[40] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based

Recommender System: A Survey and New Perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.

[41] Xiaoxue Zhao, Weinan Zhang, and Jun Wang. 2013. Interactive Collaborative Filtering. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 1411–1420.

[42] Lixin Zou, Long Xia, Yulong Gu, Xiangyu Zhao, Weidong Liu, Jimmy Xiangji Huang, and Dawei Yin. 2020. Neural Interactive Collaborative Filtering. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 749–758.