Unbiased Learning to Rank with Unbiased Propensity Estimation

Qingyao Ai
CICS, UMass Amherst
Amherst, MA, USA
aiqy@cs.umass.edu

Keping Bi
CICS, UMass Amherst
Amherst, MA, USA
kbi@cs.umass.edu

Cheng Luo
DCST, Tsinghua University
Beijing, China
luochengleo@gmail.com

Jiafeng Guo
ICT, Chinese Academy of Sciences
Beijing, China
guojiafeng@ict.ac.cn

W. Bruce Croft
CICS, UMass Amherst
Amherst, MA, USA
croft@cs.umass.edu

ABSTRACT

Learning to rank with biased click data is a well-known challenge. A variety of methods has been explored to debias click data for learning to rank such as click models, result interleaving and, more recently, the unbiased learning-to-rank framework based on inverse propensity weighting. Despite their differences, most existing studies separate the estimation of click bias (namely the *propensity model*) from the learning of ranking algorithms. To estimate click propensities, they either conduct online result randomization, which can negatively affect the user experience, or offline parameter estimation, which has special requirements for click data and is optimized for objectives (e.g. click likelihood) that are not directly related to the ranking performance of the system. In this work, we address those problems by unifying the learning of propensity models and ranking models. We find that the problem of estimating a propensity model from click data is a dual problem of unbiased learning to rank. Based on this observation, we propose a Dual Learning Algorithm (DLA) that jointly learns an unbiased ranker and an *unbiased propensity model*. DLA is an automatic unbiased learning-to-rank framework as it directly learns unbiased ranking models from biased click data without any preprocessing. It can adapt to the change of bias distributions and is applicable to online learning. Our empirical experiments with synthetic and real-world data show that the models trained with DLA significantly outperformed the unbiased learning-to-rank algorithms based on result randomization and the models trained with relevance signals extracted by click models.

KEYWORDS

Learning to rank, propensity estimation, inverse propensity weighting

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA

ACM ISBN 978-1-4503-5657-2/18/07…$15.00
https://doi.org/10.1145/3209978.3209986

ACM Reference Format:
Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W. Bruce Croft. 2018. Unbiased Learning to Rank with Unbiased Propensity Estimation. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209978.3209986

1 INTRODUCTION

Machine learning techniques for Information Retrieval (IR) become widely used in both academic research and commercial search engines [19]. Although there have been studies that use unsupervised data or pseudo supervision for learning-to-rank models [2, 8], the best retrieval system is typically constructed based on supervised learning. Many of the state-of-the-art retrieval systems today make use of deep models [11, 22], which require large amounts of labeled data. Despite the development of crowdsourcing systems [9, 18], obtaining large-scale and high quality human annotations (e.g. TREC-style relevance judgments) is still expensive, if not impossible. Therefore, implicit feedback such as clicks are still the most attractive data source for the training of ranking systems.

Directly training a ranking model to optimize click data, however, is infeasible because click data are heavily biased [14, 15, 17, 38]. In particular, the order of documents in a search engine result page (SERP) has a strong influence on where users click [14]. Studies of position bias show that users tend to examine and click results on the top of a SERP while ignoring those on the bottom. A naive method that treats click/non-click signals as positive/negative feedback will lead to a ranking model that optimizes the order of a search result page but not the relevance of documents.

To leverage the full power of click data for learning to rank, IR researchers have attempted to debias click data before training ranking models. One such effort is the development of click models. The basic idea of click model is to make hypotheses about user browsing behaviors and estimate true relevance feedback by optimizing the likelihood of the observed user clicks. Such methods work well on head queries in Web search but not on tail queries or other retrieval applications where multiple observations of same query-document pairs may not be available (e.g. personal search [34]). Also, the construction of click models is separated from the learning of ranking models. Click models are usually optimized for the likelihood of observed clicks but not the ranking performance of the overall

system. Their parameters need to be re-estimated whenever there are changes in user behaviors.

Another effort to debias click data is result interleaving [4, 25, 28, 30, 31, 37]. By collecting clicks on swapped results from the same result list, we can obtain unbiased pair preferences for documents and use them to train learning-to-rank models in an online manner. This paradigm, however, introduces non-deterministic ranking functions into the product system [16]. It may hurt the user experience by putting more irrelevant documents on the top of SERPs.

Based on these problems, a new research direction emerged recently that focuses on directly training ranking models with biased click data, which is often referred to as *unbiased learning to rank* [16, 34]. The unbiased learning-to-rank framework treats click bias as a counterfactual effect and debiases user feedback by weighting each click with their Inverse Propensity Weights [27]. It uses a propensity model to quantify click biases and does not explicitly estimate the query-document relevance with training data. As theoretically proven by Joachims et al. [16], given the correct bias estimation, ranking models trained with click data under this framework will converge to the same model trained with true relevance signals.

Despite their advantages, existing unbiased learning-to-rank algorithms share a common drawback with click models as they need a separate experiment to estimate click bias. One of the most popular methods for click bias estimation is result randomization [16, 34], which randomizes the order of documents so that the collected user clicks on randomized SERPs can reflect the examination bias of users on each result position. This paradigm is similar to result interleaving as they both can negatively affect user experience. Additionally, because result randomization needs to be conducted on a proportion of search engine traffic separately, existing unbiased learning-to-rank models cannot adapt to changes in user behavior automatically.

In this paper, we introduce a new framework for automatic unbiased learning to rank. Most limitations of existing unbiased learning-to-rank models are caused by their additional user experiments for propensity estimation. As Wang et al. [34] and Joachims et al. [16] observed that unbiased rankers can be directly learned from user clicks with the help of propensity models, we observed that click propensity can be automatically estimated with click data given an unbiased ranking model. We formulate the problem of automatically estimating a propensity model from user clicks as *unbiased propensity estimation* and propose a Dual Learning Algorithm (DLA) for unbiased learning to rank. DLA jointly learns propensity models and ranking models based on raw click data. Since it doesn't rely on any result randomization or offline experiments, DLA should be preferable in production systems and applicable to online learning to rank. Furthermore, we theoretically prove that models trained with DLA will converge to their global optima under certain circumstances. To evaluate the effectiveness of DLA in practice, we conducted both simulation and real-world experiments. Empirical experimental results show that models trained with DLA are adaptive to changes in user behavior and significantly outperformed the models trained with click model signals and existing unbiased learning-to-rank frameworks.

The contributions of this paper are summarized as follows:

- We formulate a problem of *unbiased propensity estimation* and discuss its relationship with unbiased learning to rank.
- We propose a Dual Learning Algorithm that automatically and jointly learns unbiased propensity models and ranking models from raw click data.
- We conduct both theoretical analysis and empirical experiments to understand the effect of the joint learning process used by DLA.

The rest of the paper is organized as follows. In Section 2, we review previous work on learning to rank with click data. We introduce existing unbiased learning-to-rank frameworks and the Dual Learning Algorithm in Section 3&4. Our experiment setup and results are described in Section 5&6. Finally, we conclude this paper and discuss future work in Section 7.

2 RELATED WORK

There are two groups of approaches to handle biased user feedback for learning to rank. The first group focuses on debiasing user clicks and extracting reliable relevance signals. The second group tries to directly learn unbiased rankers from biased feedback.

Debias User Feedback from Click Data. As shown by prior work [14, 15, 17, 32, 38], implicit user feedback from click data is severely biased. For the robustness of learning-to-rank models trained with click data, IR researchers have conducted extensive studies on user browsing behaviors and constructed click models [6, 7, 10, 32, 33, 36] to extract real relevance feedback from click signals. For example, Craswell et al. [7] designed a Cascade model to separate click bias from relevance signals by assuming that users read search result pages sequentially from top to bottom. Dupret and Piwowarski [10] proposed a User Browsing Model (UBM) that allows users to skip some results by computing the examination probability of documents according to their positions and last clicks. To further incorporate search abandon behaviors, Chapelle and Zhang [6] constructed a Dynamic Bayesian Network model (DBN) that uses a separate variable to model whether a user is satisfied by a click and ends the search session. In spite of their underlying hypotheses, the goal of click models is to estimate the "true" relevance feedback and use them for learning to rank. Most click models need to be constructed offline and require each query-document pair to appear multiple times for reliable relevance estimation. In contrast, we propose to estimate click bias automatically and jointly with the learning of ranking systems so that our model can be applied to online learning and retrieval applications where multiple appearances of query-document pairs are not available (e.g. email search).

To avoid the modeling of user behaviors, another direction of research tries to collect unbiased relevance feedback directly from users. For instance, it has been shown that presenting randomized ranked lists and collecting user clicks accordingly can reveal the true preferences between swapped documents [4, 25, 28, 30, 31, 37]. Yue and Joachims [37] used result interleaving to collect reliable user feedback and used Dual Bandit Gradient Descent (DBCG) to learn ranking models in an online manner. Schuth et al. [28] extended DBCG and proposed Multileave Gradient Descent to compare multiple rankings and find good rankers. The effectiveness of these learning paradigms has been justified in theory [37], but they

Table 1: A summary of notations used in this paper.

Q, Q, q	The universal set of queries Q, a sample set Q and a query instance $q \sim P(q)$.
S, θ, E, ϕ	A ranking system S parameterized by θ and a propensity model E parameterized by ϕ.
π_q, x, i, y	A ranked list π_q produced by S for q, a document x on the ith position in π_q and its relevance y.
$\mathbf{o}_q, \mathbf{r}_q, \mathbf{c}_q$	The sets of Bernoulli variables that represent whether a document x in q is observed (o_q^x), perceived as relevant (r_q^x) and clicked (c_q^x).

are not popular in practice because result interleaving hurts ranking quality and introduces non-deterministic factors into search engines [16]. The approach proposed in our work is also applicable to online learning to rank but it learns rankers from real user clicks without any result randomization.

Unbiased Learning to Rank. Based on the intrinsic limitations of click models, IR researchers have explored a new approach to account for click bias for learning to rank. Instead of inferring relevance signals by optimizing observed click likelihood [6, 7, 10, 32, 33, 36], the examination propensity of each document can be estimated through result randomization and used to construct unbiased ranking loss for learning to rank. For example, Wang et al. [34] proposed estimating the selection bias at query level through a randomization experiment and used Inverse Propensity Weighting (IPW) [27] to debias the training loss computed with click signals. Joachims et al. [16] analyzed the IPW framework for unbiased learning to rank and showed that it can find the unbiased ranker theoretically and empirically. The framework of existing unbiased learning-to-rank algorithms doesn't require multiple observations for each query-document pair. In contrast to online learning algorithms with result interleaving, it constructs a deterministic ranking model [16]. Nonetheless, most existing unbiased learning-to-rank algorithms still rely on a separate result randominzation experiment to estimate the propensity model for IPW. They are not immune to the problems resulting from result randomization. Wang et al. [35] proposed a novel EM algorithm to estimate click propensity without result randomization, but models trained with their or other unbiased learning-to-rank framework are not adaptive. The EM process and the result randomization have to be conducted every time when there is any change in search engine interfaces or user behaviors. Similar to previous studies [16, 34, 35], we adopt the IPW framework for unbiased learning to rank. However, we discard result randomization and automate the entire unbiased learning-to-rank framework so that propensity models and unbiased rankers can be jointly learned with raw user clicks.

3 UNBIASED LEARNING TO RANK

In this section, we discuss the existing unbiased learning-to-rank framework. The core of unbiased learning to rank is to debias loss functions built with user clicks so that the ranking model converges to the model trained with true relevance labels.

A summary of notations used in this paper is shown in Table 1. Without the loss of generality, we describe learning to rank with true relevance information as follows. Let Q be the universal set of all possible queries and q be an instance from Q which follows the distribution of $q \sim P(q)$. Suppose that we have a ranking system S and a loss function l defined over S and q, then the global loss $\mathcal{L}$ of S is defined as

$$\mathcal{L}(S) = \int_{q \in Q} l(S, q)\, dP(q)$$

The goal of learning to rank is to find the best ranking system S that minimizes $\mathcal{L}(S)$. Because $\mathcal{L}(S)$ cannot be computed directly, we often estimate it empirically based on a separate training query set Q and the uniform assumption on $P(q)$:

$$\hat{\mathcal{L}}(S) = \frac{1}{|Q|} \sum_{q \in Q} l(S, q)$$

Usually, $l(S, q)$ is defined over the order of documents and their relevance with the query. Let π_q be the ranked list retrieved by S for query q, x be a document in π_q and y be the binary label that denotes whether x is relevant to q. In most cases, we are only concerned with the position of relevant documents ($y = 1$) in retrieval evaluations (e.g. MAP, nDCG [12], ERR [5]), so we can formulate the local ranking loss $l(S, q)$ as:

$$l(S, q) = \sum_{x \in \pi_q, y=1} \Delta(x, y|\pi_q) \tag{1}$$

where $\Delta(x, y|\pi_q)$ is a function that computes the individual loss on each relevant document in π_q.

The relevance labels y are typically elicited either explicitly through expert judgments or implicitly via user feedback. The former is often considered to be more reliable, but it is expensive or impossible to obtain in many retrieval scenarios (e.g. personal search). Also, it generates relevance judgments based on the aggregation of all intents that underlie the same query string with the distributions estimated by judges but not real users. The latter, which refers to clicks collected from real users, are cheap yet heavily biased. It is affected by multiple factors including presentation bias [38], trust bias [15, 17] and, most commonly, position bias [14]. To utilize the relevance information hidden in user clicks, we must debias the click signals before applying it to learning-to-rank frameworks. One of the standard methods for bias correction is the inverse propensity weighting algorithm [16, 34].

3.1 Inverse Propensity Weighting

Inverse propensity weighting (IPW) is first proposed for unbiased learning to rank by Wang et al. [34] and Joachims et al. [16]. It introduces a counterfactual model that removes the effect of click bias. Let $\mathbf{o}_q, \mathbf{c}_q$ be the sets of Bernoulli variables that represent whether the documents in π_q are observed and clicked by a user. For simplicity, we assume that x will be clicked ($c_q^x = 1$) when it is observed ($o_q^x = 1$) and relevant ($y = 1$). The main idea of IPW is to optimize ranking systems S with an inverse propensity weighted loss defined as

$$l_{IPW}(S, q) = \sum_{x \in \pi_q} \Delta_{IPW}(x, y|\pi_q) = \sum_{x \in \pi_q, o_q^x = 1, y=1} \frac{\Delta(x, y|\pi_q)}{P(o_q^x = 1|\pi_q)} \tag{2}$$

There are two important properties of the inverse propensity weighted loss. First, $\Delta_{IPW}(x, y|\pi_q)$ is computed only when x is both observed and relevant, so we can ignore non-clicked documents in $l_{IPW}(S, q)$. This is essential because we do not know the reason that causes $c_q^x = 0$ (either $o_q^x = 0$ or $y = 0$ or both). Second, the IPW loss is theoretically principled because it is an unbiased estimation

of $l(S,q)$. As shown by Joachims et al. [16], the expectation of the inverse propensity weighted loss is

$$\begin{aligned}\mathbb{E}_{\mathbf{o}_q}[l_{IPW}(S,q)] &= \mathbb{E}_{\mathbf{o}_q}\Big[\sum_{x\in\pi_q, o_q^x=1, y=1}\frac{\Delta(x,y|\pi_q)}{P(o_q^x=1|\pi_q)}\Big]\\ &= \mathbb{E}_{\mathbf{o}_q}\Big[\sum_{x\in\pi_q, y=1}\frac{o_q^x\cdot\Delta(x,y|\pi_q)}{P(o_q^x=1|\pi_q)}\Big]\\ &= \sum_{x\in\pi_q, y=1}\mathbb{E}_{\mathbf{o}_q}\Big[o_q^x\Big]\cdot\frac{\Delta(x,y|\pi_q)}{P(o_q^x=1|\pi_q)}\\ &= \sum_{x\in\pi_q, y=1}P(o_q^x=1|\pi_q)\cdot\frac{\Delta(x,y|\pi_q)}{P(o_q^x=1|\pi_q)}\\ &= \sum_{x\in\pi_q, y=1}\Delta(x,y|\pi_q) = l(S,q)\end{aligned} \tag{3}$$

The ranking model trained with clicks and the IPW loss will converge to the same model trained with true relevance labels. Thus, the whole learning-to-rank framework is unbiased.

3.2 Randomization-based Estimation

The key of the IPW algorithm is the estimation of propensity model $P(o_q^x=1|\pi_q)$. Although the framework can be easily extended to other biases [16], most existing work on unbiased learning to rank only focuses on the effect of position bias [14] for simplicity. This work assumes that $P(o_q^x=1|\pi_q)$ only depends on the position of x:

$$P(o_q^x=1|\pi_q) = P(o_i=1)$$

where i is the position of x in the ranked list π_q.

A simple yet effective solution to estimate a position-based propensity model is result randomization [14, 34]. The idea of result randomization is to shuffle the order of documents and collect user clicks on different positions to compute the propensity model. Because the expected document relevance is the same on all positions, it is easy to prove that result randomization method produces an optimal estimator for position-based propensity model:

$$\begin{aligned}\mathbb{E}[c_k] &= \int_{(q,x,\pi_q), i=k}P(c_q^x=1|\pi_q)\,dP(q,x,\pi_q)\\ &= \int_{(q,x,\pi_q), i=k}P(o_i=1)\cdot y\,dP(q,x,\pi_q)\\ &= P(o_k=1)\cdot\int_{(q,x,\pi_q), i=k}y\,dP(q,x,\pi_q)\\ &\propto P(o_k=1)\end{aligned} \tag{4}$$

Despite its simplicity and effectiveness, result randomization has intrinsic drawbacks that limit its applications. First, it can significantly affect the user experience. Shuffling documents in the ranked list will inevitably put more irrelevant results in high positions. Previous studies have explored several strategies (e.g. pair-based randomization [14]) to alleviate this problem, but none of them can solve it completely. Second, the use of result randomization makes existing unbiased learning-to-rank framework less efficient and adaptive. Randomization experiments is time-consuming and has to be conducted separately with the training of ranking models. It is painful to re-train the whole system and thus difficult to keep the model updated with changes in user behaviors. As far as we know, most existing unbiased learning-to-rank algorithms rely on result randomization to estimate propensity model, which makes them vulnerable to the two problems discussed above. To solve them, we discard randomization experiments completely and propose to automatically learn both the ranking model and the propensity model based on user clicks.

4 OUR APPROACH

We now describe our approach for automatic unbiased learning to rank. The key component that limits existing unbiased learning-to-rank algorithms to be fully automatic is the estimation of click propensity. In this work, we find that estimating a propensity model with user clicks is actually a dual problem of unbiased learning to rank, which could be unbiasedly learned in a similar way. Thus, we refer to it as *unbiased propensity estimation* and propose a Dual Learning Algorithm that jointly learns the unbiased ranking model and the propensity model with click data. Theoretical analysis shows that our approach is guaranteed to find the unbiased ranker and propensity model under certain circumstances.

4.1 Unbiased Propensity Estimation

Let $\mathbf{r}_q$ be a set of Bernoulli variables which denote whether the documents in π_q will be perceived as relevant when users observe them. Then the probability that a document $x\in\pi_q$ will be clicked can be computed as

$$P(c_q^x=1|\pi_q) = P(o_q^x=1|\pi_q)\cdot P(r_q^x=1|\pi_q)$$

Because users click a search result ($c_q^x=1$) only when it is both observed ($o_q^x=1$) and perceived as relevant ($r_q^x=1$), we cannot directly infer the relevance of a document without knowing whether it has been examined. Similarly, we cannot estimate the propensity of examination without knowing whether the documents are relevant or not. From this point of view, the problem of estimating examination propensity is symmetric with the problem of estimating real document relevance from user clicks. Since the latter is referred to as *unbiased learning to rank*, we formulate the former problem as *unbiased propensity estimation*.

We now formally describe the problem of unbiased propensity estimation. Similar to learning to rank, the goal of propensity estimation is to find the optimal propensity model E that minimizes a global loss function:

$$\mathcal{L}(E) = \int_{q\in Q}l(E,q)\,dP(q)$$

where $l(E,q)$ is a function that computes the local loss of E in query q. Suppose that we only care about the performance of propensity estimation on documents that have been observed by users, then $l(E,q)$ can be defined as

$$l(E,q) = \sum_{x\in\pi_q, o_q^x=1}\Delta(x,o_q^x|\pi_q) \tag{5}$$

Under this formulation, it is obvious that the learning of a propensity model is similar to the learning of a ranking function. Thus, the inverse propensity weighting algorithm for unbiased learning to rank can also be directly applied to unbiased propensity estimation.

Algorithm 1: Dual Learning Algorithm

Input: $Q = \{q, \pi_q, \mathbf{c}_q\}$, f, g, learning rate α
Output: θ, ϕ
Initialize $\theta \Leftarrow 0$, $\phi \Leftarrow 0$
repeat
  Randomly sample a batch Q' from Q
  for $(q, \pi_q, \mathbf{c}_q) \in Q'$ **do**
    for $x \in \pi_q$ **do**
      Compute $P(r_q^x = 1|\pi_q)$, $P(o_q^x = 1|\pi_q)$ with Eq 9.
    end
  end
  Compute $\hat{L}(S, q)$, $\hat{L}(E, q)$ with Eq 11.
  $\theta = \theta + \alpha \cdot \frac{\partial \hat{L}(S,q)}{\partial \theta}$, $\phi = \phi + \alpha \cdot \frac{\partial \hat{L}(E,q)}{\partial \phi}$
until *Convergence*;
return θ, ϕ

Similar to Equation (2), we define the *Inverse Relevance Weighted* (IRW) loss for E as

$$l_{IRW}(E,q) = \sum_{x \in \pi_q} \Delta_{IRW}(x, o_q^x|\pi_q) = \sum_{x \in \pi_q, o_q^x=1, r_q^x=1} \frac{\Delta(x, o_q^x|\pi_q)}{P(r_q^x = 1|\pi_q)} \quad (6)$$

Following the same logic flow in Equation (3), we can easily prove that this is an unbiased estimate of $l(E, q)$ because:

$$\begin{aligned}
\mathbb{E}_{\mathbf{r}_q}[l_{IRW}(E,q)] &= \mathbb{E}_{\mathbf{r}_q}\Big[\sum_{x \in \pi_q, o_q^x=1, r_q^x=1} \frac{\Delta(x, o_q^x|\pi_q)}{P(r_q^x = 1|\pi_q)}\Big] \\
&= \mathbb{E}_{\mathbf{r}_q}\Big[\sum_{x \in \pi_q, o_q^x=1} \frac{r_q^x \cdot \Delta(x, o_q^x|\pi_q)}{P(r_q^x = 1|\pi_q)}\Big] \\
&= \sum_{x \in \pi_q, o_q^x=1} \mathbb{E}_{\mathbf{r}_q}\big[r_q^x\big] \cdot \frac{\Delta(x, o_q^x|\pi_q)}{P(r_q^x = 1|\pi_q)} \\
&= \sum_{x \in \pi_q, o_q^x=1} P(r_q^x = 1|\pi_q) \cdot \frac{\Delta(x, o_q^x|\pi_q)}{P(r_q^x = 1|\pi_q)} \\
&= \sum_{x \in \pi_q, o_q^x=1} \Delta(x, o_q^x|\pi_q) = l(E,q)
\end{aligned} \quad (7)$$

If we compare the problem of unbiased learning to rank with the problem of unbiased propensity estimation, we can see that the goal of the former is to estimate $P(r_q^x = 1|\pi_q)$ while the goal of the latter is to estimate $P(o_q^x = 1|\pi_q)$. This indicates that a good ranking model can help us estimate a good propensity model and vice versa. Based on this observation, we propose a Dual Learning Algorithm to jointly learn the propensity model and the ranking model with click data.

4.2 Dual Learning Algorithm

The idea of the Dual Learning Algorithm (DLA) is to solve the problems of unbiased learning to rank and unbiased propensity estimation simultaneously. It has two important components: the loss function $l(S, q)$, $l(E, q)$ and the estimation of $P(o_q^x = 1|\pi_q)$ and $P(r_q^x = 1|\pi_q)$. Let $g_q^x(\phi)$ and $f_q^x(\theta)$ be the propensity score and ranking score produced by the propensity model E (parameterized by ϕ) and ranking model S (parameterized by θ) for document x in query q. As discussed previously, $l(S, q)$ and $l(E, q)$ only have values on clicked documents and click behavior only happens on documents that are observed and relevant. Thus, pointwise loss functions are likely to fail in the IPW framework because we only use relevant documents to train the ranking model. Inspired by [1], we adapted a list-wise loss based on softmax-based cross entropy for DLA as:

$$\begin{aligned}
l(E,q) &= \sum_{x \in \pi_q, o_q^x=1, r_q^x=1} \Delta(x, o_q^x|\pi_q) = -\sum_{x \in \pi_q, c_q^x=1} \log \frac{e^{g_q^x(\phi)}}{\sum_{z \in \pi_q} e^{g_q^z(\phi)}} \\
l(S,q) &= \sum_{x \in \pi_q, o_q^x=1, r_q^x=1} \Delta(x, r_q^x|\pi_q) = -\sum_{x \in \pi_q, c_q^x=1} \log \frac{e^{f_q^x(\theta)}}{\sum_{z \in \pi_q} e^{f_q^z(\theta)}}
\end{aligned} \quad (8)$$

The softmax-based cross entropy naturally converts the outputs of ranking models and propensity models into probability distributions, which are then used for the propensity and relevance estimation:

$$P_E(o_q^x = 1|\pi_q) = \frac{e^{g_q^x(\phi)}}{\sum_{z \in \pi_q} e^{g_q^z(\phi)}}, \; P_S(r_q^x = 1|\pi_q) = \frac{e^{f_q^x(\theta)}}{\sum_{z \in \pi_q} e^{f_q^z(\theta)}} \quad (9)$$

Note that other loss functions can also be adopted in DLA as long as they follow a similar probability framework in Equation (8)&(9). We leave the investigation of other loss functions for future studies.

As shown in Equation (9), the use of the softmax function assumes that the examination probabilities on different positions in a ranked list will sum up to 1, which is not true in practice. This, however, does not hurt the effectiveness of model training. The predicted values for $P(r_q^x = 1|\pi_q)$ and $P(o_q^x = 1|\pi_q)$ have a minor effect on the unbiased learning process as long as their relative proportions are correct. In fact, the actual inverse propensity weighted loss functions used in the DLA are:

$$\begin{aligned}
\hat{l}_{IRW}(E,q) &= -\sum_{x \in \pi_q, c_q^x=1} \frac{P_S(r_q^1 = 1|\pi_q)}{P_S(r_q^x = 1|\pi_q)} \cdot \log \frac{e^{g_q^x(\phi)}}{\sum_{z \in \pi_q} e^{g_q^z(\phi)}} \\
\hat{l}_{IPW}(S,q) &= -\sum_{x \in \pi_q, c_q^x=1} \frac{P_E(o_q^1 = 1|\pi_q)}{P_E(o_q^x = 1|\pi_q)} \cdot \log \frac{e^{f_q^x(\theta)}}{\sum_{z \in \pi_q} e^{f_q^z(\theta)}}
\end{aligned} \quad (10)$$

where $P(o_q^1 = 1|\pi_q)$ and $P(r_q^1 = 1|\pi_q)$ are the marginal probabilities for the first document in π_q. The expected values of $\hat{l}_{IRW}(E, q)$ and $\hat{l}_{IPW}(S, q)$ are proportional to $l(S, q)$, $l(E, q)$, which doesn't affect the effectiveness of unbiased learning discussed in Equation 3&7. Finally, the empirical loss of S and E can be computed as:

$$\hat{\mathcal{L}}(E) = \frac{1}{|Q|} \sum_{q \in Q} \hat{l}_{IRW}(E,q), \hat{\mathcal{L}}(S) = \frac{1}{|Q|} \sum_{q \in Q} \hat{l}_{IPW}(S,q) \quad (11)$$

To compute the loss on a batch of queries $Q' \subseteq Q$, we can simply replace Q with Q' in Equation (11).

An overview of the complete algorithm is shown in Algorithm 1. In DLA, we first initialize all parameters to zero. Then for each batch of queries, we compute $P(r_q^x = 1|\pi_q)$, $P(o_q^x = 1|\pi_q)$ with Equation (9) and $\hat{L}(S, q)$, $\hat{L}(E, q)$ with Equation (11). We update θ and ϕ with the derivatives of $\hat{L}(S, q)$ and $\hat{L}(E, q)$ respectively and repeat the process until the algorithm converges.

4.3 Convergence Analysis

As discussed in Section 3.1&4.1, we can learn the optimal ranker given the correct propensity model and learn the optimal propensity model given the correct ranker. Now we show that both of them can be achieved with the joint learning process of DLA.

For simplicity, we first consider the cases where θ is fixed for S. Let f_x and g_x be the value of $P_S(r_q^x = 1|\pi_q)$ and $P_E(o_q^x = 1|\pi_q)$ in Equation (9). When θ is fixed, ϕ is the only learnable parameter in DLA. As we only consider position bias in this work, $g_x = g_i$ (i is the position of x in π_q) and $\{g_i\}$ are independent with each other. Suppose that g_i is the softmax of ϕ_i over ϕ where ϕ_i is the ith column of ϕ, then DLA will converge when:

$$\begin{aligned}\frac{\partial \hat{\mathcal{L}}(E)}{\partial \phi_i} &= -\frac{1}{|Q|}\sum_{q\in Q}\frac{\partial \hat{l}_{IPW}(E,q)}{\partial \phi_i} \\ &= \frac{1}{|Q|}\sum_{q\in Q}\sum_{j=1}^{|\pi_q|}\frac{c_q^j f_1}{f_j}g_i - \frac{c_q^i f_1}{f_i} \\ &= g_i\sum_{j=1}^{|\pi_q|}\frac{1}{|Q|}\sum_{q\in Q}\frac{c_q^j f_1}{f_j} - \frac{1}{|Q|}\sum_{q\in Q}\frac{c_q^i f_1}{f_i} \\ &= g_i\sum_{j=1}^{|\pi_q|}\mathbb{E}[\frac{c_q^j f_1}{f_j}] - \mathbb{E}[\frac{c_q^i f_1}{f_i}] = 0\end{aligned} \tag{12}$$

We use the fact that $\frac{\partial g_i}{\partial \phi_j} = \mathbf{1}_{j=i} - g_j$ in step 2 and finally get that $g_i = \mathbb{E}[\frac{c_q^i f_1}{f_i}]/\sum_{j=1}^{|\pi_q|}\mathbb{E}[\frac{c_q^j f_1}{f_j}]$. It worth noticing that $\hat{\mathcal{L}}(E)$ will always converge to its global minimum because it is concave:

$$\frac{\partial^2 \hat{\mathcal{L}}(E)}{\partial \phi_i^2} = (1-g_i)\sum_{j=1}^{|\pi_q|}\mathbb{E}[\frac{c_q^j f_1}{f_j}] \geq 0$$

Therefore, when DLA converges, the inverse propensity weights produced by E on position i is

$$\frac{g_1}{g_i} = \frac{\mathbb{E}[\frac{c_q^1 f_1}{f_1}]}{\mathbb{E}[\frac{c_q^i f_1}{f_i}]} = \frac{\mathbb{E}[c_q^1]}{\mathbb{E}[\frac{c_q^i f_1}{f_i}]} = \frac{\mathbb{E}[o_q^1 \cdot r_q^1]}{\mathbb{E}[\frac{o_q^i \cdot r_q^i \cdot f_1}{f_i}]} = \frac{\mathbb{E}[\frac{r_q^1}{r_q^i}]}{\mathbb{E}[\frac{f_1}{f_i}]}\cdot\frac{\mathbb{E}[o_q^1]}{\mathbb{E}[o_q^i]} \tag{13}$$

We use the fact that $c_q^i = o_q^i \cdot r_q^i$ and $\{o_q^i\}, \{r_q^i\}, \{f_i\}$ are independent given θ.

In Equation (13), $\mathbb{E}[r_q^1/r_q^i], \mathbb{E}[f_1/f_i]$ are the real and estimated inverse relevance weights for $\hat{l}_{IRW}(E,q)$, and $\mathbb{E}[o_q^1]/\mathbb{E}[o_q^i] = P(o^1 = 1)/P(o^i = 1)$ is the true inverse propensity weight we want to estimate for $\hat{l}_{IPW}(S,q)$. This indicates that *the better the S is as an unbiased ranker, the better the E is as an unbiased propensity estimator*. Similarly, we can prove its inverse proposition by fixing ϕ and deriving the derivative of θ with respect to $\hat{\mathcal{L}}(S)$. As shown by McLachlan and Krishnan [21], jointly optimizing two functions that control each other can converge to their global optima when both of them are concave. Because $\hat{\mathcal{L}}(E)$ is concave with respect to ϕ, DLA will converge to the best unbiased ranker S and propensity estimator E when $\hat{\mathcal{L}}(S)$ is also concave with respect to θ.

4.4 Model Implementation

Theoretically, any machine learning model that works with stochastic gradient decent (SGD) can be used to implement the ranker and propensity model in DLA. In this paper, we implement the ranker S in DLA with deep neural networks (DNN). Given a document's feature vector $\boldsymbol{x}$, we have

$$\begin{aligned}\boldsymbol{h_0} &= \boldsymbol{x} \\ \boldsymbol{h_k} &= elu(\boldsymbol{W}_{k-1}\cdot\boldsymbol{h}_{k-1} + \boldsymbol{b}_{k-1}), k = 1,2,3,4\end{aligned} \tag{14}$$

where $\theta = \{\boldsymbol{W}_k, \boldsymbol{b}_k | k = 0,1,2,3\}$ are the parameters learned from training data and $elu(x)$ is a non-linear activation function that equals to x when $x \geq 0$ and $e^x - 1$ otherwise. The output of the network h_4 is a scalar, which will be used as the ranking score $f_q^x(\theta)$ for the document.

As we only consider position bias in this work, the most straightforward method to implement the propensity estimator E is to represent the propensity score for each position with a separate variable. We tried other methods like converting positions into one-hot input vectors and using a DNN or recurrent neural network to predict the propensity scores. However, we observe no benefit from applying these complicated models for E. Thus, we only report the results of DLA that directly represents the propensity score of position i with ϕ_i.

5 EXPERIMENTAL SETUP

To analyze the effectiveness of DLA, we conducted two types of experiments. The first one is a simulation experiment based on a public learning-to-rank dataset. The second one is a real-world experiment based on the actual ranked lists and user clicks collected from a commercial Web search engine.

5.1 Simulation Experiment Setup

To fully explore the spectrum of click biases and the corresponding performance of DLA under different situations, we conducted experiments on the Yahoo! LETOR set 1[1] with derived click data. Yahoo! LETOR data is one of the largest public learning-to-rank dataset from commercial English search engines. In total, it contains 29,921 queries with 710k documents. Each query-document pair has a 5-level relevance judgment and 700 features selected by a separate feature selection step in which the most predictive production features are kept [3]. We follow the same data split of training, valiation and testing in the Yahoo! LETOR set 1. Due to privacy concerns, no user information and click data was released with this dataset. Thus, we need to sample synthetic click data for the training of unbiased learning-to-rank models.

Click simulation. Similar to the setting used by Joachims et al. [16], we generate click data on Yahoo! LETOR dataset with a two-step process. First, we trained a Ranking SVM model [13] using 1% of the training data with real relevance judgments to generate the initial ranked list π_q for each query q. We refer to this model as the *Initial Ranker*. Second, we sampled clicks on documents by simulating the browsing process of search users. We assume that a user will click a search result ($c_q^x = 1$) if and only if the document is observed ($o_q^x = 1$) and perceived as relevant ($r_q^x = 1$). To sample

[1] http://webscope.sandbox.yahoo.com

o_q^x, we adopted the presentation bias $\boldsymbol{\rho}$ estimated by Joachims et al. [14] through eye-tracking experiments:

$$P(o_q^x = 1|\pi_q) = P(o_i = 1) = \rho_i^\eta$$

where $\eta \in [0, +\infty]$ is a hyper-parameter that controls the severity of presentation biases. We set $\eta = 1.0$ if not discussed explicitly. Following the methodology proposed by Chapelle et al. [5], we sampled r_q^x with:

$$P(r_q^x = 1|\pi_q) = \epsilon + (1 - \epsilon)\frac{2^y - 1}{2^{y_{max}} - 1}$$

where $y \in [0, 4]$ is the 5-level relevance label for document x and y_{max} is the maximum value of y (which is 4 in our case). We use a parameter ϵ to model click noise so that irrelevant documents ($y = 0$) have non-zero probability to be perceived as relevant and clicked. Joachims et al. [16] have proved that click noise does not affect the effectiveness of unbiased learning-to-rank with IPW framework as long as $P(r_q^x = 1|\pi_q)$ is higher on relevant documents than irrelevant documents. For simplicity, we fixed the value of ϵ as 0.1.

Baselines. We included two groups of baselines in the simulation experiments. The first group is the ranking models trained with click data directly without any bias correction, which is referred to as *NoCorrect*. The second group is the existing unbiased learning-to-rank algorithms based on randomization experiments [16], which is referred to as *RandList*. We randomly shuffle the results in the initial lists provided by the initial ranker and sampled 2 million click sessions to estimate the examination propensity on each position. For ranking algorithms, we tested the Ranking SVM (which is used by Joachims et al. [16] in their initial study of unbiased learning to rank) and the deep neural network (DNN) described in Section 4.4. In total, we have four baselines in our simulation experiments: the Ranking SVM with NoCorrect/RandList and the DNN with NoCorrect/RandList.

Model training. We trained all models with the training set of Yahoo! LETOR dataset based on synthetic clicks. Click sessions for training queries are sampled on the fly to avoid unnecessary biases introduced by off-line generations. We used the Ranking SVM[2] from Joachims et al. [16] and implemented the DNN model with Tensorflow[3]. We tuned the parameter c from 20 to 200 for the Ranking SVM and tuned the number of hidden units from 128 to 512 for the DNN. We trained the DNN with stochastic gradient descent and tuned the learning rate α from 0.005 to 0.05. We set batch size as 256 and stopped training after 10k steps, which means that each model observed approximately 2.5 million click sessions. In this paper, we only report the best results for each baseline. Our code and synthetic data can be found in the following link[4].

Evaluation. The evaluation of retrieval performance for baselines and DLA are conducted on the test set of Yahoo! LETOR data with expert judged relevance labels. The evaluation metrics we used include the mean average precision (MAP), the normalized Discounted Cumulative Gain (nDCG) [12] and the Expected Reciprocal Rank (ERR) [5]. For both nDCG and ERR, we reported the results at rank 1, 3, 5 and 10 to show the performance of models on different positions. Statistic differences are computed based on the Fisher randomization test [29] with $p \leq 0.05$. We will discuss the results of the simulation experiments in Section 6.1.

[2]https://www.cs.cornell.edu/people/tj/svm_light/svm_proprank.html
[3]https://www.tensorflow.org/
[4]https://github.com/QingyaoAi/Dual-Learning-Algorithm-for-Unbiased-Learning-to-Rank

Table 2: A summary of the ranking features extracted for our real-world experiments.

TF	The average term frequency of query terms in url, title, content and the whole document.
IDF	The average inverse document frequency of query terms in url, title, content and the whole document.
TF-IDF	The average value of $tf \cdot idf$ of query terms in url, title, content and the whole document.
BM25	The scores of BM25 [26] on url, title, content and the whole document.
LMABS	The scores of Language Model (LM) [23] with absolute discounting [39] on url, title, content and the whole document.
LMDIR	The scores of LM with Dirichlet smoothing [39] on url, title, content and the whole document.
LMJM	The scores of LM with Jelinek-Mercer [39] on url, title, content and the whole document.
Length	The length of url, title, content and the whole document.
Slash	The number of slash in url.

5.2 Real-world Experiment Setup

In order to show the effectiveness of DLA for unbiased learning to rank in practice, we collected click data from a commercial Web search engine. We randomly sampled 3,449 queries written by real search engine users and collected the top 10 results from a two-week search log. We downloaded the raw HTML documents based on urls and removed ranked lists which contain documents that cannot be reached by our crawler. After cleaning, we have 333,813 documents, 71,106 ranked lists and 3,268,177 anonymized click sessions in total.

Feature extraction. For the training of learning-to-rank algorithms, we manually extracted features based on the text of queries and documents. Following a similar methodology used by Microsoft Letor data [24], we designed features based on url, title, content and the whole text of the documents. In total, we have 33 features for each query-document pair. The ranking features used in our experiments are summarized in Table 2.

Baselines. Due to the limits of our access to the commercial system, we cannot conduct result randomization experiments on real users. Therefore, we focus on the comparison of DLA with other bias correction methods built on user clicks. More specifically, we compared our approach with the model trained with relevance signals estimated by click models [6, 10]. Click models are designed to extract the true relevance feedback from click data through making hypotheses on user browsing behaviors. In our experiments, we implemented two click models: the user browsing model (UBM) [10] and the dynamic bayesian network model (DBN) [6]. UBM assumes that the examination of a document depends on its position and its distance to the last click. DBN assumes that users will keep reading documents sequentially and click them if they look attractive. If not satisfied, users will have a constant probability to return to the search result page and continue reading. Both UBM and DBN use the Expectation-Maximization algorithm [21] to estimate their parameters based on click logs. To insure the quality of relevance estimation with click models, we removed ranked lists with less

than 10 click sessions in the cleaning process described previously. The final baselines are the DNN models (in Section 4.4) trained with the relevance signals extracted by UBM and DBN. Other training settings are the same as those used in the simulation experiments.

Evaluation. The evaluation of unbiased learning-to-rank algorithms requires human judgments of query-document relevance. In our experiments, we constructed a separate test dataset with 100 queries and recruited professional assessors to judge the relevance of top 100 documents retrieved by BM25 [26] in five level (*irrelevant*, *fair*, *good*, *excellent* and *perfect*). We trained our models and baselines on the training set with clicks and evaluated their performance on the test set with human annotations. Similar to the simulation experiments, we reported the value of MAP, nDCG and ERR for all models in Section 6.2. Please refer to [40] for the data used in this paper.

6 RESULTS AND ANALYSIS

In this section, we discuss the results of our simulation experiments and real-world experiments. In particular, we focus on the following research questions:

- **RQ1**: Can DLA effectively estimate the true presentation bias and produce an unbiased ranker at the same time?
- **RQ2**: Compared to the methodology that debiases click data and trains learning-to-rank models separately, are there any benefits from the joint learning of rankers and propensity models empirically?

6.1 Comparison with Result Randomization

To answer RQ1, we compare DLA with the unbiased learning-to-rank algorithms built on result randomization in the simulation experiments. Specifically, we consider two scenarios. In the first scenario, we generated the synthetic clicks with a single bias model $\boldsymbol{\rho}$ in result randomization and model training. In the second scenario, we fixed $\boldsymbol{\rho}$ in result randomization but disturb its severity parameter η in model training. We refer to the first scenario as the *Oracle Mode* and the second scenario as the *Realistic Mode*.

Oracle Mode. The motivation of Oracle Mode is to test unbiased learning-to-rank algorithms in cases where click bias does not change over time. The performance of ranking models trained with different bias correction methods in this scenario is summarized in Table 3. For better illustration, we also include the results of the initial ranked lists (Initial Ranker) and the DNN model trained with human annotations (Oracle DNN).

As shown in Table 3, feedback information from click data indeed helped improve the performance of ranking models. Even with no bias correction, the performance of Ranking SVM and DNN trained with click data are better than the Initial Ranker. Comparing the two ranking algorithms, the DNN with softmax cross entropy consistently outperformed Ranking SVM. After incorporating bias corrections with the propensity model estimated by result randomization (RandList), we observed approximately 3% improvements with respect to ERR@10 on Ranking SVM and DNN over the models trained with raw clicks. This demonstrated the effectiveness of unbiased learning to rank with inverse propensity weighting and RandList. In Oracle Mode, we manually ensured that the presentation bias in the randomization experiments is the same as those in the training data. As discussed in Section 3.2, result randomization is guaranteed to find the true click propensity in theory [16]. Therefore, the ranking models trained with RandList can be treated as the optimal ranker we can get with unbiased learning-to-rank algorithms.

Comparing the DNN models trained with DLA and RandList, we find that DLA is as effective as (if not better than) RandList in terms of bias correction. The DNN with DLA performed similarly with the DNN with RandList and was significantly better than other baselines. Its performance is close to the Oracle DNN, which is trained with human relevance judgments. Because the DNN with DLA does not use result randomization and performed as effectively as the model trained with RandList, it has advantages in real retrieval applications.

Realistic Mode. The assumption of Oracle Mode that click biases remain unchanged in randomization experiments and model training is not realistic. Because we frequently introduce new features into search engine interfaces, user behaviors evolve rapidly and the propensity model estimated by result randomization can be out-of-date and inconsistent with the true click bias. To model such scenarios, we fixed η as 1 for click sampling in result randomization and used different η to generate clicks for model training.

Figure 1 depicts the performance of the DNN models trained with NoCorrect, RandList and DLA with respect to different η. The presentation bias is severe when η is large and vanishes when $\eta = 0$. As shown in Figure 1, the performance of the DNN with NoCorrect is negative correlated with the value of η. Without bias correction, the training of ranking models are exposed to click bias. Thus, an increase of η will hurt the performance of models trained with raw clicks. Compared to NoCorrect, the effect of misspecified η on the DNN with RandList is more complicated. When η is 1 (which is same with the click sampling process used in result randomization), the DNN with RandList outperformed the DNN with NoCorrect. When $\eta > 1$, the relative improvements of RandList over NoCorrect are positive but keep decreasing as η increases. Although the models with RandList underestimated the real presentation bias, they are still better than those with no bias correction. When $\eta < 1$, however, the DNN with RandList performed poorly and are worse than the DNN with NoCorrect. When click biases in the training data are not as severe as they are in the randomization experiments, RandList overestimated the real biases and introduced extra noise into the training of ranking models. In comparison, the performance of the DNN with DLA is robust to changes of η and significantly outperformed NoCorrect and RandList in most cases. Because it automatically and directly estimates propensity model from training data, DLA is adaptive to changes in click biases and more robust in practice.

To explain why DLA produced a better unbiased ranker than RandList, we computed the mean square error (MSE) between the true inverse propensity weights ($\rho_0^\eta / \rho_i^\eta$) and the inverse propensity weights (g_0/g_i) estimated by DLA or RandList:

$$MSE = \frac{1}{|\pi_q|} \sum_{i=0}^{|\pi_q|-1} \left(\frac{g_0}{g_i} - \frac{\rho_0^\eta}{\rho_i^\eta} \right)^2$$

As shown in Figure 2, the MSE of RandList is small when $\eta = 1$ but large otherwise. In contrast, the MSE of DLA is small for all

Table 3: Comparison of different unbiased learning-to-rank models on Yahoo! LETOR set 1. Significant improvements or degradations with respect to the DNN with DLA are indicated with +/−.

Yahoo! LETOR set 1										
Ranking Model	Correction Method	MAP	nDCG@1	ERR@1	nDCG@3	ERR@3	nDCG@5	ERR@5	nDCG@10	ERR@10
DNN with DLA		0.816	0.658	0.338	0.662	0.412	0.683	0.433	0.729	0.447
Ranking SVM	NoCorrect	0.814^{-}	0.628^{-}	0.316^{-}	0.638^{-}	0.395^{-}	0.661^{-}	0.416^{-}	0.711^{-}	0.432^{-}
	RandList	0.812^{-}	0.642^{-}	0.330^{-}	0.653^{-}	0.407^{-}	0.675^{-}	0.428^{-}	0.721^{-}	0.442^{-}
DNN	NoCorrect	0.807^{-}	0.622^{-}	0.317^{-}	0.631^{-}	0.394^{-}	0.653^{-}	0.416^{-}	0.704^{-}	0.431^{-}
	RandList	0.814^{-}	0.658	0.338	0.659^{-}	0.412	0.679^{-}	0.433	0.725^{-}	0.447
Initial Ranker		0.804^{-}	0.559^{-}	0.271^{-}	0.586^{-}	0.357^{-}	0.617^{-}	0.381^{-}	0.675^{-}	0.397^{-}
Oracle DNN		0.830^{+}	0.667^{+}	0.339^{+}	0.675^{+}	0.414^{+}	0.695^{+}	0.435^{+}	0.740^{+}	0.449^{+}

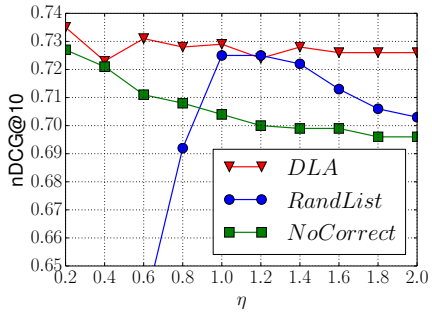

Figure 1: The performance of DNN trained with different bias corrections with respect to the value of η.

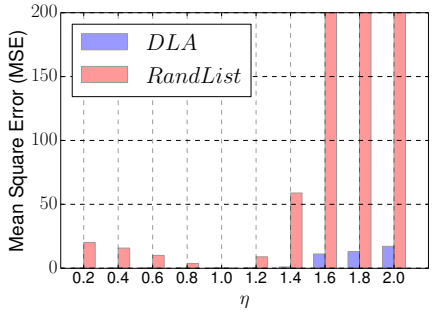

Figure 2: The MSE between the true click propensity and those estimated by DLA and RandList with respect to η.

η. This indicates that the propensity model estimated by DLA is always better for approaching the true presentation bias in spite of η. As discussed in Section 3.1, the estimation of inverse propensity weights is the key for unbiased learning to rank [16, 34]. Therefore, the models trained with DLA outperformed the models trained with RandList significantly and consistently.

6.2 Comparison with Click Models

To answer RQ2 and to demonstrate the effectiveness of DLA on real click data, we compare the performance of the DNN models trained with DLA and click models in the real-world experiment. A summary of the results is shown in Table 4.

As we can see from Table 4, the ranking models trained with click model signals (DBN and UBM) consistently outperformed the model trained with raw clicks (NoCorrect). This empirically demonstrates that click models can extract better relevance signals from user clicks and are beneficial for the learning of unbiased rankers. Among the two click models tested in our experiments, the ranker trained with DBN performed better than the ranker trained with UBM. If we compare DLA with click models, we can see that the model trained with DLA achieved significantly improvements over the models trained with DBN and UBM. It obtained more than 10% improvements over DBN and UBM in terms of ERR@10.

The fact that the DNN model trained with DLA produced better results than those trained with click models indicates that the joint learning of bias correction and ranking models is promising for unbiased learning to rank. First, the joint learning paradigm simplifies the design of search engines as it directly utilizes raw click data without requiring separate experiments to debias user feedback. This is important because such experiments are either expensive or have special requirements for data (e.g. click models require each query-document pair to appear multiple times). Second, as discussed in Section 4.3, the learning of propensity models and ranking models can help each other. A propensity model jointly learned with rankers observes more document information in the ranked lists, so it has a better chance to estimate the true click bias.

Last but not least, joint learning enables DLA to conduct end-to-end optimization for unbiased learning to rank. In fact, an important drawback of existing learning paradigms for unbiased learning to rank is that they optimize different objectives in different learning steps. For example, most click models are designed to optimize the likelihood of observed clicks. Although Malkevich et al. [20] have shown that UBM is better than DBN in terms of click simulation, the ranker trained with DBN performed better than the ranker trained with UBM in our experiments. This suggests that optimizing click likelihood doesn't necessarily produce the best bias correction model for unbiased learning to rank. Even though we implemented the propensity model with a simple examination hypothesis compared to UBM and DBN, the ranker trained with DLA still significantly outperformed the rankers trained with the click models. This demonstrates the benefits of end-to-end training with the joint learning paradigm.

7 CONCLUSION AND FUTURE WORK

In this work, we propose a Dual Learning Algorithm for automatic unbiased learning to rank. DLA jointly learns unbiased propensity models and ranking models from user clicks without any offline parameter estimation or online result randomization. Our analysis and experiments show that DLA is an theoretically principled and empirically effective framework for unbiased learning to rank. This indicates that jointly learning propensity models and ranking

Table 4: Comparison of DNN trained with DLA and relevance signals extracted by click models. Significant improvements or degradations with respect to DLA are indicated with +/−.

Correction Method	MAP	nDCG@1	ERR@1	nDCG@3	ERR@3	nDCG@5	ERR@5	nDCG@10	ERR@10
DLA	0.881	0.433	0.406	0.410	0.537	0.422	0.571	0.421	0.582
DBN	0.865^{-}	0.363	0.340	0.370	0.468^{-}	0.390	0.504^{-}	0.419	0.521^{-}
UBM	0.849^{-}	0.359^{-}	0.336^{-}	0.343^{-}	0.464^{-}	0.352^{-}	0.502^{-}	0.365^{-}	0.519^{-}
NoCorrect	0.810^{-}	0.357^{-}	0.334^{-}	0.348^{-}	0.459^{-}	0.349^{-}	0.484^{-}	0.358^{-}	0.500^{-}

models could be a fruitful direction for learning to rank with biased training signals.

Our work represents an initial attempt for automatic unbiased learning to rank and there are still many problems to study in this field. For example, as shown in Section 4.3, the performance of propensity estimation depends on the quality of the ranking model. It seems that DLA does not work well when the best ranker we can get has poor performance. We leave the investigation of these problems for future studies.

8 ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

REFERENCES

[1] Qingyao Ai, Keping Bi, Jiafeng Guo, and W. Bruce Croft. 2018. Learning a Deep Listwise Context Model for Ranking Refinement. In *Proceedings of the 41th ACM SIGIR*. ACM.
[2] Qingyao Ai, Liu Yang, Jiafeng Guo, and W Bruce Croft. 2016. Analysis of the paragraph vector model for information retrieval. In *Proceedings of the 2rd ACM ICTIR*. ACM, 133–142.
[3] Olivier Chapelle and Yi Chang. 2011. Yahoo! Learning to Rank Challenge Overview.. In *Yahoo! Learning to Rank Challenge*. 1–24.
[4] Olivier Chapelle, Thorsten Joachims, Filip Radlinski, and Yisong Yue. 2012. Large-scale validation and analysis of interleaved search evaluation. *ACM Transactions on Information Systems* 30, 1 (2012), 6.
[5] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. 2009. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM CIKM*. ACM, 621–630.
[6] Olivier Chapelle and Ya Zhang. 2009. A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th WWW*. ACM, 1–10.
[7] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An experimental comparison of click position-bias models. In *Proceedings of the 1st WSDM*. ACM, 87–94.
[8] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural Ranking Models with Weak Supervision. In *Proceedings of the 40th ACM SIGIR (SIGIR '17)*. ACM, 65–74.
[9] Anhai Doan, Raghu Ramakrishnan, and Alon Y Halevy. 2011. Crowdsourcing systems on the world-wide web. *Commun. ACM* 54, 4 (2011), 86–96.
[10] Georges E Dupret and Benjamin Piwowarski. 2008. A user browsing model to predict search engine click data from past observations.. In *Proceedings of the 31st ACM SIGIR*. ACM, 331–338.
[11] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM CIKM*. ACM, 55–64.
[12] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems* 20, 4 (2002), 422–446.
[13] Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD*. ACM, 217–226.
[14] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th annual ACM SIGIR*. Acm, 154–161.
[15] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, Filip Radlinski, and Geri Gay. 2007. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems* 25, 2 (2007), 7.
[16] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased learning-to-rank with biased feedback. In *Proceedings of the 10th ACM WSDM*. ACM, 781–789.
[17] Mark T Keane and Maeve O'Brien. 2006. Modeling Result-List Searching in the World Wide Web: The Role of Relevance Topologies and Trust Bias. In *Proceedings of the Cognitive Science Society*, Vol. 28.
[18] Aniket Kittur, Ed H Chi, and Bongwon Suh. 2008. Crowdsourcing user studies with Mechanical Turk. In *Proceedings of the SIGCHI*. ACM, 453–456.
[19] Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval* 3, 3 (2009), 225–331.
[20] Stepan Malkevich, Ilya Markov, Elena Michailova, and Maarten de Rijke. 2017. Evaluating and Analyzing Click Simulation in Web Search. In *Proceedings of the ACM ICTIR (ICTIR '17)*. ACM, 281–284.
[21] Geoffrey McLachlan and Thriyambakam Krishnan. 2007. *The EM algorithm and extensions*. Vol. 382. John Wiley & Sons.
[22] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to Match Using Local and Distributed Representations of Text for Web Search. In *Proceedings of the 26th WWW (WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 1291–1299. https://doi.org/10.1145/3038912.3052579
[23] Jay M Ponte and W Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st annual ACM SIGIR*. ACM, 275–281.
[24] 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.
[25] Karthik Raman and Thorsten Joachims. 2013. Learning socially optimal information systems from egoistic users. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 128–144.
[26] Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th annual ACM SIGIR*. Springer-Verlag New York, Inc., 232–241.
[27] Paul R Rosenbaum and Donald B Rubin. 1983. The central role of the propensity score in observational studies for causal effects. *Biometrika* 70, 1 (1983), 41–55.
[28] Anne Schuth, Harrie Oosterhuis, Shimon Whiteson, and Maarten de Rijke. 2016. Multileave gradient descent for fast online learning to rank. In *Proceedings of the 9th ACM WSDM*. ACM, 457–466.
[29] Mark D Smucker, James Allan, and Ben Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the 16th ACM CIKM*. ACM, 623–632.
[30] Adith Swaminathan and Thorsten Joachims. 2015. Batch learning from logged bandit feedback through counterfactual risk minimization. *Journal of Machine Learning Research* 16 (2015), 1731–1755.
[31] Adith Swaminathan and Thorsten Joachims. 2015. Counterfactual risk minimization: Learning from logged bandit feedback. In *ICML*. 814–823.
[32] Chao Wang, Yiqun Liu, Min Zhang, Shaoping Ma, Meihong Zheng, Jing Qian, and Kuo Zhang. 2013. Incorporating vertical results into search click models. In *Proceedings of the 36th ACM SIGIR*. ACM, 503–512.
[33] Hongning Wang, ChengXiang Zhai, Anlei Dong, and Yi Chang. 2013. Content-aware click modeling. In *Proceedings of the 22nd WWW*. ACM, 1365–1376.
[34] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to rank with selection bias in personal search. In *Proceedings of the 39th ACM SIGIR*. ACM, 115–124.
[35] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position Bias Estimation for Unbiased Learning to Rank in Personal Search. In *Proceedings of the 11th ACM WSDM (WSDM '18)*. ACM, New York, NY, USA, 610–618. https://doi.org/10.1145/3159652.3159732
[36] Wanhong Xu, Eren Manavoglu, and Erick Cantu-Paz. 2010. Temporal click model for sponsored search. In *Proceedings of the 33rd ACM SIGIR*. ACM, 106–113.
[37] Yisong Yue and Thorsten Joachims. 2009. Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proceedings of the 26th ICML*. ACM, 1201–1208.
[38] Yisong Yue, Rajan Patel, and Hein Roehrig. 2010. Beyond position bias: Examining result attractiveness as a source of presentation bias in clickthrough data. In *Proceedings of the 19th WWW*. ACM, 1011–1018.
[39] Chengxiang Zhai and John Lafferty. 2017. A study of smoothing methods for language models applied to ad hoc information retrieval. In *ACM SIGIR Forum*, Vol. 51. ACM, 268–276.
[40] Yukun Zheng, Zhen Fan, Yiqun Liu, Cheng Luo, and Shaoping Ma. 2018. SogouQCL: A New Dataset with Click Relevance Label. In *Proceedings of the 41th ACM SIGIR*. ACM.