Unbiased LambdaMART: An Unbiased Pairwise Learning-to-Rank Algorithm

Ziniu Hu*
University of California, Los Angeles, USA
bull@cs.ucla.edu

Yang Wang, Qu Peng, Hang Li
ByteDance AI Lab, Beijing, China
{wangyang.127,pengqu,lihang.lh}@bytedance.com

ABSTRACT

Recently a number of algorithms under the theme of 'unbiased learning-to-rank' have been proposed, which can reduce position bias, the major type of bias in click data, and train a high-performance ranker with click data in learning-to-rank. Most of the existing algorithms, based on the inverse propensity weighting (IPW) principle, first estimate the click bias at each position, and then train an unbiased ranker with the estimated biases using a learning-to-rank algorithm. However, there has not been a method for unbiased pairwise learning-to-rank that can simultaneously conduct debiasing of click data and training of a ranker using a pairwise loss function. In this paper, we propose a novel framework to accomplish the goal and apply this framework to the state-of-the-art pairwise learning-to-rank algorithm, LambdaMART. Our algorithm named Unbiased LambdaMART can jointly estimate the biases at click positions and the biases at unclick positions, and learn an unbiased ranker. Experiments on benchmark data show that Unbiased LambdaMART can significantly outperform existing algorithms by large margins. In addition, an online A/B Testing at a commercial search engine shows that Unbiased LambdaMART can effectively conduct debiasing of click data and enhance relevance ranking.

CCS CONCEPTS

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

KEYWORDS

Learning-to-Rank, Unbiased Learning-to-Rank, LambdaMART

ACM Reference Format:
Ziniu Hu and Yang Wang, Qu Peng, Hang Li. 2019. Unbiased LambdaMART: An Unbiased Pairwise Learning-to-Rank Algorithm. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3308558.3313447

*This work was done when the first author was an intern at ByteDance AI Lab.

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05…$15.00
https://doi.org/10.1145/3308558.3313447

1 INTRODUCTION

Learning-to-rank, which refers to machine learning techniques on automatically constructing a model (ranker) from data for ranking in search, has been widely used in current search systems. Existing algorithms can be categorized into pointwise, pairwise, and listwise approaches according to the loss functions they utilize [18, 19, 21]. Among the proposed algorithms, LambdaMART is a state-of-the-art algorithm [4, 26]. The data for training in learning-to-rank is usually labeled by human assessors so far, and the labelling process is often strenuous and costly. This raises the question of whether it is possible to train a ranker by using click data collected from the same search system. Click data is indicative of individual users' relevance judgments and is relatively easy to collect with low cost. On the other hand, it is also noisy and biased [14, 27]. Notably, the orders of documents in the search results affect users' judgments. Users tend to more frequently click documents presented at higher positions, which is called position bias. This has been preventing practical use of click data in learning-to-rank.

Recently a new research direction, referred to as unbiased learning-to-rank, is arising and making progress. Unbiased learning-to-rank aims at eliminating bias in click data, particularly position bias, and making use of the debiased data to train a ranker. Wang et al. [24] and Joachims et al. [15] respectively propose employing the inverse propensity weighting (IPW) principle [23] to learn an 'unbiased ranker' from click data. It is proved that the objective function in learning using IPW is an unbiased estimate of the risk function defined on a relevance measure (a pointwise loss). The authors also develop methods for estimating position bias by randomization of search results online. Wang et al. [25] further develop a method for estimating position bias from click data offline. More recently Ai et al. [1] propose a method that can jointly estimate position bias and train a ranker from click data, again on the basis of IPW. In the previous work, the IPW principle is limited to the pointwise setting in the sense that position biases are only defined on click positions.

In this paper, we address the problem of jointly estimating position biases and training a ranker from click data for pairwise learning-to-rank, particularly using a pairwise algorithm, LambdaMART. To do so, we extend the inverse propensity weighting principle to the pairwise setting, and develop a new method for jointly conducting position bias estimation and ranker training.

Specifically, we give a formulation of unbiased learning-to-rank for the pairwise setting and extend the IPW principle. We define position biases as the ratio of the click probability to the relevance probability at each position, as well as the ratio of the unclick probability to the irrelevance probability. This definition takes both the position biases at click positions and those at unclick positions into consideration. We prove that under the extended IPW principle, the objective function becomes an unbiased estimate of risk function

defined on pairwise loss functions. In this way, we can learn an unbiased ranker using a pairwise ranking algorithm.

We then develop a method for jointly estimating position biases for both click and unclick positions and training a ranker for pairwise learning-to-rank, called Pairwise Debiasing. The position bias and the ranker can be iteratively learned through minimization of the same objective function. As an instance, we further develop Unbiased LambdaMART*, an algorithm of learning an unbiased ranker using LambdaMART.

Experiments on the Yahoo learning-to-rank challenge benchmark dataset demonstrate that Unbiased LambdaMART can effectively conduct debiasing of click data and significantly outperform the baseline algorithms in terms of all measures, for example, 3-4% improvements in terms of NDCG@1. An online A/B Testing at a commercial news search engine, Jinri Toutiao, also demonstrates that Unbiased LambdaMART can enhance the performance of relevance ranking at the search engine.

The contribution of this paper includes the following proposals.

- A general framework on unbiased learning-to-rank in the pairwise setting, particularly, an extended IPW.
- Pairwise Debiasing, a method for jointly estimating position bias and training a pairwise ranker.
- Unbiased LambdaMART, an algorithm of unbiased pairwise learning-to-rank using LambdaMART.

2 RELATED WORK

In this section, we introduce related work on learning-to-rank, click model, and unbiased learning to rank.

2.1 Learning-to-Rank

Learning-to-rank is to automatically construct a ranking model from data, referred to as a ranker, for ranking in search. A ranker is usually defined as a function of feature vector based on a query document pair. In search, given a query, the retrieved documents are ranked based on the scores of the documents given by the ranker. The advantage of employing learning-to-rank is that one can build a ranker without the need of manually creating it, which is usually tedious and hard. Learning-to-rank is now becoming a standard technique for search.

There are many algorithms proposed for learning-to-rank. The algorithms can be categorized as pointwise approach, pairwise approach, and listwise approach, based on the loss functions in learning [18, 19, 21]. The pairwise and listwise algorithms usually work better than the pointwise algorithms [19], because the key issue of ranking in search is to determine the orders of documents but not to judge the relevance of documents, which is exactly the goal of the pairwise and listwise algorithms. For example, the pairwise algorithms of RankSVM [6, 13] and LambdaMART [4, 26] are state-of-the-art algorithms for learning-to-rank.

Traditionally, data for learning a ranker is manually labeled by humans, which can be costly. To deal with the problem, one may consider using click data as labeled data to train a ranker. Click data records the documents clicked by the users after they submit queries, and it naturally represents users' implicit relevance judgments on the search results. The utilization of click data has both pros and cons. On one hand, it is easy to collect a large amount of click data with low cost. On the other hand, click data is very noisy and has position bias, presentation bias, etc. Position bias means that users tend to more frequently click documents ranked at higher positions [14, 27]. How to effectively cope with position bias and leverage click data for learning-to-rank thus becomes an important research issue.

2.2 Click Model

One direction of research on click data aims to design a click model to simulate users' click behavior, and then estimate the parameters of the click model from data. It then makes use of the learned click model for different tasks, for example, use them as features of a ranker.

Several probabilistic models have been developed. For example, Richardson et al. [22] propose the Position Based Model (PBM), which assumes that a click only depends on the position and relevance of the document. Craswell et al. [9] develop the Cascade Model (CM), which formalizes the user's behavior in browsing of search results as a sequence of actions. Dupret et al. [10] propose the User Browsing Model (UBM), asserting that a click depends not only on the position of a document, but also on the positions of the previously clicked documents. Chapelle et al. [8] develop the Dynamic Bayesian Network Model (DBN), based on the assumption that the user's behavior after a click does not depend on the perceived relevance of the document but on the actual relevance of the document. Borisov et al. [3] develop the Neural Click Model, which utilizes neural networks and vector representations to predict user's click behavior. Recently, Kveton et al. [17] propose a multi-armed bandit learning algorithm of the Cascade Model to identify the k most attractive items in the ranking list. Li et al. [20] makes use of click models to evaluate the performance of ranking model offline.

Click models can be employed to estimate position bias and other biases, as well as query document relevance. They are not designed only for the purpose of debiasing and thus could be suboptimal for the task. In our experiments, we use the click models to generate synthetic click datasets for evaluating our proposed unbiased learning-to-rank algorithm offline.

2.3 Unbiased Learning-to-Rank

Recently, a new direction in learning-to-rank, referred to as unbiased learning-to-rank, is arising and making progress. The goal of unbiased learning-to-rank is to develop new techniques to conduct debiasing of click data and leverage the debiased click data in training of a ranker[2].

Wang et al. [24] apply unbiased learning-to-rank to personal search. They conduct randomization of search results to estimate query-level position bias and adjust click data for training of a ranker in personal search on the basis of inverse propensity weighting (IPW) [23]. Joachims et al. [15] theoretically prove that with the inverse propensity weighting (IPW) principle, one can obtain an unbiased estimate of a risk function on relevance in learning-to-rank. They also utilize online randomization to estimate position bias and perform training of a RankSVM model. Wang et al. [25] employ a regression-based EM algorithm to infer position bias by

*Code is available at https://github.com/acbull/Unbiased_LambdaMart

Table 1: A summary of notations.

q, D_q	query q and documents D_q of q
i, d_i, x_i, r_i, c_i	i-th (representing the position by original ranker where click data is collected) document d_i in D_q with feature vector x_i, relevance information r_i (1/0) and click information c_i (1/0)
$I_q = \{(d_i, d_j)\}$	set of pairs of documents of q, in which d_i is more relevant or more clicked than d_j
C_q, $\mathcal{D} = \{(q, D_q, C_q)\}$	click information C_q of D_q and click data set $\mathcal{D}$ for all queries

maximizing the likelihood of click data. The estimated position bias is then utilized in learning of LambdaMART. Recently, Ai et al. [1] design a dual learning algorithm which can jointly learn an unbiased propensity model for representing position bias and an unbiased ranker for relevance ranking, by optimizing two objective functions. Both models are implemented as neural networks. Their method is also based on IPW, while the loss function is a pointwise loss function.

Our work mainly differs from the previous work in the following points:

- In previous work, position bias (propensity) is defined as the observation probability, and thus IPW is limited to the pointwise setting in which the loss function is pointwise and debiasing is performed at a click position each time. In this work, we give a more general definition of position bias (propensity), and extend IPW to the pairwise setting, in which the loss function is pairwise and debiasing is carried out at both click positions and unclick positions.
- In previous work, estimation of position bias either relies on randomization of search results online, which can hurt user experiences [15, 24], or resorts to separate learning of a propensity model from click data offline, which can be suboptimal to relevance ranking [1, 25]. In this paper, we propose to jointly conduct estimation of position bias and learning of a ranker through minimizing one objective function. We further apply this framework to the state-of-the-art LambdaMART algorithm.

3 FRAMEWORK

In this section, we give a general formulation of unbiased learning-to-rank, for both the pointwise and pairwise settings. We also extend the inverse propensity weighting principle to the pairwise setting.

3.1 Pointwise Unbiased Learning-to-Rank

In learning-to-rank, given a query document pair denoted as x, the ranker f assigns a score to the document. The documents with respect to the query are then ranked in descending order of their scores. Traditionally, the ranker is learned with labeled data. In the pointwise setting, the loss function in learning is defined on a single data point x.

Let q denote the query and D_q the set of documents associated with q. Let d_i denote the i-th document in D_q and x_i the feature vector of q and d_i (see Table 1). Let r_i^+ and r_i^- represent that d_i is relevant and irrelevant respectively (i.e., $r_i = 1$ and $r_i = 0$). For simplicity we only consider binary relevance here and one can easily extend it to the multi-level relevance case. The risk function in learning is defined as

$$R_{rel}(f) = \int L(f(x_i), r_i^+)\, dP(x_i, r_i^+) \tag{1}$$

where f denotes a ranker, $L(f(x_i), r_i^+)$ denotes a pointwise loss function based on an IR measure [15] and $P(x_i, r_i^+)$ denotes the probability distribution on x_i and r_i^+. Most ranking measures in IR only utilize relevant documents in their definitions, and thus the loss function here is defined on relevant documents with label r_i^+. Furthermore, the position information of documents is omitted from the loss function for notation simplicity.

Suppose that there is a labeled dataset in which the relevance of documents with respect to queries is given. One can learn a ranker $\hat{f}_{rel}$ through the minimization of the empirical risk function (objective function) as follows.

$$\hat{f}_{rel} = \arg\min_f \sum_q \sum_{d_i \in D_q} L(f(x_i), r_i^+) \tag{2}$$

One can also consider using click data as relevance feedbacks from users, more specifically, viewing clicked documents as relevant documents and unclicked documents as irrelevant documents, and training a ranker with a click dataset. This is what we call 'biased learning-to-rank', because click data has position bias, presentation bias, etc. Suppose that there is a click dataset in which the clicks of documents with respect to queries by an original ranker are recorded. For convenience, let us assume that document d_i in D_q is exactly the document ranked at position i by the original ranker. Let c_i^+ and c_i^- represent that document d_i is clicked and unclicked in the click dataset respectively (i.e., $c_i = 1$ and $c_i = 0$). The risk function and minimization of empirical risk function can be defined as follows.

$$R_{click}(f) = \int L(f(x_i), c_i^+)\, dP(x_i, c_i^+) \tag{3}$$

$$\hat{f}_{click} = \arg\min_f \sum_q \sum_{d_i \in D_q} L(f(x_i), c_i^+) \tag{4}$$

The loss function is defined on clicked documents with label c_i^+. The ranker $\hat{f}_{click}$ learned in this way is biased, however.

Unbiased learning-to-rank aims to eliminate the biases, for example position bias, in the click data and train a ranker with the debiased data. The training of ranker and debiasing of click data can be performed simultaneously or separately. The key question is how to fill the gap between click and relevance, that is, $P(c_i^+|x_i)$ and $P(r_i^+|x_i)$. Here we assume that the click probability is proportional to the relevance probability at each position, where the ratio $t_i^+ > 0$ is referred to as bias at a click position i.

$$P(c_i^+|x_i) = t_i^+ P(r_i^+|x_i) \tag{5}$$

There are k ratios corresponding to k positions. The ratios can be affected by different types of bias, but in this paper, we only consider position bias.

We can conduct learning of an unbiased ranker $\hat{f}_{unbiased}$, through minimization of the empirical risk function as follows.

$$R_{unbiased}(f) = \int \frac{L(f(x_i), c_i^+)}{t_i^+} \, dP(x_i, c_i^+) \tag{6}$$

$$= \int \frac{L(f(x_i), c_i^+)}{\frac{P(c_i^+|x_i)}{P(r_i^+|x_i)}} \, dP(x_i, c_i^+) \tag{7}$$

$$= \int L(f(x_i), c_i^+) \, dP(x_i, r_i^+) \tag{8}$$

$$= \int L(f(x_i), r_i^+) \, dP(x_i, r_i^+) = R_{rel}(f) \tag{9}$$

$$\hat{f}_{unbiased} = \arg\min_f \sum_q \sum_{d_i \in D_q} \frac{L(f(x_i), c_i^+)}{t_i^+} \tag{10}$$

In (9) click label c_i^+ in the loss function is replaced with relevance label r_i^+, because after debiasing click implies relevance.

One justification of this method is that $R_{unbiased}$ is in fact an unbiased estimate of R_{rel}. This is the so-called inverse propensity weighting (IPW) principle proposed in previous work. That is to say, if we can properly estimate position bias (ratio) t_i^+, then we can reliably train an unbiased ranker $\hat{f}_{unbiased}$.

An intuitive explanation of position bias (ratio) t_i^+ can be found in the following relation, under the assumption that a clicked document must be relevant ($c^+ \Rightarrow r^+$).

$$t_i^+ = \frac{P(c_i^+|x_i)}{P(r_i^+|x_i)} = \frac{P(c_i^+, r_i^+|x_i)}{P(r_i^+|x_i)} = P(c_i^+|r_i^+, x_i) \tag{11}$$

It means that t_i^+ represents the conditional probability of how likely a relevant document is clicked at position i after examination of the document. In the original IPW, t_i^+ is defined as the observation probability that the user examines the document at position i before clicking the document [15, 25], which is based on the same assumption as (11).

3.2 Pairwise Unbiased Learning-to-Rank

In the pairwise setting, the ranker f is still defined on a query document pair x, and the loss function is defined on two data points x_i and x_j. Traditionally, the ranker is learned with labeled data.

Let q denote a query. Let d_i and d_j denote the i-th and j-th documents with respect to query q. Let x_i and x_j denote the feature vectors from d_i and d_j as well as q. Let r_i^+ and r_j^- represent that document d_i and document d_j are relevant and irrelevant respectively. Let I_q denote the set of document pairs (d_i, d_j) where d_i is relevant and d_j is irrelevant. For simplicity we only consider binary relevance here and one can easily extend it to the multi-level relevance case. The risk function and the minimization of empirical risk function are defined as

$$R_{rel}(f) = \int L(f(x_i), r_i^+, f(x_j), r_j^-) \, dP(x_i, r_i^+, x_j, r_j^-) \tag{12}$$

$$\hat{f}_{rel} = \arg\min_f \sum_q \sum_{(d_i, d_j) \in I_q} L(f(x_i), r_i^+, f(x_j), r_j^-) \tag{13}$$

where $L(f(x_i), r_i^+, f(x_j), r_j^-)$ denotes a pairwise loss function.

One can consider using click data to directly train a ranker, that is, to conduct 'biased learning-to-rank'. Let c_i^+ and c_j^- represent that document d_i and document d_j are clicked and unclicked respectively. Let I_q denote the set of document pairs (d_i, d_j) where d_i is clicked and d_j is unclicked. The risk function and minimization of empirical risk function can be defined as follows.

$$R_{click}(f) = \int L(f(x_i), c_i^+, f(x_j), c_j^-) \, dP(x_i, c_i^+, x_j, c_j^-) \tag{14}$$

$$\hat{f}_{click} = \arg\min_f \sum_q \sum_{(d_i, d_j) \in I_q} L(f(x_i), c_i^+, f(x_j), c_j^-) \tag{15}$$

The ranker $\hat{f}_{click}$ is however biased.

Similar to the pointwise setting, we consider dealing with position bias in the pairwise setting and assume that the click probability is proportional to the relevance probability at each position and the unclick probability is proportional to the irrelevance probability at each position. The ratios $t_i^+ > 0$ and $t_j^- > 0$ are referred to as position biases at a click position i and an unclick position j.

$$P(c_i^+|x_i) = t_i^+ P(r_i^+|x_i) \tag{16}$$

$$P(c_j^-|x_j) = t_j^- P(r_i^-|x_j) \tag{17}$$

There are $2k$ position biases (ratios) corresponding to k positions.

We can conduct learning of an unbiased ranker $\hat{f}_{unbiased}$, through minimization of the empirical risk function as follows.

$$R_{unbiased}(f) = \int \frac{L(f(x_i), c_i^+, f(x_j), c_j^-)}{t_i^+ \cdot t_j^-} \, dP(x_i, c_i^+, x_j, c_j^-) \tag{18}$$

$$= \int\int \frac{L(f(x_i), c_i^+, f(x_j), c_j^-) dP(c_i^+, x_i) dP(c_j^-, x_j)}{\frac{P(c_i^+|x_i)P(c_j^-|x_j)}{P(r_i^+|x_i)P(r_i^-|x_j)}} \tag{19}$$

$$= \int\int L(f(x_i), c_i^+, f(x_j), c_j^-) dP(r_i^+, x_i) dP(r_i^-, x_j) \tag{20}$$

$$= \int L(f(x_i), r_i^+, f(x_j), r_j^-) dP(x_i, r_i^+, x_j, r_j^-) \tag{21}$$

$$= R_{rel}(f) \tag{22}$$

$$\hat{f}_{unbiased} = \arg\min_f \sum_q \sum_{(d_i, d_j) \in I_q} \frac{L(f(x_i), c_i^+, f(x_j), c_j^-)}{t_i^+ \cdot t_j^-} \tag{23}$$

In (18) it is assumed that relevance and click at position i are independent from those at position j. (Experimental results show that the proposed Unbiased LambdaMART works very well under this assumption, even one may think that it is strong.) In (21), click labels c_i^+ and c_j^- are replaced with relevance labels r_i^+ and r_j^- because after debiasing click implies relevance and unclick implies irrelevance.

One justification of this method is that $R_{unbiased}$ is an unbiased estimate of R_{rel}. Therefore, if we can accurately estimate the position biases (ratios), then we can reliably train an unbiased ranker $\hat{f}_{unbiased}$. This is an extension of the inverse propensity weighting (IPW) principle to the pairwise setting.

Position bias (ratio) t_i^+ has the same explanation as that in the pointwise setting. An explanation of position bias (ratio) t_j^- is that it represents the reciprocal of the conditional probability of how likely an unclicked document is irrelevant at position j, as shown

below.

$$t_j^- = \frac{P(c_j^-|x_j)}{P(r_i^-|x_j)} = \frac{P(c_j^-|x_j)}{P(r_j^-, c_j^-|x_j)} = \frac{1}{P(r_j^-|c_j^-, x_j)} \tag{24}$$

It is under the assumption that an irrelevant document must be unclicked ($r^- \Rightarrow c^-$), which is equivalent to ($c^+ \Rightarrow r^+$). Note that t_j^- is not a probability and it has a different interpretation from t_i^+. The unclicked document j can be either examined or unexamined. Thus, in the extended IPW the condition on examination of document in the original IPW is dropped.

4 APPROACH

In this section, we present Pairwise Debiasing as a method of jointly estimating position bias and training a ranker for unbiased pairwise learning-to-rank. Furthermore, we apply Pairwise Debiasing on LambdaMart and describe the learning algorithm of Unbiased LambdaMART.

4.1 Learning Strategy

We first give a general strategy for pairwise unbiased learning-to-rank, named Pairwise Debiasing.

A key issue of unbiased learning-to-rank is to accurately estimate position bias. Previous work either relies on randomization of search results online, which can hurt user experiences [15, 24], or resorts to a separate learning of position bias from click data offline, which can be suboptimal to the ranker [1, 25]. In this paper, we propose to simultaneously conduct estimation of position bias and learning of a ranker offline through minimizing the following regularized loss function (objective function).

$$\min_{f, t^+, t^-} \mathcal{L}(f, t^+, t^-) \tag{25}$$

$$= \min_{f, t^+, t^-} \sum_q \sum_{(d_i, d_j) \in I_q} \frac{L(f(x_i), c_i^+, f(x_j), c_j^-)}{t_i^+ \cdot t_j^-} + ||t^+||_p^p + ||t^-||_p^p \tag{26}$$

$$s.t.\ t_1^+ = 1,\ t_1^- = 1 \tag{27}$$

where f denotes a ranker, t^+ and t^- denote position biases (ratios) at all positions, L denotes a pairwise loss function, $||\cdot||_p^p$ denotes L_p regularization. Because the position biases are relative values with respect to positions, to simplify the optimization process we fix the position biases of the first position to 1 and only learn the (relative) position biases of the rest of the positions. Here $p \in [0, +\infty)$ is a hyper-parameter. The higher the value of p is, the more regularization we impose on the position biases.

In the objective function, the position biases t^+ and t^- are inversely proportional to the pairwise loss function $L(f(x_i), c_i^+, f(x_j), c_j^-)$, and thus the estimated position biases will be high if the losses on those pairs of positions are high in the minimization. The position biases are regularized and constrained to avoid a trivial solution of infinity.

It would be difficult to directly optimize the objective function in (26). We adopt a greedy approach to perform the task. Specifically, for the three optimization variables f, t^+, t^-, we iteratively optimize the objective function $\mathcal{L}$ with respect to one of them with the others fixed; we repeat the process until convergence.

4.2 Estimation of position bias ratios

Given a fixed ranker, we can estimate the position biases at all positions. There are in fact closed form solutions for the estimation.

The partial derivative of objective function $\mathcal{L}$ with respect to position bias t^+ is

$$\frac{\partial \mathcal{L}(f^*, t^+, (t^-)^*)}{\partial t_i^+} = \sum_q \sum_{j:(d_i, d_j) \in I_q} \frac{L(f^*(x_i), c_i^+, f^*(x_j), c_j^-)}{-(t_i^+)^2 \cdot (t_j^-)^*} + p \cdot (t_i^+)^{p-1} \tag{28}$$

Thus, we have*

$$\arg\min_{t_i^+} \mathcal{L}(f^*, t^+, (t^-)^*) = \left[\sum_q \sum_{j:(d_i, d_j) \in I_q} \frac{L(f^*(x_i), c_i^+, f^*(x_j), c_j^-)}{p \cdot (t_j^-)^*} \right]^{\frac{1}{p+1}} \tag{29}$$

$$t_i^+ = \left[\frac{\sum_q \sum_{j:(d_i, d_j) \in I_q} (L(f^*(x_i), c_i^+, f^*(x_j), c_j^-) \ / \ (t_j^-)^*)}{\sum_q \sum_{k:(d_1, d_k) \in I_q} (L(f^*(x_1), c_1^+, f^*(x_k), c_k^-) \ / \ (t_k^-)^*)} \right]^{\frac{1}{p+1}} \tag{30}$$

In (30) the result is normalized to make the position bias at the first position to be 1.

Similarly, we have

$$t_j^- = \left[\frac{\sum_q \sum_{i:(d_i, d_j) \in I_q} (L(f^*(x_i), c_i^+, f^*(x_j), c_j^-) \ / \ (t_i^+)^*)}{\sum_q \sum_{k:(d_k, d_1) \in I_q} (L(f^*(x_k), c_k^+, f^*(x_1), c_1^-) \ / \ (t_k^+)^*)} \right]^{\frac{1}{p+1}} \tag{31}$$

In this way, we can estimate the position biases (ratios) t^+ and t^- in one step given a fixed ranker f^*. Note that the method here, referred to as Pairwise Debiasing, can be applied to any pairwise loss function. In this paper, we choose to apply the pairwise learning-to-rank algorithm LambdaMART.

4.3 Learning of Ranker

Given fixed position biases, we can learn an unbiased ranker. The partial derivative of $\mathcal{L}$ with respect to f can be written in the following general form.

$$\frac{\partial \mathcal{L}(f, (t^+)^*, (t^-)^*)}{\partial f} = \sum_q \sum_{(d_i, d_j) \in I_q} \frac{1}{(t_i^+)^* \cdot (t_j^-)^*} \frac{\partial L(f(x_i), c_i^+, f(x_j), c_j^-)}{\partial f} \tag{32}$$

We employ LambdaMART to train a ranker. LambdaMART [5, 26] employs gradient boosting or MART [11] and the gradient function of the loss function called lambda function. Given training data, it performs minimization of the objective function using the lambda function.

In LambdaMART, the lambda gradient λ_i of document d_i is calculated using all pairs of the other documents with respect to the query.

$$\lambda_i = \sum_{j:(d_i, d_j) \in I_q} \lambda_{ij} - \sum_{j:(d_j, d_i) \in I_q} \lambda_{ji} \tag{33}$$

*The derivation is based on the fact $p \in (0, +\infty)$. The result is then extended to the case of $p = 0$.

$$\lambda_{ij} = \frac{-\sigma}{1 + e^{\sigma(f(x_i) - f(x_j))}} \left|\Delta Z_{ij}\right| \tag{34}$$

where λ_{ij} is the lambda gradient defined on a pair of documents d_i and d_j, σ is a constant with a default value of 2, $f(x_i)$ and $f(x_j)$ are the scores of the two documents given by LambdaMART, ΔZ_{ij} denotes the difference between NDCG[12] scores if documents d_i and d_j are swapped in the ranking list.

Following the discussion above, we can make an adjustment on the lambda gradient $\tilde{\lambda}_i$ with the estimated position biases:

$$\tilde{\lambda}_i = \sum_{j:(d_i,d_j)\in I_q} \tilde{\lambda}_{ij} - \sum_{j:(d_j,d_i)\in I_q} \tilde{\lambda}_{ji} \tag{35}$$

$$\tilde{\lambda}_{ij} = \frac{\lambda_{ij}}{(t_i^+)^* \cdot (t_j^-)^*} \tag{36}$$

Thus, by simply replacing the lambda gradient λ_i in LambdaMART with the adjusted lambda gradient $\tilde{\lambda}_i$, we can reliably learn an unbiased ranker with the LambdaMART algorithm. We call the algorithm Unbiased LambdaMART.

Estimation of position biases in (30) and (31) needs calculation of the loss function $L_{ij} = L(f(x_i), c_i^+, f(x_j), c_j^-)$. For LambdaMART the loss function can be derived from (34) as follows.

$$L_{ij} = \log(1 + e^{-\sigma(f(x_i) - f(x_j))}) \left|\Delta Z_{ij}\right| \tag{37}$$

4.4 Learning Algorithm

The learning algorithm of Unbiased LambdaMART is given in Algorithm 1. The input is a click dataset $\mathcal{D}$. The hyper-parameters are regularization parameter p and total number of boosting iterations M. The output is an unbiased ranker f and estimated position biases at all positions t^+, t^-. As is outlined in Algorithm 1, Unbiased LambdaMART iteratively calculates adjusted lambda gradient in line 4, re-trains a ranker with the gradients in line 6, and re-estimates position biases in line 7. The time complexity of Unbiased LambdaMART is the same as that of LambdaMART.

Algorithm 1 Unbiased LambdaMART

Require: click dataset $\mathcal{D} = \{(q, D_q, C_q)\}$; hyper-parameters p, M;
Ensure: unbiased ranker f; position biases (ratios) t^+ and t^-;
1: Initialize all position biases (ratios) as 1;
2: **for** $m = 1$ to M **do**
3: **for** each query q and each document d_i in D_q **do**
4: Calculate $\tilde{\lambda}_i$ with $(t^+)^*$ and $(t^-)^*$ using (35) and (36);
5: **end for**
6: Re-train ranker f with $\tilde{\lambda}$ using LambdaMART algorithm
7: Re-estimate position biases (ratios) t^+ and t^- with ranker f^* using (30) and (31)
8: **end for**
9: **return** f, t^+, and t^-;

5 EXPERIMENTS

In this section, we present the results of two experiments on our proposed algorithm Unbiased LambdaMART. One is an offline experiment on a benchmark dataset, together with empirical analysis on the effectiveness, generalizability, and robustness of the algorithm. The other experiment is an online A/B testing at a commercial search engine.

5.1 Experiment on Benchmark Data

We made use of the Yahoo! learning-to-rank challenge dataset[†] to conduct an experiment. The Yahoo dataset is one of the largest benckmark dataset for learning-to-rank. It consists of 29921 queries and 710k documents. Each query document pair is represented by a 700-dimensional feature vector manually assigned with a label denoting relevance at 5 levels [7].

There is no click data associated with the Yahoo dataset. We followed the procedure in [1] to generate synthetically click data from the Yahoo dataset for offline evaluation.[‡]

We chose NDCG at position 1,3,5,10 and MAP as evaluation measures in relevance ranking.

5.1.1 Click Data Generation. The click data generation process in [1] is as follows. First, one trains a Ranking SVM model using 1% of the training data with relevance labels, and uses the model to create an initial ranking list for each query. Next, one samples clicks from the ranking lists by simulating the browsing process of users. The position-based click model (PBM) is utilized. It assumes that a user decides to click a document according to probability $P(c_i^+) = P(o_i^+)P(r_i^+)$. Here $P(o_i^+)$ and $P(r_i^+)$ are the observation probability and relevance probability respectively.

The probability of observation $P(o_i^+)$ is calculated by

$$P(o_i^+|x_i) = \rho_i^\theta$$

where ρ_i represents position bias at position i and $\theta \in [0, +\infty]$ is a parameter controlling the degree of position bias. The position bias ρ_i is obtained from an eye-tracking experiment in [14] and the parameter θ is set as 1 by default.

The probability of relevance $P(r_i^+)$ is calculated by

$$P(r_i^+) = \epsilon + (1 - \epsilon)\frac{2^y - 1}{2^{y_{max}} - 1}$$

where $y \in [0, 4]$ represents a relevance level and y_{max} is the highest level of 4. The parameter ϵ represents click noise due to that irrelevant documents ($y = 0$) are incorrectly perceived as relevant documents ($y > 0$) , which is set as 0.1 by default.

5.1.2 Baseline Methods. We made comprehensive comparisons between our method and the baselines. The baselines were created by combining the state-of-the-art debiasing methods and learning-to-rank algorithms. There were six debiasing methods. To make fair comparison, we used click model to generate 165660 query sessions as training dataset, and utilized the same dataset for all debiasing methods. All the hyper-parameters of the baseline models were the same as those in the original papers.

Randomization: The method, proposed by Joachims et al. [15], uses randomization to infer the observation probabilities as position biases. We randomly shuffled the rank lists and then estimated the position biases as in [1].

[†] http://webscope.sandbox.yahoo.com
[‡] We plan to release the synthetically generated click data as well as the source code of Unbiased LambdaMART.

Table 2: Comparison of different unbiased learning-to-rank methods.

Ranker	Debiasing Method	MAP	NDCG@1	NDCG@3	NDCG@5	NDCG@10
LambdaMART	Labeled Data (Upper Bound)	0.854	0.745	0.745	0.757	0.790
	Pairwise Debiasing	**0.836**	**0.717**	**0.716**	**0.728**	**0.764**
	Regression-EM [25]	0.830	0.685	0.684	0.700	0.743
	Randomization	0.827	0.669	0.678	0.690	0.728
	Click Data (Lower Bound)	0.820	0.658	0.669	0.672	0.716
DNN	Labeled Data (Upper Bound)	0.831	0.677	0.685	0.705	0.737
	Dual Learning Algorithm [1]	0.828	0.674	0.683	0.697	0.734
	Regression-EM	0.829	0.676	0.684	0.699	0.736
	Randomization	0.825	0.673	0.679	0.693	0.732
	Click Data (Lower Bound)	0.819	0.637	0.651	0.667	0.711
RankSVM	Labeled Data (Upper Bound)	0.815	0.631	0.649	0.675	0.707
	Regression-EM	0.815	0.629	0.648	0.674	0.705
	Randomization [15]	0.814	0.628	0.644	0.672	0.707
	Click Data (Lower Bound)	0.811	0.614	0.629	0.658	0.697

Regression-EM: The method, proposed by Wang et al. [25], directly estimates the position biases using a regression-EM model implemented by GBDT.

Dual Learning Algorithm: The method, proposed by Ai et al. [1], jointly learns a ranker and conducts debiasing of click data. The algorithm implements both the ranking model and the debiasing model as deep neural networks.

Pairwise Debiasing: Our proposed debiasing method, which is combined with LambdaMART. In this experiment, we set the hyper-parameter p as 0 by default. As explained below, a further experiment was conducted with different values of p.

Click Data: In this method, the raw click data without debiasing is used to train a ranker, whose performance is considered as a lower bound.

Labeled Data: In this method, human annotated relevance labels without any bias are used as data for training of ranker, whose performance is considered as an upper bound.

There were three learning-to-rank algorithms.

DNN: A deep neural network was implemented as a ranker, as in [1]. We directly used the code provided by Ai et al.[§].

RankSVM: We directly used the Unbiased RankSVM Software provided by Joachims et al.[¶], with hyper-parameter C being 200.

LambdaMART: We implemented Unbiased LambdaMART by modifying the LambdaMART tool in LightGBM [16]. We utilized the default hyper-parameters of the tool. The total number of trees was 300, learning rate was 0.05, number of leaves for one tree was 31, feature fraction was 0.9, and bagging fraction was 0.9.

In summary, there were 13 baselines to compare with our proposed Unbiased LambdaMART algorithm. Note that Dual Learning Algorithm and DNN are tightly coupled. We did not combine Pairwise Debiasing with RankSVM and DNN, as it is beyond the scope of this paper.

[§] https://github.com/QingyaoAi/Dual-Learning-Algorithm-for-Unbiased-Learning-to-Rank
[¶] https://www.cs.cornell.edu/people/tj/svm_light/svm_proprank.html

5.1.3 Experimental Results. Table 2 summarizes the results. We can see that our method of Unbiased LambdaMART (LambdaMART + Pairwise Debiasing) significantly outperforms all the other baseline methods. The results of Regression-EM and Dual Learning Algorithm are compariable with those reported in the original papers. In particular, we have the following findings.

- Our method of LambdaMART+Pairwise Debiasing (Unbiased LambdaMART) achieves better performances than all the state-of-the-art methods in terms of all measures. For example, in terms of NDCG@1, our method outperforms LambdaMART+Regression-EM by 3.2%, outperforms DNN+Dual Learning by 4.3%, and outperforms RankSVM+Randomization by 8.9%.
- Pairwise Debiasing works better than the other debiasing methods. When combined with LambdaMART, Pairwise Debiasing outperforms Regression-EM by 3.2%, outperforms Randomization by 4.8% in terms of NDCG@1.
- LambdaMART trained with human labeled data achieves the best performance (upper bound). An unbiased learning-to-rank algorithm can still not beat it. This indicates that there is still room for improvement in unbiased learning-to-rank.
- When trained with Click Data, the performance of LambdaMART decreases significantly and gets closer to those of RankSVM and DNN. This implies that a sophisticated algorithm like LambdaMART is more sensitive to position bias.

5.2 Empirical Analysis

We conducted additional experiments to investigate the effectiveness, generalizability, and robustness of Unbiased LambdaMART.

5.2.1 Effectiveness of Unbiased LambdaMART. We investigated whether the performance improvement by Unbiased LambdaMART is indeed from reduction of position bias.

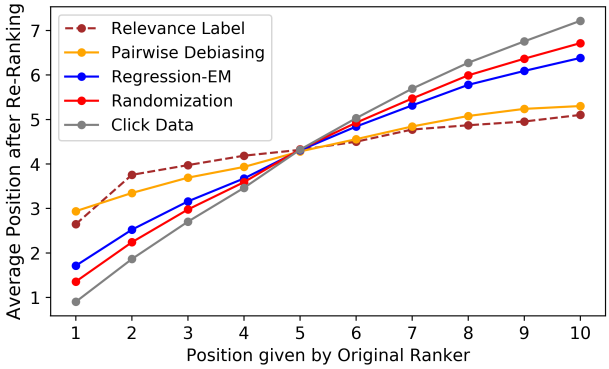


Figure 1: Average positions after re-ranking of documents at each original position by different debiasing methods with LamdbaMART.

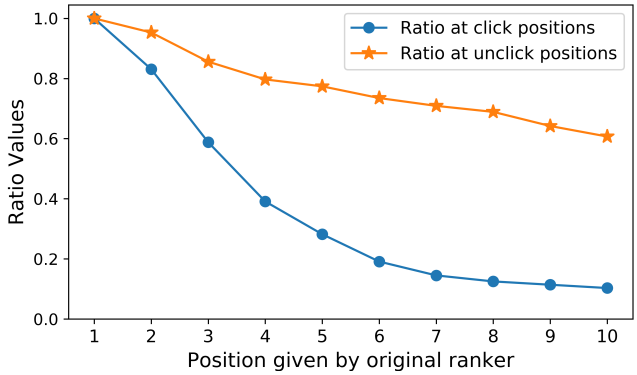


Figure 2: Position biases (ratios) at click and unclick positions estimated by Unbiased LambdaMART.

We first identified the documents at each position given by the original ranker. We then calculated the average positions of the documents at each original position after re-ranking by Pairwise Debiasing and the other debiasing methods, combined with LambdaMART. We also calculated the average positions of the documents after re-ranking by their relevance labels, which is the ground truth. Ideally, the average positions by the debiasing methods should get close to the average positions by the relevance labels. Figure 1 shows the results.

One can see that the curve of LambdaMART + Click Data (in grey) is away from that of relevance labels or ground truth (in brown), indicating that directly using click data without debiasing can be problematic. The curve of Pairwise Debiasing (in orange) is the closest to the curve of relevance labels, indicating that the performance enhancement by Pairwise Debiasing is indeed from effective debiasing.

Figure 2 shows the normalized (relative) position biases for click and unclick positions given by Unbiased LambdaMART. The result indicates that both the position biases at click positions and position biases at unclick positions monotonically decrease, while the former decrease at a faster rate than the latter. The result exhibits how Unbiased LambdaMART can reduce position biases in the pairwise setting.

5.2.2 Generalizability of Unbiased LambdaMART. The Position Based Model (PBM) assumes that the bias of a document only depends on its position, which is an approximation of user click behavior in practice. The Cascade Model [10], on the other hand, assumes that the user browses the search results in a sequential order from top to bottom, which may more precisely model user behavior. We therefore analyzed the generalizability of Unbiased LambdaMART by using simulated click data from both Position Based Model and Cascade Model, and studied whether regularization of position bias, i.e., hyper-parameter p, affects performance.

We used a variant of Cascade Model which is similar to Dynamic Bayesian Model in [8]. There is a probability ϕ that the user is satisfied with the result after clicking the document. If the user is satisfied, he / she will stop searching; and otherwise, there is a probability β that he / she will examine the next result and there is a probability $1-\beta$ that he / she will stop searching. Obviously, the smaller β indicates that the user will have a smaller probability to continue reading, which means a more severe position bias. In our experiment, we set ϕ as half of the relevance probability and used the default value of β i.e., 0.5.

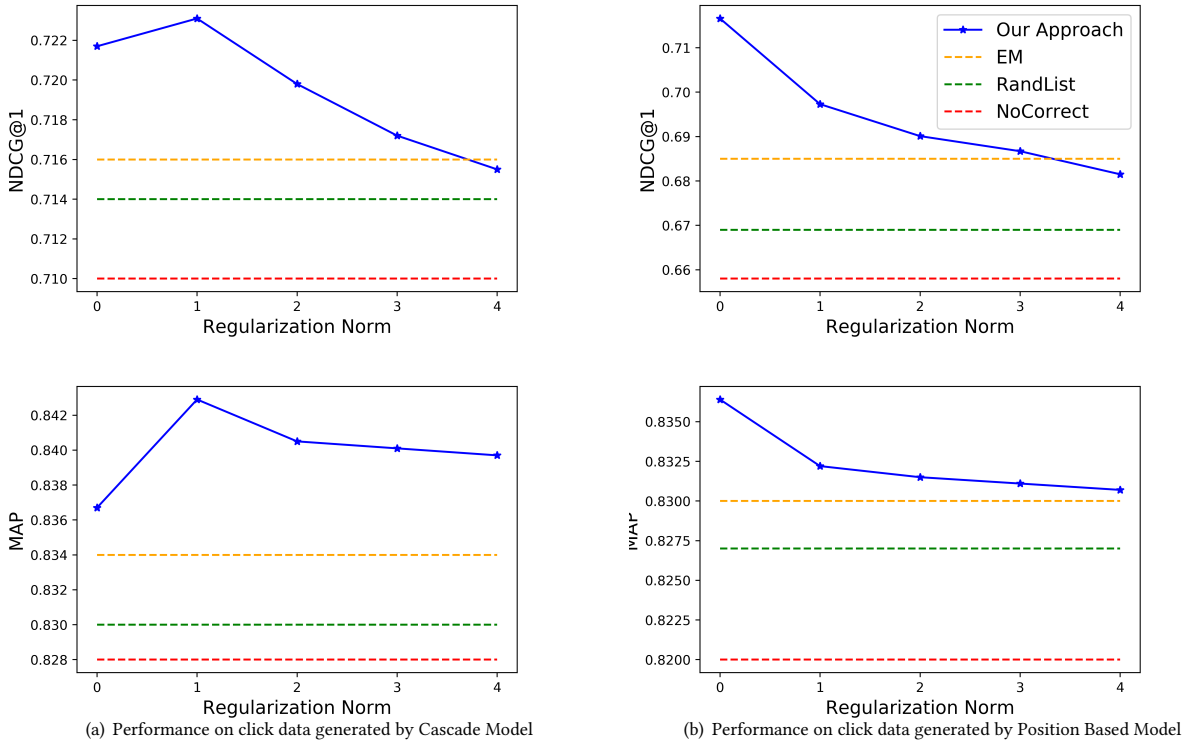
We compared Unbiased LambdaMART (LambdaMART + Pairwise Debiasing) with LambdaMART + two different debiasing methods, Regression-EM and Randomization, and also Click Data without debiasing on the two datasets. Again, we found that Unbiased LambdaMART significantly outperforms the baselines, indicating that Pairwise Debiasing is indeed an effective method.

Figure 3 shows the results of the methods in terms of NDCG@1 and MAP, where we choose NDCG@1 as representative of NDCG scores. For Unbiased LambdaMART, it shows the results under different hyper-parameter values. We can see that Unbiased LambdaMART is superior to all the three baselines on both datasets generated by Position Based Model and Cascade Model. We can also see that in general to achieve high performance the value of p in L_p regularization should not be so high. For the dataset generated by Cascade Model, the performance with L_1 regularization is better than that with L_0 regularization. It indicates that when the data violates its assumption, Unbiased LambdaMART can still learn a reliable model with a higher order of regularization.

5.2.3 Robustness of Unbiased LambdaMART. We further evaluated the robustness of Unbiased LambdaMART under different degrees of position bias.

In the above experiments, we only tested the performance of Unbiased LambdaMART with click data generated from a single click model, i.e., θ as 1 for Position Based Model and β as 0.5 for Cascade Model. Therefore, here we set the two hyper-parameters to different values and examined whether Unbiased LambdaMART can still work equally well.

Figure 4 shows the results in terms of NDCG@1 with different degrees of position bias. The results in terms of other measures have similar trends. When θ in Position Based Model equals 0, and β in Cascade Model equals 1, there is no position bias. The results of all debiasing methods are similar to that of using click data only. As we add more position bias, i.e., θ increases and β decreases, the



(a) Performance on click data generated by Cascade Model

(b) Performance on click data generated by Position Based Model

Figure 3: Performances of LambdaMART versus regularization norms by different debiasing methods, when click data is generated by two different click models.

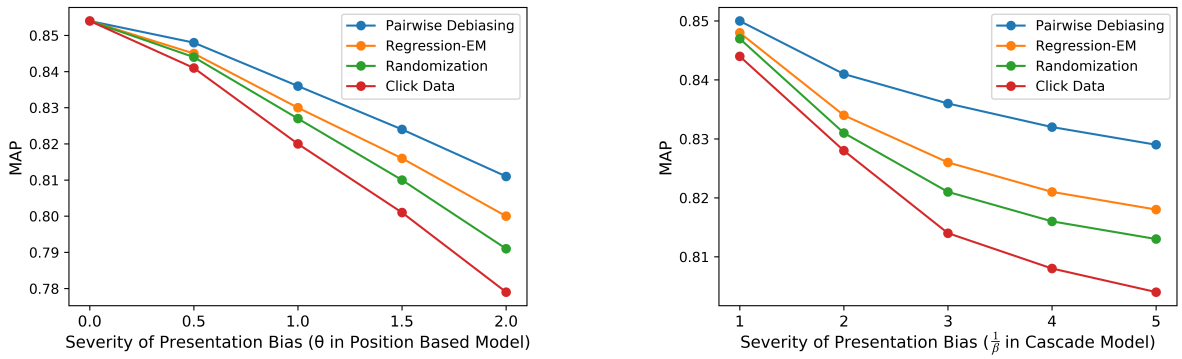


Figure 4: Performances of Pairwise Debiasing against other debiasing methods with different degrees of position bias.

performances of all the debiasing methods decrease dramatically. However, under all settings Unbiased LambdaMART can get less affected by position bias and consistently maintain the best results. This indicates that Unbiased LambdaMART is robust to different degrees of position bias.

Next, we investigate the robustness of Unbiased LambdaMart under different sizes of training data. We randomly selected a subset of training data, (i.e., 20% - 100%) to generate different sizes of click datasets, and used the datasets to evaluate the performances of LambdaMART with different debiasing methods. To make fair comparison, we used the same subsets of training data for running of the Randomization and Regression-EM algorithm.

As shown in Figure 5, when the size of training data decreases, the improvements obtained by the debiasing methods also decrease.

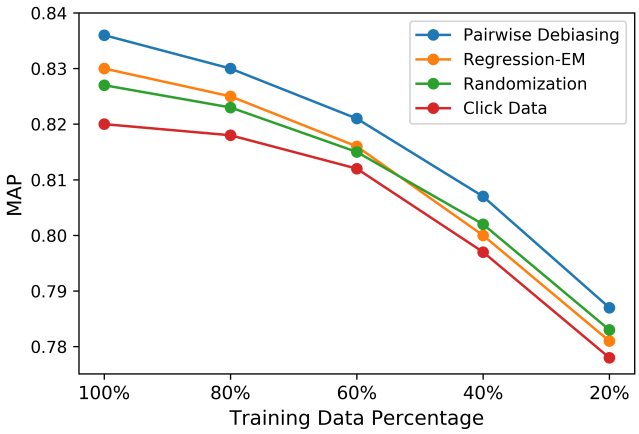


Figure 5: Performances of Pairwise Debiasing against other debiasing methods with different sizes of training data.

The reason seems to be that the position bias estimated from insufficient training data is not accurate, which can hurt the performances of the debiasing methods. In contrast, Unbiased LambdaMART, which adopts a joint training mechanism, can still achieve the best performances in such cases. When the data size increases from 80% to 100%, the performance enhancement of LambdaMART + Click Data is quite small, while the performance enhancements of the debiasing methods are much larger. This result is in accordance with the observation reported in[15], that simply increasing the amount of biased training data cannot help build a reliable ranking model, but after debiasing it is possible to learn a better ranker with more training data. The experiment shows that Unbiased LambdaMART can still work well even with limited training data, and it can consistently increase its performances as training data increases.

5.3 A/B Testing at Commercial Search Engine

We further evaluated the performance of Unbiased LambdaMART by deploying it at the search engine of Jinri Toutiao, a commercial news recommendation app in China with over 100 million daily active users. We trained two rankers with Unbiased LambdaMART and LambdaMART + Click Data using click data of approximately 19.6 million query sessions collected over two days at the search engine. Then we deployed the two rankers at the search system to conduct A/B testing. The A/B testing was carried out for 16 days. In each experiment group, the ranker was randomly assigned approximately 1.5 million queries per day.

In the online environment, we observed that different users have quite different click behaviors. It appeared to be necessary to have a tighter control on debiasing. We therefore set the hyper-parameter p as 1, i.e., we conducted L_1 regularization to impose a stronger regularization on the position biases. We validated the correctness of this hyper-parameter selection on a small set of relevance dataset.

We compared the results of the two rankers in terms of first click ratios, which are the percentages of sessions having first clicks at top 1,3,5 positions among all sessions. A ranker with higher first click ratios should have better performance.

Table 3: Relative increases of first click ratios by Unbiased LambdaMART in online A/B testing.

Measure	Click@1	Click@3	Click@5
Increase	2.64%	1.21%	0.80%
P-value	0.001	0.004	0.023

Table 4: Human assessors' evaluation on results of same queries ranked at top five positions by the two rankers.

Unbiased LambdaMART	Win	Same	Loss
vs. LambdaMart + Click	21	68	11

As shown in Table 3, Unbiased LambdaMART can significantly outperform LambdaMART + Click Data in terms of first click ratios at the A/B Testing. It increases the first click ratios at positions 1,3,5 by 2.64%, 1.21% and 0.80%, respectively, which are all statistically significant (p-values < 0.05). It indicates that Unbiased LambdaMART can make significantly better relevance ranking with its debiasing capability.

We next asked human assessors to evaluate the results of the two rankers. We collected all the different results of the same queries given by the two rankers during the A/B testing period, presented the results to the assessors randomly side-by-side, and asked assessors to judge which results are better. They categorized the results at the top five positions of 100 randomly chosen queries into three categories, i.e., 'Win', 'Same' and 'Loss'.

As shown in Table 4, the win/loss ratio of Unbiased LambdaMart over LambdaMart + Click Data is as high as 1.91, indicating that Unbiased LambdaMART is indeed effective as an unbiased learning-to-rank algorithm.

6 CONCLUSION

In this paper, we have proposed a general framework for pairwise unbiased learning-to-rank, including the extended inverse propensity weighting (IPW) principle. We have also proposed a method called Pairwise Debiasing to jointly estimate position biases and train a ranker by directly optimizing a same objective function within the framework. We develop a new algorithm called Unbiased LambdaMART as application of the method. Experimental results show that Unbiased LambdaMART achieves significantly better results than the existing methods on a benchmark dataset, and is effective in relevance ranking at a real-world search system.

There are several items to work on in the future. We plan to apply Pairwise Debiasing to other pairwise learning-to-rank algorithms. We also consider developing a more general debiasing method that can deal with not only position bias but also other types of bias such as presentation bias. More theoretical analysis on unbiased pairwise learning-to-rank is also necessary.

REFERENCES

[1] Ai, Q., Bi, K., Luo, C., Guo, J., and Croft, W. B. Unbiased learning to rank with unbiased propensity estimation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018* (2018), pp. 385–394.

[2] Ai, Q., Mao, J., Liu, Y., and Croft, W. B. Unbiased learning to rank: Theory and practice. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018* (2018), ACM, pp. 2305–2306.

[3] Borisov, A., Markov, I., de Rijke, M., and Serdyukov, P. A neural click model for web search. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016* (2016), pp. 531–541.

[4] Burges, C. J. From ranknet to lambdarank to lambdamart: An overview. Tech. rep., June 2010.

[5] Burges, C. J. C., Ragno, R., and Le, Q. V. Learning to rank with nonsmooth cost functions. In *Proceedings of the 20th Annual Conference on Neural Information Processing Systems, NIPS 2006, Vancouver, British Columbia, Canada, December 4-7, 2006* (2006), pp. 193–200.

[6] Cao, Y., Xu, J., Liu, T.-Y., Li, H., Huang, Y., and Hon, H.-W. Adapting ranking svm to document retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2006* (New York, NY, USA, 2006), ACM, pp. 186–193.

[7] Chapelle, O., and Chang, Y. Yahoo! learning to rank challenge overview. In *Proceedings of the Yahoo! Learning to Rank Challenge, held at ICML 2010, Haifa, Israel, June 25, 2010* (2011), pp. 1–24.

[8] Chapelle, O., and Zhang, Y. A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009* (2009), pp. 1–10.

[9] Craswell, N., Zoeter, O., Taylor, M. J., and Ramsey, B. An experimental comparison of click position-bias models. In *Proceedings of the International Conference on Web Search and Web Data Mining, WSDM 2008, Palo Alto, California, USA, February 11-12, 2008* (2008), pp. 87–94.

[10] Dupret, G., and Piwowarski, B. A user browsing model to predict search engine click data from past observations. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008* (2008), pp. 331–338.

[11] Friedman, J. H. Greedy function approximation: A gradient boosting machine. *Annals of Statistics 29* (2000), 1189–1232.

[12] Järvelin, K., and Kekäläinen, J. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst. 20*, 4 (2002), 422–446.

[13] Joachims, T. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada* (2002), pp. 133–142.

[14] Joachims, T., Granka, L. A., Pan, B., Hembrooke, H., and Gay, G. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005* (2005), pp. 154–161.

[15] Joachims, T., Swaminathan, A., and Schnabel, T. Unbiased learning-to-rank with biased feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017* (2017), pp. 781–789.

[16] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, NIPS 2017, 4-9 December 2017, Long Beach, CA, USA* (2017), pp. 3149–3157.

[17] Kveton, B., Szepesvari, C., Wen, Z., and Ashkan, A. Cascading bandits: Learning to rank in the cascade model. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015* (2015), pp. 767–776.

[18] Li, H. A short introduction to learning to rank. *IEICE Transactions 94-D*, 10 (2011), 1854–1862.

[19] Li, H. *Learning to Rank for Information Retrieval and Natural Language Processing, Second Edition*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2014.

[20] Li, S., Abbasi-Yadkori, Y., Kveton, B., Muthukrishnan, S., Vinay, V., and Wen, Z. Offline evaluation of ranking policies with click models. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD* (2018), ACM, pp. 1685–1694.

[21] Liu, T. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval 3*, 3 (2009), 225–331.

[22] Richardson, M., Dominowska, E., and Ragno, R. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007* (2007), pp. 521–530.

[23] Rosenbaum, P. R., and Rubin, D. B. The central role of the propensity score in observational studies for causal effects. *Biometrika 70* (1983), 41–55.

[24] Wang, X., Bendersky, M., Metzler, D., and Najork, M. Learning to rank with selection bias in personal search. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016* (2016), pp. 115–124.

[25] Wang, X., Golbandi, N., Bendersky, M., Metzler, D., and Najork, M. Position bias estimation for unbiased learning to rank in personal search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018* (2018), pp. 610–618.

[26] Wu, Q., Burges, C. J. C., Svore, K. M., and Gao, J. Adapting boosting for information retrieval measures. *Inf. Retr. 13*, 3 (2010), 254–270.

[27] Yue, Y., Patel, R., and Roehrig, H. Beyond position bias: examining result attractiveness as a source of presentation bias in clickthrough data. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010* (2010), pp. 1011–1018.