Correcting for Selection Bias in Learning-to-rank Systems

Zohreh Ovaisi
zovais2@uic.edu
University of Illinois at Chicago

Ragib Ahsan
rahsan@uic.edu
University of Illinois at Chicago

Yifan Zhang
zhyifan3@mail3.sysu.edu.cn
Sun Yat-sen University

Kathryn Vasilaky
kvasilak@calpoly.edu
California Polytechnic State University

Elena Zheleva
ezheleva@uic.edu
University of Illinois at Chicago

ABSTRACT

Click data collected by modern recommendation systems are an important source of observational data that can be utilized to train learning-to-rank (LTR) systems. However, these data suffer from a number of biases that can result in poor performance for LTR systems. Recent methods for bias correction in such systems mostly focus on position bias, the fact that higher ranked results (e.g., top search engine results) are more likely to be clicked even if they are not the most relevant results given a user's query. Less attention has been paid to correcting for selection bias, which occurs because clicked documents are reflective of what documents have been shown to the user in the first place. Here, we propose new counterfactual approaches which adapt Heckman's two-stage method and accounts for selection and position bias in LTR systems. Our empirical evaluation shows that our proposed methods are much more robust to noise and have better accuracy compared to existing unbiased LTR algorithms, especially when there is moderate to no position bias.

CCS CONCEPTS

• **Information systems** → **Learning to rank**; **Recommender systems**; • **Computing methodologies** → *Learning from implicit feedback*.

KEYWORDS

recommender systems, learning-to-rank, position bias, selection bias

ACM Reference Format:
Zohreh Ovaisi, Ragib Ahsan, Yifan Zhang, Kathryn Vasilaky, and Elena Zheleva. 2020. Correcting for Selection Bias in Learning-to-rank Systems. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380255

1 INTRODUCTION

The abundance of data found online has inspired new lines of inquiry about human behavior and the development of machine-learning algorithms that learn individual preferences from such data. Patterns in such data are often driven by the underlying algorithms supporting online platforms, rather than naturally-occurring user behavior. For example, interaction data from social media news feeds, such as user clicks and comments on posts, reflect not only latent user interests but also news feed personalization and what the underlying algorithms chose to show to users in the first place. Such data in turn are used to train new news feed algorithms, propagating the bias further [9]. This can lead to phenomena such as filter bubbles and echo chambers and can challenge the validity of social science research that relies on found data [26, 30].

One of the places where these biases surface is in personalized recommender systems whose goal is to learn user preferences from available interaction data. These systems typically rely on learning procedures to estimate the parameters of *new ranking algorithms* that are capable of ranking items based on inferred user preferences, in a process known as *learning-to-rank (LTR)* [32]. Much of the work on unbiasing the parameter estimation for learning-to-rank systems has focused on position bias [29], the bias caused by the position where a result was displayed to a user. Position bias makes higher ranked results (e.g., top search engine results) more likely to be clicked even if they are not the most relevant.

Algorithms that correct for position bias typically assume that all relevant results have non-zero probability of being observed (and thus clicked) by the user and focus on boosting the relevance of lower ranked relevant results [29]. However, users rarely have the chance to observe all relevant results, either because the system chose to show a truncated list of top k recommended results or because users do not spend the time to peruse through tens to hundreds of ranked results. In this case, lower ranked, relevant results have zero probability of being observed (and clicked) and never get the chance to be boosted in LTR systems. This leads to selection bias in clicked results which is the focus of our work.

Here, we frame the problem of learning to rank as a counterfactual problem of predicting whether a document would have been clicked had it been observed. In order to recover from selection bias for clicked documents, we focus on identifying the relevant documents that were never shown to users. Our formulation is different from previous counterfactual formulations which correct for position bias and study the likelihood of a document being clicked

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380255

had it been placed in a higher position given that it was placed in a lower position [29].

Here, we propose a general framework for recovering from selection bias that stems from both limited choices given to users and position bias. First, we propose $Heckman^{rank}$, an algorithm for addressing selection bias in the context of learning-to-rank systems. By adapting Heckman's two-stage method, an econometric tool for addressing selection bias, we account for the limited choice given to users and the fact that some items are more likely to be shown to a user than others. Because this correction method is very general, it is applicable to any type of selection bias in which the system's decision to show documents can be learned from features. Because $Heckman^{rank}$ treats selection as a binary variable, we propose two bias-correcting ensembles that account for the nuanced probability of being selected due to position bias and combine $Heckman^{rank}$ with existing position-bias correction methods.

Our experimental evaluations demonstrate the utility of our proposed method when compared to state-of-the-art algorithms for unbiased learning-to-rank. Our ensemble methods have better accuracy compared to existing unbiased LTR algorithms under realistic selection bias assumptions, especially when the position bias is not severe. Moreover, $Heckman^{rank}$ is more robust to noise than both ensemble methods and position-bias correcting methods across difference position bias assumptions. The experiments also show that selection bias affects the performance of LTR systems even in the absence of position bias, and $Heckman^{rank}$ is able to correct for it.

2 RELATED WORK

Here, we provide the context for our work and present the three areas that best encompass our problem: bias in recommender systems, selection bias correction, and unbiased learning-to-rank.

Bias in recommender system. Many technological platforms, such as recommendation systems, tailor items to users by filtering and ranking information according to user history. This process influences the way users interact with the system and how the data collected from users is fed back to the system and can lead to several types of biases. Chaney et al. [9] explore a closely related problem called algorithmic confounding bias, where live systems are retrained to incorporate data that was influenced by the recommendation algorithm itself. Their study highlights the fact that training recommendation platforms with naive data that are not debiased can cause a severe decrease in the utility of such systems. For example, "echo chambers" are consequence of this problem [17, 20], where users are limited to an increasingly narrower choice set over time which can lead to a phenomenon called polarization [18]. Popularity bias, is another bias affecting recommender system that is studied by Celma and Cano [8]. Popularity bias refers to the idea that a recommender system will display the most popular items to a user, even if they are not the most relevant to a user's query. Recommender systems can also affect users decision making process, known as decision bias, and Chen et al. [13] show how understanding this bias can improve recommender systems. Position bias is yet another type of bias that is studied in the context of learning-to-rank systems and refers to documents that higher ranked will be more likely to be selected regardless of the document's relevancy. Joachims et al. [29] focus on this bias and we compare our results to theirs throughout.

Selection bias correction. Selection bias occurs when a data sample is not representative of the underlying data distribution. Selection bias can have various underlying causes, such as participants self-selecting into a study based on certain criteria, or subjects choosing over a choice set that is restricted in a non-random way. Selection bias could also encompass the biases listed above. Various studies attempt to correct for selection bias in different contexts.

Heckman correction, and more generally, bivariate selection models, control for the probability of being selected into the sample when predicting outcomes [21]. Smith and Elkan [40] study Heckman correction for different types of selection bias through Bayesian networks, but not in the context in learning-to-rank systems. Zadrozny [45] study selection bias in the context of well-known classifiers, where the outcome is binary rather than continuous as with ranking algorithms. Selection bias has also been studied in the context of causal graphs [4–6, 14, 15]. For example, if an underlying data generation model is assumed, Bareinboim and Pearl [3] show that selection bias can be removed even in the presence of confounding bias, i.e., when a variable can affect both treatment and control. We leverage this work in our discussion of identifiability under selection bias.

The most related work to our context are studies by Hernández-Lobato et al. [22], Schnabel et al. [37], Wang et al. [43]. Both Schnabel et al. [37] and Hernández-Lobato et al. [22] use a matrix factorization model to represent data (ratings by users) that are missing not-at-random, where Schnabel et al. [37] outperform Hernández-Lobato et al. [22]. More recently, Joachims et al. [29] propose a position debiasing approach in the context of learning-to-rank systems as a more general approach compared to Schnabel et al. [37]. Throughout, we compare our results to Joachims et al. [29], although, it should be noted that the latter deals with a more specific bias - position bias - than what we address here. Finally, Wang et al. [43] address selection bias due to confounding, whereas we address selection bias that is treatment-dependent only.

Unbiased learning-to-rank. The problem we study here investigates debiasing data in learning-to-rank systems. There are two approaches to LTR systems, offline and online, and the work we propose here falls in the category of offline LTR systems.

Offline LTR systems learn a ranking model from historical click data and interpret clicks as absolute relevance indicators [2, 7, 12, 16, 24, 27–29, 36, 41, 42]. Offline approaches must contend with the many biases that found data are subject to, including position and selection bias, among others. For example, Wang et al. [41] use a propensity weighting approach to overcome position bias. Similarly, Joachims et al. [29] propose a method to correct for position bias, by augmenting SVM^{rank} learning with an Inverse Propensity Score defined for clicks rather than queries. They demonstrate that Propensity-Weighted SVM^{rank} outperforms a standard Ranking SVM^{rank} by accounting for position bias. More recently Agarwal et al. [1] proposed nDCG SVM^{rank} that outperforms Propensity-Weighted SVM^{rank} [29], but only when position bias *is* severe. We show that our proposed algorithm outperforms [29] when position bias *is not* severe. Thus, we do not compare our results to [1].

Other studies aim to improve on Joachims et al. [29], such as Wang et al. [42] and Ai et al. [2], but only in the ease of their methodology. Wang et al. [42] propose a regression-based Expectation Maximization method for estimating the click position bias, and its main advantage over Joachims et al. [29] is that it does not require randomized tests to estimate the propensity model. Similarly, the Dual Learning Algorithm (DLA) proposed by Ai et al. [2] jointly learns the propensity model and ranking model without randomization tests. Hu et al. [24] introduce a method that jointly estimates position bias and trains a ranker using a pairwise loss function. The focus of these latter studies is position bias and not selection bias, namely the fact that some relevant documents may not be exposed to users at all, which is what we study here.

In contrast to offline LTR systems, online LTR algorithms intervene during click collection by interactively updating a ranking model after each user interaction [11, 23, 25, 33, 35, 38, 39, 44]. This can be costly, as it requires intervening with users' experience of the system. The main study in this context is Jagerman et al. [25] who compare the online learning approach by Oosterhuis and de Rijke [33] with the offline LTR approach proposed by Joachims et al. [29] under selection bias. The study shows that the method by Oosterhuis and de Rijke [33] outperforms [29] when selection bias and moderate position bias exist, and when no selection bias and severe position bias exist. One advantage of our offline algorithms over online LTR ones is that they do not have a negative impact on user experience while learning.

3 PROBLEM DESCRIPTION

In this section, we review the definition of learning-to-rank systems, position and selection bias in recommender systems, as well as our framing of bias-corrected ranking with counterfactuals.

3.1 Learning-to-Rank Systems

We first describe learning-to-rank systems assuming knowledge of true relevances (full information setting) following [29]. Given a sample $\mathbf{x}$ of i.i.d. queries ($\mathbf{x}_i \sim P(\mathbf{x})$) and relevancy score $\text{rel}(\mathbf{x}, y)$ for all documents y, we denote $\Delta\ (\mathbf{y}|\mathbf{x}_i)$ to be the loss of any ranking $\mathbf{y}$ for query $\mathbf{x}_i$. The risk of ranking system S that returns ranking $S(\mathbf{x})$ for queries $\mathbf{x}$ is given by:

$$R(S) = \int \Delta\ (S(\mathbf{x})|\mathbf{x})\ d\ P(\mathbf{x}). \tag{1}$$

Since the distribution of queries is not known in practice, $R(S)$ cannot be computed directly, it is often estimated empirically as follows:

$$\hat{R}(S) = \frac{1}{|\mathbf{X}|} \sum_{\mathbf{x}_i \in \mathbf{X}} \Delta\ (S(\mathbf{x}_i)|\mathbf{x}_i). \tag{2}$$

The goal of learning-to-rank systems is to find a ranking function $S \subset \mathcal{S}$ that minimizes the risk $\hat{R}(S)$. Learning-to-rank systems are a special case of a recommender system where, appropriate ranking is learned.

The relevancy score $\text{rel}(\mathbf{x}_i, y)$ denotes the true relevancy of document y for a specific query $\mathbf{x}_i$. It is typically obtained via human annotation, and is necessary for the full information setting. Despite being reliable, true relevance assignments are frequently impossible or expensive to obtain because they require a manual evaluation of every possible document given a query.

Due to the cost of annotation, recommender system training often relies on implicit feedback from users in what is known as *partial information setting*. Click logs collected from users are easily observable in any recommender system, and can serve as a proxy to the relevancy of a document. For this reason clicks are frequently used to train new recommendation algorithms. Unfortunately, there is a cost for using click log data because of noise (e.g., people can click on items that are not relevant) and various biases that the data are subject to, including position bias and selection bias which we discuss next.

3.2 Position bias

Implicit feedback (clicks) in LTR systems is inherently biased. Position bias refers to the notion that higher ranked results are more likely to be clicked by a user even if they are not the most relevant results given a user's query.

Previous work [29, 41] has focused on tempering the effects of position bias via inverse propensity weighting (IPW). IPW reweights the relevance of documents using a factor inversely related to the documents' position on a page. For a given query instance $\mathbf{x}_i$, the relevance of document y to query $\mathbf{x}_i$ is $r_i(y) \in \{0, 1\}$, and $\mathbf{o}_i \in \{0, 1\}$ is a set of vectors indicating whether a document y is observed. Suppose the performance metric of interest is the sum of the rank of relevant documents:

$$\Delta\ (\mathbf{y}|\mathbf{x}_i, r_i) = \sum_{y \in \mathbf{y}} rank(y|\mathbf{y})\ r_i(y). \tag{3}$$

Due to position bias, given a presented ranking $\bar{\mathbf{y}}_i$, clicks are more likely to occur for top-ranked documents. Therefore, the goal is to obtain an unbiased estimate of $\Delta\ (\mathbf{y}|\mathbf{x}_i, r_i)$ for a new ranking $\mathbf{y}$.

There are existing approaches that address position bias in LTR systems. For example, *Propensity* SVM^{rank}, proposed by Joachims et al. [29], is one such algorithm. It uses inverse propensity weights (IPW) to counteract the effects of position bias:

$$\hat{\Delta}_{IPW}\ (\mathbf{y}|\mathbf{x}_i, \bar{\mathbf{y}}_i, o_i) = \sum_{y: o_i = 1 \wedge r_i = 1} \frac{rank(y|\mathbf{y})}{Q(o_i = 1|\mathbf{x}_i, \bar{\mathbf{y}}_i, r_i)} \tag{4}$$

where the propensity weight $Q(o_i = 1|\mathbf{x}_i, \bar{\mathbf{y}}_i, r_i)$ denotes the marginal probability of observing the relevance $r_i(y)$ of result y for query $\mathbf{x}_i$, when the user is presented with ranking $\bar{\mathbf{y}}_\mathbf{i}$. Joachims et al. [29] estimated the IPW to be:

$$Q(o_i = 1|\mathbf{x}_i, \bar{\mathbf{y}}_i, r_i) = \left(\frac{1}{rank(y|\bar{\mathbf{y}}_i)}\right)^{\eta} \tag{5}$$

where η is severity of position bias. The IPW has two main properties. First, it is computed only for documents that are observed and clicked. Therefore, documents that are never clicked do not contribute to the IPW calculation. Second, as shown by Joachims et al. [29], a ranking model trained with clicks and the IPW method will converge to a model trained with true relevance labels, rendering a LTR framework robust to position bias.

3.3 Selection bias

LTR systems rely on implicit feedback (clicks) to improve their performance. However, a sample of relevant documents from click data does not reflect the true distribution of all possible relevant documents because a user observes a limited choice of documents. This can occur because i) a recommender system ranks relevant documents too low for a user to feasibly see, or ii) because a user can examine only a truncated list of top k recommended items. As a result, clicked documents are not randomly selected for LTR systems to be trained on, and therefore cannot reveal the relevancy of documents that were excluded from the ranking $\bar{\mathbf{y}}$. This leads to selection bias.

Selection bias and position bias are closely related. Besides selection bias due to unobserved relevant documents, selection bias can also arise due to position bias: lower ranked results are less likely to be observed, and thus selected more frequently than higher-ranked ones. Previous work on LTR algorithms that corrects for position bias assigns a non-zero observation probability to all documents, and proofs of debiasing are based on this assumption [29]. However, in practice it is rarely realistic to assume that all documents can be observed by a user. When there is a large list of potentially relevant documents, the system may choose to show only the top k results and a user can only act on these results. Therefore, lower-ranked results are never observed, which leads to selection bias. Here, we consider the selection bias that arises when some documents have a zero probability of being observed if they are ranked below a certain cutoff k. The objective of this paper is to propose a ranking algorithm that corrects for both selection and position bias, and therefore is a better tool for training future LTR systems (see Section 4.1).

3.4 Ranking with counterfactuals

We define the problem of ranking documents as a counterfactual problem [34]. Let $O(\mathbf{x}, y) \in \{0, 1\}$ denote a treatment variable indicating whether a user observed document y given query $\mathbf{x}$. Let $C_{O=1}(\mathbf{x}, y) \in \{0, 1\}$ represent the *click counterfactual* indicating whether a document y would have been clicked had y been observed under query $\mathbf{x}$. The goal of *ranking with counterfactuals* is to reliably estimate the probability of click counterfactuals for all documents:

$$P(C_{O=1} = 1|\mathbf{X} = \mathbf{x}, Y = y) \tag{6}$$

and then rank the documents according to this probability. Solving the ranking with counterfactuals problem would allow us to find a ranking system S that returns ranking $S(\mathbf{x})$ for query $\mathbf{x}$ that is robust to selection bias.

Current techniques that correct for position bias aim to provide reliable estimates of this probability by taking into consideration the rank-dependent probability of being observed. However, this approach is only effective for documents that have a non-zero probability of being observed:

$$P(C_{O=1} = 1|O = 1, rank = i, \mathbf{X} = \mathbf{x}, Y = y). \tag{7}$$

The challenge with selection bias is to estimate this probability for documents that have neither been observed nor clicked in the first place:

$$P(C_{O=1} = 1|O = 0, C = 0, \mathbf{X} = \mathbf{x}, Y = y) \tag{8}$$

$$= P(C_{O=1} = 1|O = 0, \mathbf{X} = \mathbf{x}, Y = y). \tag{9}$$

To address this challenge, in the following Section 4 we turn to econometric methods, which have a long history of addressing selection bias.

Note that in order to recover from selection bias we must address the concept of identifiability and whether causal estimates can even be obtained in the context of our setup. A bias is identified to be recoverable if the treatment is known [6]. In our context the treatment is whether a document enters into the data training pool (clicked). While it is difficult to guarantee that a user observed a document that was shown to them (i.e. we cannot know whether an absence of a click is due non-observance or to non-relevance), it is easier to guarantee that a document was not observed by a user if it was not shown to them in the first place (e.g., it is below a cutoff for top k results or the user never scrolled down to that document in a ranked list). Our proposed solution, therefore, identifies the treatment first as a binary variable (whether the document is shown versus not shown) and then as a continuous variable that takes position bias into account.

4 BIAS-CORRECTED RANKING WITH COUNTERFACTUALS

In this section we adapt a well-known sample selection correction method, known as Heckman's two-stage correction, to the context of LTR systems. Integrating the latter framework requires a detailed understanding of how LTR systems generally process and cut interaction data to train new recommendation algorithms, and at what stages in that process selection biases are introduced. Thus, while the methodology we introduce is a well established tool in the causal inference literature, integrating it within the multiple stages of training a machine learning algorithm is a complex translational problem. We then introduce two aggregation methods to combine our proposed $Heckman^{rank}$, correcting for selection bias, with existing methods for position bias to further improve the accuracy in ranking prediction.

4.1 Selection bias correction with $Heckman^{rank}$

Econometrics, or the application of a statistical methods to economic problems, has long been concerned with confounded or held-out data in the context of consumer choices. Economists are interested in estimating models of consumer choice to both learn consumers' preferences and to predict their outcomes. Frequently, the data used to estimate these models are observational, not experimental. As such, the outcomes observed in the data are based on a limited and self-selected sample. A quintessential example of this problem is estimating the impact of education on worker's wages based on only those workers who are employed [21]. However, those who are employed are a self-selected sample, and estimates of education's effect on wages will be biased.

A broad class of models in econometrics that deal with such selection biases are known as bivariate sample selection models. A well-known method for correcting these biases in economics is known as Heckman correction or two-step Heckman. In the first stage the probability of self selection is estimated, and in the second stage the latter probability is accounted for. As Heckman [21] pointed out self selection bias can occur for two reasons. "First, there may be self selection by the individuals or data units being investigated (as in the labor example). Second, sample selection decisions by analysts or data processors operate in much the same fashion as self selection (by individuals)."

Adapting a sample selection model, such as Heckman's, to LTR systems requires an understanding of when and how data are progressively truncated when training a recommender algorithm. We introduce notation and a framework to outline this problem here.

Let $c_{x,y}$ denote whether a document y is selected (e.g., clicked) under query x for each $< query, document >$ pair; $\boldsymbol{F}_{x,y}$ represents the features of the $< query, document >$, and $\epsilon_{x,y}$ is a normally distributed error term. The same query can produce multiple $< query, document >$ pairs, where the documents are then ordered by a LTR algorithm. However, it is important to note that a LTR algorithm will not rank every single document in the data given a *query*. Unranked documents are typically discarded when training future algorithms. Herein lies the selection bias. Documents that are not shown to the user can then never be predicted as a potential choice. Moreover, documents far down in the rankings may still be kept in future training data, but will appear infrequently. Both these points will contribute to generating increasingly restrictive data that new algorithms are trained on.

If we fail to account for the repercussions of these selection biases, then modeling whether a document is selected will be based only upon the features of documents that were ranked and shown to the user, which can be written as:

$$c_{x,y} = \boldsymbol{\alpha}^{biased}\boldsymbol{F}_{x,y} + \epsilon_{x,y}. \tag{10}$$

In this setup we only consider a simple linear model; however, future research will incorporate nonlinear models. In estimating (10), we refer to the feature weights estimator, $\boldsymbol{\alpha}^{biased}$, as being biased, because the feature design matrix will only reflect documents that were shown to the user. But documents that were not shown to the user could also have been selected. Thus, (10) reflects the limitation outlined in (7). When we discard unseen documents then we can only predict clicks for documents that were shown, while our objective is to predict the unconditional probability that a document is clicked regardless of whether it was shown.

To address this point, we will first explicitly model an algorithm's document selection process. Let $o_{x,y}$ denote a binary variable that indicates whether a document y is shown and observed ($o_{x,y} = 1$) or not shown and not observed ($o_{x,y} = 0$). For now, we assume that if a document is shown to the user that user also sees the document. We relax this assumption in Section 4.2. $\boldsymbol{Z}_{x,y}$ is a set of explanatory variables that determine whether a document is shown, which includes the features in $\boldsymbol{F}_{x,y}$, but can also include external features, including characteristics of the algorithm that first generated the data:

$$o_{x,y} = \boldsymbol{\theta}\boldsymbol{Z}_{x,y} + \epsilon^{(1)}_{x,y}. \tag{11}$$

In the first stage of $Heckman^{rank}$, we estimate the probability of a document being observed using a Probit model:

$$P(o_{x,y} = 1|Z_{x,y}) = P(\boldsymbol{\theta}\boldsymbol{Z}_{x,y} + \epsilon_{x,y} > 0|Z_{x,y}) = \Phi(\boldsymbol{\theta}\boldsymbol{Z}_{x,y}) \tag{12}$$

where $\Phi()$ denotes the standard normal CDF. Note that a crucial assumption here is that we will use both seen and unseen documents for a given *query* in estimating (12). Therefore, the dimensions of our data will be far larger than if we had discarded unseen documents, as most LTR systems typically do. After estimating (12) we can compute what is known as an Inverse Mills ratio for every $< query, document >$ pair:

$$\lambda_{x,y} = \frac{\phi(\boldsymbol{\theta}\boldsymbol{Z}_{x,y})}{\Phi(\boldsymbol{\theta}\boldsymbol{Z}_{x,y})} \tag{13}$$

where $\phi()$ is the standard normal distribution. $\lambda_{x,y}$ reflects the severity of selection bias and corresponds to our desire to condition on $O = 0$ versus $O = 1$, as described in Equations 7 and 9, but using a continuous variable reflecting the probability of selection.

In the second stage of $Heckman^{rank}$, we estimate the probability of whether a user will click on a document. Heckman's seminal work showed that if we condition our estimates on the $\lambda_{x,y}$ our estimated feature weights will be statistically unbiased in expectation. This can improve our predictions if we believe that including $\lambda_{x,y}$ is relevant in predicting clicks. We assume joint normality of the errors, and our setup naturally implies that the error terms $\epsilon_{x,y}$ and $\epsilon^{(1)}_{x,y}$ are correlated, namely that clicking on a document depends upon whether a document is observed by users and, therefore, has the potential for being selected.

The conditional expectation of clicking on a document conditional on the document being shown is given by:

$$\mathbb{E}[c_{x,y}|F, p_{x,y} = 1] = \boldsymbol{\alpha}\boldsymbol{F}_{x,y} + \mathbb{E}(\epsilon_{x,y}|F, o_{x,y} = 1) = \boldsymbol{\alpha}\boldsymbol{F}_{x,y} + \sigma\boldsymbol{\lambda}_{x,y} \tag{14}$$

We can see that if the error terms in (10) and (11) are correlated then $\mathbb{E}(\epsilon_{x,y}|F, o_{x,y} = 1) > 0$), and estimating (14) without accounting for this correlation will lead to biased estimates of $\boldsymbol{\alpha}$. Thus, in the second stage, we correct for selection bias to obtain an unbiased estimate of $\boldsymbol{\alpha}$ by controlling for $\hat{\boldsymbol{\lambda}}_{x,y}$:

$$c_{x,y} = \boldsymbol{\alpha}^{unbiased}\boldsymbol{F}_{x,y} + \sigma\hat{\boldsymbol{\lambda}}_{x,y}(\hat{\boldsymbol{\theta}}\boldsymbol{Z}_{x,y}) + \epsilon_{x,y} \tag{15}$$

Estimation of (15) allows us to predict click probabilities, $\hat{c}$, where $\hat{c} = \hat{\boldsymbol{\alpha}}^{unbiased}\boldsymbol{F}_{x,y} + \hat{\sigma}\hat{\boldsymbol{\lambda}}_{x,y}(\hat{\boldsymbol{\theta}}\boldsymbol{Z}_{x,y})$. This click probability refers to our ability to estimate (9), the unconditional click probability, using $Heckman^{rank}$. We then compute document rankings for a given query by sorting documents according to their predicted click probabilities. Note that our main equation (15) has a bivariate outcome. Thus, in this selection correction setup we are following a Heckprobit model, as opposed to the initial model that Heckman proposed in Heckman [21] where the main outcome is a continuous variable.

Our setup helps account for the inherent selection bias that can occur in any LTR system, as all LTR systems must make a choice in what documents they show to a user. What is unique to our formulation of the problem is our use of a two stage estimation process to account for the two stage document selection process: namely, whether the document is shown, and whether the document is then selected. Accounting for the truncation of the data is critical for training a LTR system, and previously has not been considered. In order to improve a system's ranking accuracy it must be able to predict document selection for both unseen as well as seen documents. If not, the choice set of documents that are available to a user can only become progressively smaller. Our correction is a simple method to counteract such a trend in found data.

4.2 Bias-correcting ensembles

Biased data limits the ability to accurately train an LTR algorithm on click logs. In this section, we present methods for addressing two types of selection bias, one stemming from truncated recommendations and the other one from position bias. One of the deficiencies of using $Heckman^{rank}$ to deal with biased data is that it assumes that all documents that are shown to a user are also observed by the user. However, due to position bias that is not necessarily the case, and lower-ranked shown documents have lower probability of being observed. Therefore, it is natural to consider combining $Heckman^{rank}$, which focuses on recovering from selection bias due to unobserved documents, with a ranking algorithm that accounts for the nuanced observation probability of shown documents due to position bias.

Algorithms that rely on IPW [1, 29, 41] consider the propensity of observation for any document given a ranking for a certain query and it is exponentially dependent on the rank of the document in the given ranking. This is clearly different from our approach for recovering from selection bias where we model the observation probability to be either 0 or 1 depending on its position relative in the ranking.

Ensemble ranking objective. In order to harness the power of correcting for these biases in a collective manner, we propose to use ensembles that can combine the results produced by $Heckman^{rank}$ and any position bias correcting method. We refer to the algorithm correcting for selection bias as A_s and for position bias as A_p while $\mathbf{y}_s$ and $\mathbf{y}_p$ are the rankings generated for a certain query $\mathbf{x}$ by the algorithms respectively. Our goal is to produce an ensemble ranking $\mathbf{y}_e$ based on $\mathbf{y}_s$ and $\mathbf{y}_p$ for all queries that is more accurate than either ranking alone.

There is a wide scope for designing an appropriate ensemble method to serve our objective. We propose two simple but powerful approaches, as our experimental evaluation shows. The two approaches differ in their fundamental intuition. The intuition behind the first approach is to model the value of individual ranking algorithms through a linear combination of the rankings they produce. We can learn the coefficients of that linear combination using linear models on the training data. We call this method *Linear Combination.* The second approach is a standard approach for combining ranking algorithms using Borda counts [19]. It works as a post processing step after the candidate algorithms A_s and A_p produce their respective rankings $\mathbf{y}_s$ and $\mathbf{y}_p$. We apply a certain *Rank Aggregation* algorithm over $\mathbf{y}_s$ and $\mathbf{y}_p$ to produce $\mathbf{y}_e$ for a given query for evaluation. Next, we discuss each of the approaches in the context of our problem.

4.2.1 ***Linear Combination.***

A simple aggregation method for combining A_s and A_p is to estimate the value of each algorithm in predicting a click. After training the algorithms A_s and A_p, we use the same training data to learn the weights of a linear model that considers the rank of each document produced by A_s and A_p. For any given query $\mathbf{x}$ the ranking of document y produced by A_s is given by $rank(y|\mathbf{y}_s, \mathbf{x})$. Similarly, $rank(y|\mathbf{y}_p, \mathbf{x})$ represents the ranking given by A_p. We also consider the relevance of document y, $rel(\mathbf{x}, y)$ which is either 0 for not relevant or 1 for relevant, modeled through clicks.

We train a binary classifier to predict relevance (click) of documents which incorporates the estimated value of individual algorithms. We select logistic regression to be the binary classifier in our implementation, but any other standard classification method should work as well. We model the relevance of a document y, given two rankings $\mathbf{y}_s$ and $\mathbf{y}_p$ as the following logistic function:

$$rel(\mathbf{x}, y) = \frac{1}{1 + e^{-(w_0 + w_1 * rank(y|\mathrm{y_s}, \mathbf{x}) + w_2 * rank(y|\mathrm{y_p}, \mathbf{x}))}}$$

Upon training the logistic regression model we learn the parameters w_0, w_1, w_2 where w_1 and w_2 represent the estimated impact of A_s and A_p respectively. During evaluation we predict the click counterfactual probability for each $< query, document >$ pair using the trained classifier. Then we can sort the documents for each query according to these probability values to generate the final ensemble ranking $\mathbf{y}_e$.

4.2.2 ***Rank Aggregation.***

Rank aggregation aims to combine rankings generated by multiple ranking algorithms. In a typical rank aggregation problem, we are given a set of rankings $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_m$ of a set of objects $y_1, y_2, ..., y_n$ given a query $\mathbf{x}$. The objective is to find a single ranking $\mathbf{y}$ that corroborates with all other existing rankings. Many aggregation methods have been proposed [31]. A commonly used approach is the Borda count, which scores documents based on their relative position in a ranking, and then totals all scores across all rankings for a given query [19].

In our scenario, we have two rankings $\mathbf{y}_s$ and $\mathbf{y}_p$. For a given query, there are n documents to rank. Consider $B(\mathbf{y}_s, y_i)$ as the score for document y_i ($i \in \{1, 2, ..., n\}$) given by $\mathbf{y}_s$. Similarly, $B(\mathbf{y}_p, y_i)$ refers to the score for document y_i given by $\mathbf{y}_p$. The total score for document y_i would be $B(\mathbf{y}_s, y_i) + B(\mathbf{y}_p, y_i)$. Based on these total scores we sort the documents in non-ascending order of their scores to produce the ensemble ranking $\mathbf{y}_e$. The score given to a document y_i in a specific ranking $\mathbf{y}_s$ (or $\mathbf{y}_p$) is simply the number of documents it beats in the respective ranking. For example, given a certain query, if a document is ranked 1^{st} in $\mathbf{y}_s$ and 3^{rd} in $\mathbf{y}_p$ then the total score for this document would be $(n-1)+(n-3)$. This very simple scheme reflects the power of the combined method to recover from different biases in LTR systems.

5 EXPERIMENTS

In this section, we evaluate our proposed approach for addressing selection bias under several conditions:

- Varying the number of observed documents given a fixed position bias (Section 5.2)
- Varying position bias with no noise (Section 5.2.1)
- Varying position bias with noisy clicks (Section 5.2.2)
- Varying noise level in click sampling (Section 5.2.3)

The parameter values are summarized in Table 1.

5.1 Experimental setup

Next, we describe the dataset we use, the process for click data generation, and the evaluation framework.

5.1.1 ***Base dataset***.

In order to explore selection bias in LTR systems, we conduct several experiments using semi-synthetic datasets based on set 1 and set 2 from the Yahoo! Learning to Rank Challenge (C14B) [1], denoted as D_{YLTR}. Set 1 contains 19, 944 train and 6, 983 test queries including 473, 134 train and 165, 660 test documents. Set 2 contains 1, 266 train queries and 34, 815 train documents, with 20 documents per query on average [10]. Each query is represented by an *id* and each <query, document> pair is represented by a 700-dimensional feature vector with normalized feature values $\in [0, 1]$. The dataset contains true relevance of rankings based on expert annotated relevance score $\in [0, 4]$ associated with each $< query, document >$ pair, with 0 meaning least relevant and 4 most relevant. We binarized the relevance score following Joachims et al. [29], such that 0 denotes irrelevant (a relevance score of 0, 1 or 2), and 1 relevant (a score of 3 and 4).

We first conduct extensive experiments on the train portion of the smaller set 2, where we randomly sample 70% of the queries as training data and 30% as test data, with which LTR algorithms can be trained and evaluated respectively (Section 5.2). To confirm the performance of our proposed method with out-of-sample test data, we conduct experiments on the larger set 1, where we train LTR algorithms on set 1 train data and evaluate them on set 1 test data (Section 5.3).

5.1.2 ***Semi-synthetic data generation***.

We use the real-world base dataset, D_{YLTR}, to generate semi-synthetic datasets that contain document clicks for $< query, document >$ rankings. The main motivation behind using the Yahoo! Learning To Rank dataset is that it provides unbiased ground truth for relevant results, thus enabling unbiased evaluation of ranking algorithms. In real-world scenarios, unbiased ground truth is hard to come by and LTR algorithms are typically trained on biased, click data which does not allow for unbiased evaluation. To mimic real-world scenarios for LTR, the synthetic data generation creates such biased click data.

We follow the data-generation process of Joachims et al. [29]. We train a base ranker, in our case SVM^{rank}, with 1% of the training dataset that contains true relevances, and then use the trained model to generate rankings for the remaining 99% of the queries in the training dataset. The second step of the data-generation process generates clicks on the ranked documents in the training dataset. The click probability of document y for a given query $\mathbf{x}$ is calculated as $P(c_{x,y} = 1) = \frac{r_i(y)}{(rank(y|\bar{\mathbf{y}}))^{\eta}}$ where $c_{x,y}(y)$ and $r_i(y)$ represent if a document y is clicked and relevant respectively, $rank(y|\bar{\mathbf{y}})$ denotes the ranking of document y for query $\boldsymbol{x}$ if the user was presented the ranking $\bar{\mathbf{y}}$, and η indicates the severity of position bias. Note that clicks are not generated for documents that are bellow a certain rank cutoff k to incorporate the selection bias.

In a single pass over the entire training data we generate clicks following the above click probability. We refer to this as one sampling pass. For the smaller set 2, we generate clicks over 15 sampling passes, while for the larger set 1, we generate clicks over 5 sampling passes. This click-generation process reflects a common user behavior where some relevant documents do not receive any clicks, and other relevant documents receive multiple clicks. This process captures the generation of *noiseless* clicks, where users only click on relevant documents. We also consider a click generation process with *noisy* clicks in which a small percentage of clicks (10 − 30%) occur on irrelevant documents.

parameter	value/category	description	section
k	1-30	number of observed docs (selection bias)	5.2
η	0, 0.5, 1, 1.5, 2	position bias severity	5.2.1
noise	0%, 10%, 20%, 30%	clicks on irrelevant docs	5.2.2, 5.2.3

Table 1: Experimental Parameters

5.1.3 ***Evaluation***.

We explore the performance of LTR algorithms *Naive* SVM^{rank}, *Propensity* SVM^{rank}, $Heckman^{rank}$ along with the two ensemble methods *Linear Combination* (CombinedW) and *Rank Aggregation* (RankAgg) with two different metrics: *Average Rank of Relevant Results* $ARRR = \frac{\sum_{y:o_i=1 \wedge r_i=1} rank(y|\bar{\mathbf{y}})}{|\mathbf{X}|}$ and *Normalized Discounted Cumulative Gain* $nDCG@p = nDCG_p) = \frac{DCG@p}{IDCG@p}$ where p is the rank position up to which we are interested to evaluate, $DCG@p$ represents the discounted cumulative gain of the given ranking whereas $IDCG@p$ refers to the ideal discounted cumulative gain. We can compute $DCG@p$ using the following formula $DCG@p = \sum_{i=1}^{p} \frac{2^{rel(\mathrm{x},y)}-1}{log_2(i+1)}$ Similarly, $IDCG@p = \sum_{i=1}^{|REL@p|} \frac{2^{rel(\mathrm{x},y)}-1}{log_2(i+1)}$ where $REL@p$ represents the list of relevant documents (ordered by their relevance) in the ranking up to position p for a given query. In our evaluation we chose $p = 10$ for nDCG metric and we refer to it by nDCG@10.

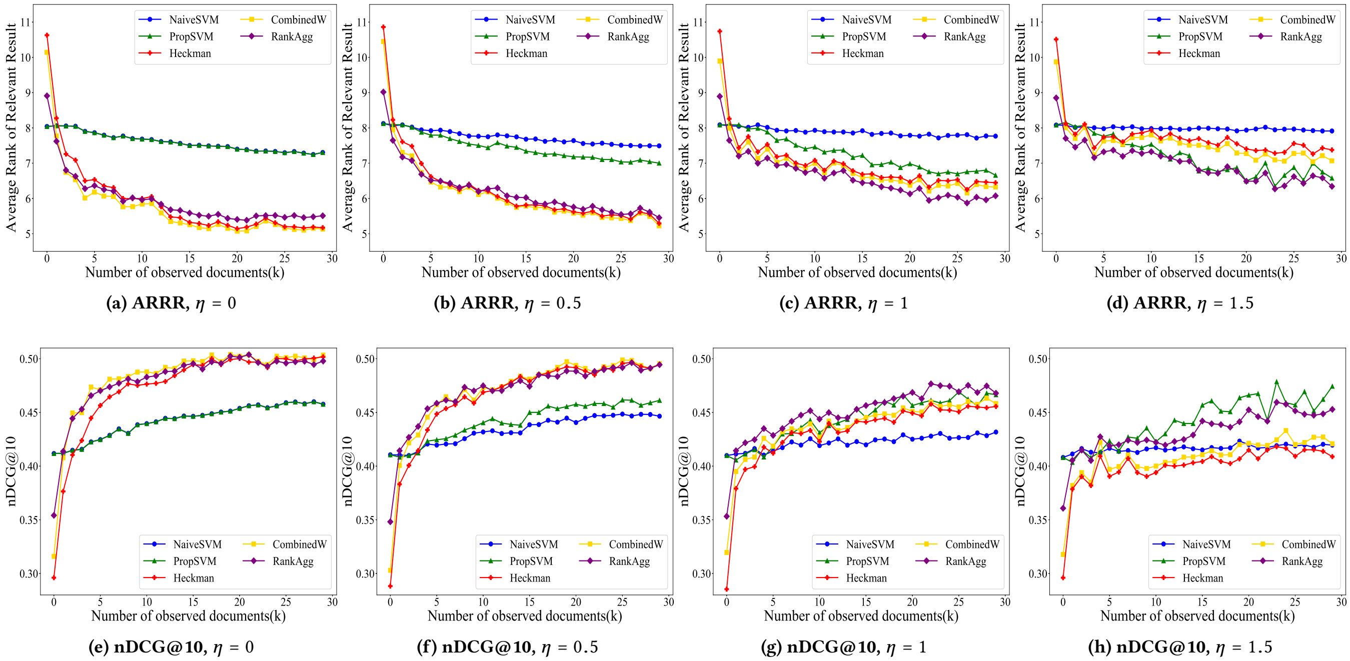
Each figure in the experiments depicts how the ARRR or nDCG@10 (y axis) changes when the user only observes the first $k \in [1, 30]$ documents (x axis). Note that k reflects the severity of selection bias as we model selection bias by assigning a zero observation probability to documents below cutoff k. In contrast, position bias is modeled by assigning a non-zero probability to every single document where η represents the severity of the position bias. We vary severity of both selection bias and position bias with or without the existence of noise in click generation.

When training *Propensity* SVM^{rank}, we apply an Inverse Propensity Score for clicked documents $Q(o(y) = 1|\mathbf{x}, \bar{\mathbf{y}}, r) = (\frac{1}{rank(y|\bar{\mathbf{y}})})^{\eta}$ where o and r represent whether a document is observed and relevant respectively, following Joachims et al. [29]. $Q(o(y) = 1|\mathbf{x}, \bar{\mathbf{y}}, r)$ is the propensity score denoting the marginal probability of observing the relevance of result y for query $\mathbf{x}$ if the user was presented the ranking $\bar{\mathbf{y}}$, and η indicates the severity of position bias.

$Heckman^{rank}$ is implemented following the steps described in section 4. In step 1, the documents that appear among the n shown results for each query are considered observed ($o_{x,y} = 1$), and the remainder as not-observed ($o_{x,y} = 0$). It is important to note that other LTR algorithms throw away the documents with $o_{x,y} = 0$ in training, while we do not. In our implementation $\boldsymbol{Z}$ only includes the feature set common to $\boldsymbol{F}_{x,y}$. For the ensemble methods, the selection bias recovery algorithm A_s is $Heckman^{rank}$ and the position bias recovery algorithm A_p is *Propensity* SVM^{rank}.

Given the model learned during training, each algorithm ranks the documents in the test set. In the following subsections, we evaluate each algorithm performance under different scenarios. For evaluation, the (noiseless) clicked documents in the test set are considered to be relevant documents, and the average rank of relevant results (ARRR) across queries along with nDCG@10 is our metric to evaluate each algorithm's performance.

[1] https://webscope.sandbox.yahoo.com/catalog.php?datatype=c



(a) ARRR, $\eta = 0$ **(b) ARRR, $\eta = 0.5$** **(c) ARRR, $\eta = 1$** **(d) ARRR, $\eta = 1.5$**

(e) nDCG@10, $\eta = 0$ **(f) nDCG@10, $\eta = 0.5$** **(g) nDCG@10, $\eta = 1$** **(h) nDCG@10, $\eta = 1.5$**

Figure 1: The performance of LTR algorithms on set 2. Lower is better for ARRR, higher is better for nDCG@10.

5.2 Experimental results on set 2

Here, we evaluate the performance of each algorithm under different levels of position bias ($\eta = 0, 0.5, 1, 1.5, 2$) and when clicks are *noisy* or *noiseless* (0%, 10%, 20% and 30% noise). In each case, we use ARRR and nDCG@10.

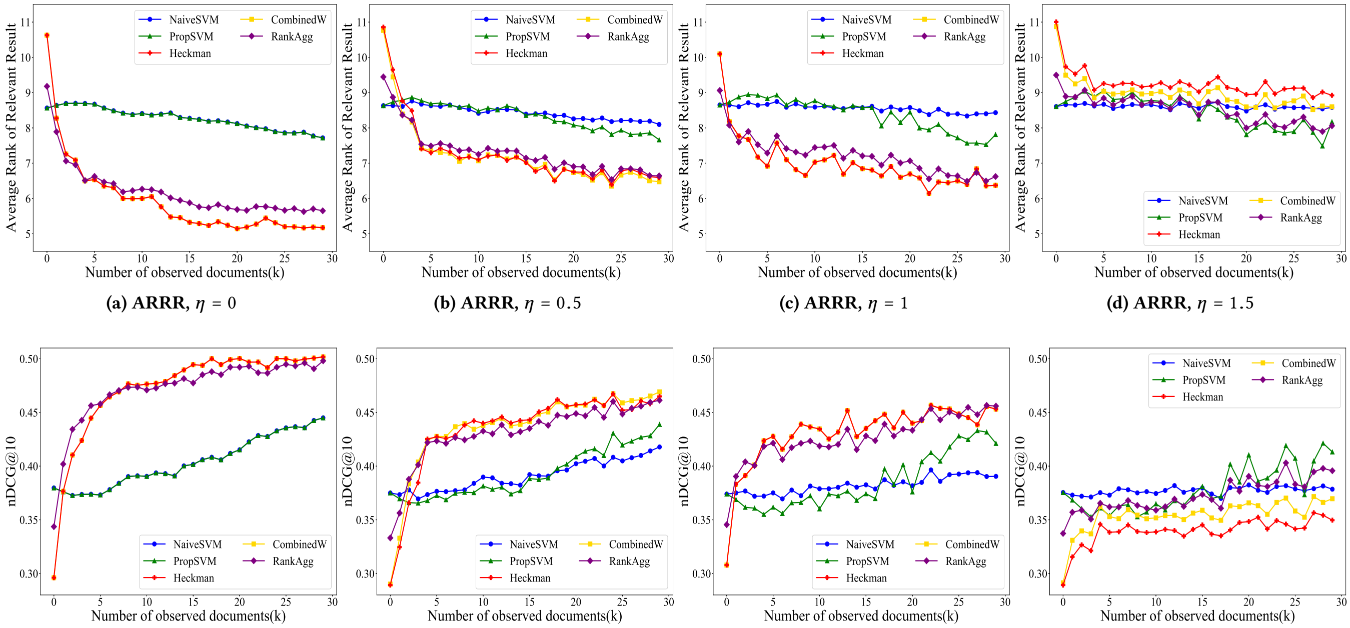
5.2.1 ***Effect of position bias.*** Figure 1 illustrates the performance of all LTR algorithms and ensembles for varying degrees of position bias ($\eta \in \{0, 0.5, 1, 1.5\}$). Figures 1a, 1b, 1c and 1d show the performance as ARRR. Figures 1e, 1f, 1g and 1h show nDCG@10. Due to space, we omit the figures for $\eta = 2$ since $\eta = 1.5$ captures the trend of $propensitySVM$ starting to work better than the other methods. $Heckman^{rank}$ suffers when there is severe position bias.

Figures 1a-1d illustrate that $Heckman^{rank}$ outperforms $Propensity\ SVM^{rank}$ in the absence of position bias ($\eta = 0$), or when position bias is low ($\eta = 0.5$) and moderate ($\eta = 1$). The better performance of $Heckman^{rank}$ over $Propensity\ SVM^{rank}$ vanishes with increased position bias, such that at a high position bias level ($\eta = 1.5$), $Heckman^{rank}$ falls behind $Propensity\ SVM^{rank}$, but still outperforms Naive SVM^{rank}. The reason for this is that a high position bias results in a high click frequency for top-ranked documents, leaving low-ranked documents with a very small chance of being clicked. $Heckman^{rank}$ is designed to control for the probability of a document being observed. If top-ranked documents have a disproportionately higher density in click data relative to low-ranked documents, then the predicted probabilities in $Heckman^{rank}$ will also reflect this imbalance. In terms of algorithms that address both position bias and selection bias, Figures 1a, 1b show that for $\eta = 0, 0.5$, *combinedW* and *RankAgg* outperform both $Propensity\ SVM^{rank}$ and $Heckman^{rank}$ for $k \lessapprox 7$. Moreover, Figures 1c and 1d show that for $\eta = 1$ and $\eta = 1.5$ *RankAgg* outperforms its component algorithms for almost all values of k.

When *ARRR* is the metric of interest, Figure 1a, 1b, 1c and 1d illustrate that $Heckman^{rank}$ outperforms $Propensity\ SVM^{rank}$ in the absence of position bias ($\eta = 0$) and when position bias is low to moderate ($\eta = \{0.5, 1\}$), while it falls behind $Propensity\ SVM^{rank}$ when position bias increases ($\eta = 1.5$). To compare it to the results for *nDCG*@10 illustrated in Figures 1e, 1f, 1g and 1h, $Heckman^{rank}$ appears to start lagging behind in performance at $\eta = 1.5$.For the ensemble methods, Figures 1e, 1f illustrate that when $\eta = \{0, 0.5\}$ *combinedW* and *RankAgg* outperform their component algorithms for $k \lessapprox 10$. 1g demonstrates the better performance of *RankAgg* to its component algorithms for all values of k when $\eta = 1$. However, for a severe position bias $\eta = 1.5$, *combinedW* and *RankAgg* do not outperform their component algorithms for any value of k, but *RankAgg* becomes the second best algorithm. Among the ensemble methods, *RankAgg* is more robust to position bias than *combinedW*.

Our main takeaways from this experiment are:

- Under small to no position bias ($\eta = 0, 0.5$) $Heckman^{rank}$ outperforms $Propensity\ SVM^{rank}$ for both metrics.
- Under moderate position bias ($\eta = 1$), while $Heckman^{rank}$ outperforms $Propensity\ SVM^{rank}$ for ARRR, it lags behind $Propensity\ SVM^{rank}$ for nDCG@10.
- Under severe position bias ($\eta = 1.5$), $Heckman^{rank}$ falls behind $Propensity\ SVM^{rank}$ for both *ARRR* and *nDCG*@10.
- *RankAgg* performs better than $Heckman^{rank}$ for all selection bias levels and it is more robust to position bias than *combinedW*. *combinedW* surpasses $Heckman^{rank}$ under severe selection bias ($k \lessapprox 10$).



(a) ARRR, $\eta = 0$ **(b) ARRR, $\eta = 0.5$** **(c) ARRR, $\eta = 1$** **(d) ARRR, $\eta = 1.5$**

(e) nDCG@10, $\eta = 0$ **(f) nDCG@10, $\eta = 0.5$** **(g) nDCG@10, $\eta = 1$** **(h) nDCG@10, $\eta = 1.5$**

Figure 2: The performance of LTR algorithms on set 2 under 10% noisy clicks.

5.2.2 *Effect of click noise.*

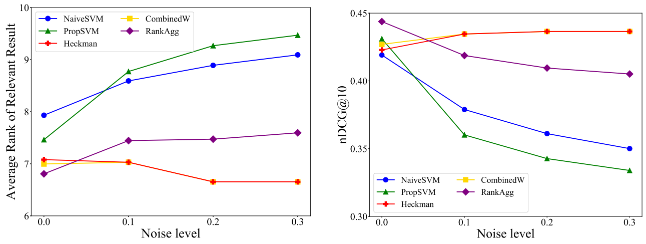
Thus far, we have considered *noiseless* clicks that are generated only over relevant documents. However, this is not a realistic assumption as users may also click on irrelevant documents. We now relax this assumption and allow for 10% of the clicked documents to be irrelevant.

When ARRR is the preferred metric, Figures 2a, 2b, 2c and 2d illustrate that $Heckman^{rank}$ outperforms $Propensity\ SVM^{rank}$ for $\eta = \{0, 0.5, 1\}$, while under higher position bias level ($\eta = 1.5$), $Heckman^{rank}$ falls behind $Propensity\ SVM^{rank}$. Comparing the noisy click performance to the noiseless one (Figures 1a, 1b, 1c), one can conclude that for $\eta = \{0, 0.5, 1\}$, $Propensity\ SVM^{rank}$ is highly affected by noise, while $Heckman^{rank}$ is much less affected. For example, Figure 2c illustrates that for $\eta = 1$, the better performance of $Heckman^{rank}$ over $Propensity\ SVM^{rank}$ is much more noticeable compared to 1c where clicks were noiseless. Interestingly, the ensembles *combinedW* nor *RankAgg*, do not outperform the most successful algorithm in the presence of noisy clicks.

When nDCG@10 is the preferred metric, one can draw the same conclusions: while $Heckman^{rank}$ is more robust to noise and outperforms $Propensity\ SVM^{rank}$ for $\eta = \{0, 0.5, 1\}$, it fails to beat $Propensity\ SVM^{rank}$ for $\eta = 1.5$. Another interesting point is that $Propensity\ SVM^{rank}$ is severely affected by noise when selection bias is high (low values of k), such that it even falls behind $Naive\ SVM^{rank}$. This exemplifies how much selection bias can degrade the performance of LTR systems if they do not correct for it.

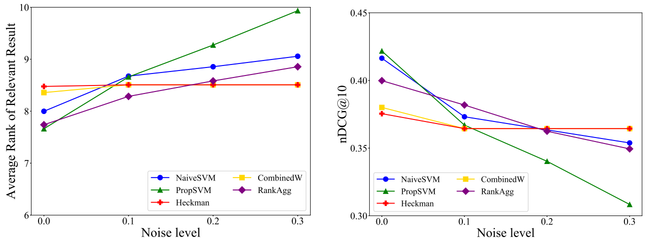
In the presence of 10% noisy clicks, the main takeaways are:

- Under severe to moderate selection bias ($k \lessapprox 15$), *Propensity* SVM^{rank} suffers a lot from the noise and it even falls behind *Naive* SVM^{rank} for both *ARRR* and *nDCG@10*.



(a) Evaluation on ARRR. **(b) Evaluation on nDCG@10.**

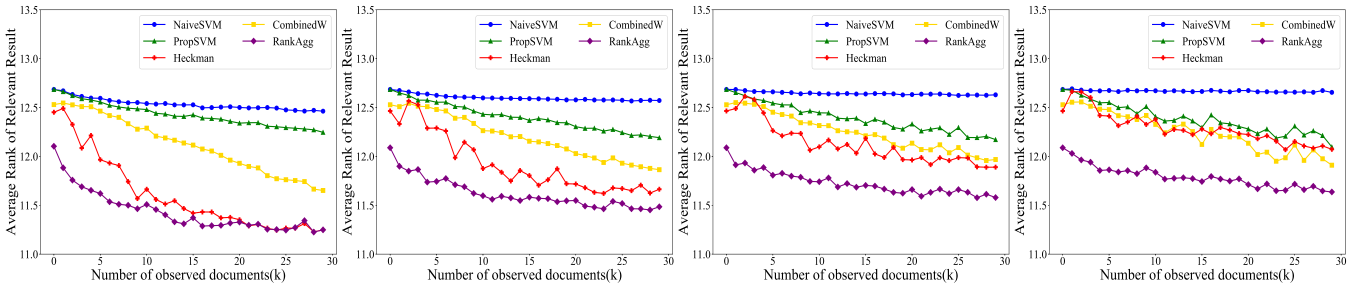
Figure 3: Effect of noisy clicks for high selection bias ($k = 10$) and moderate position bias ($\eta = 1$).



(a) Evaluation on ARRR. **(b) Evaluation on nDCG@10.**

Figure 4: Effect of noisy clicks for high selection bias ($k = 10$) and high position bias ($\eta = 2$).

- $Heckman^{rank}$ outperforms $Propensity\ SVM^{rank}$ when position bias is not severe ($\eta = \{0, 0.5, 1\}$) for both metrics.



(a) ARRR, $\eta = 0.5$ **(b) ARRR, $\eta = 1$** **(c) ARRR, $\eta = 1.5$** **(d) ARRR, $\eta = 2$**

Figure 5: The performance (ARRR) of LTR algorithms on set 1.

- Just like in the noiseless case, $Heckman^{rank}$ cannot surpass $Propensity\ SVM^{rank}$ under severe position bias ($\eta = 1.5$).
- *combinedW* and *RankAgg* surpass $Heckman^{rank}$ for a severe selection bias ($k \lessapprox 5$) when $\eta = \{0, 0.5\}$ for both *ARRR* and *nDCG*@10. However, *RankAgg* and *combinedW* cannot beat $Propensity\ SVM^{rank}$ under high position bias.

5.2.3 ***Effect of varying noise for*** $\eta = 1$ ***and*** $\eta = 2$. In this section, we investigate whether our proposed models are robust to noise. Toward this goal, we varied the noise level from 0% to 30%. Figures 3a and 3b show the performance of the LTR algorithms for different levels of noise, where $k = 10$ and $\eta = 1$. Under increasing noise, the performance of $Heckman^{rank}$ is relatively stable and even improves, while the performance of all other LTR algorithms degrades. Even $Naive\ SVM^{rank}$ is more robust to noise compared to $Propensity\ SVM^{rank}$, which is different from the results by Joachims et al. [29] where no selection bias was considered. The reason could be that their evaluation is based on the assumption that all documents have a non-zero probability of being observed, while Figure 3a and 3b are under the condition that documents ranked bellow a certain cut-off ($k = 10$) have a zero probability of being observed.

We also investigate the performance of LTR algorithms with respect to noise, when position bias is severe ($\eta = 2$). As shown in Figure 4, irrespective of metric of interest, $Heckman^{rank}$ is robust to varying noise, while the performance of all other algorithms degrades when the noise level increases. $Propensity\ SVM^{rank}$ falls behind all other algorithms in high level of noise. This implies that even though $Heckman^{rank}$ cannot surpass $Propensity\ SVM^{rank}$ when position bias is severe ($\eta = 1.5, 2$) in noiseless environments, it clearly outperforms $Propensity\ SVM^{rank}$ in the presence of selection bias with noise. This is an extremely useful property since in real world applications we cannot assume a noiseless environment.

5.3 Experimental results on set 1

To confirm the performance of our proposed methods on the larger set 1 with out-of-sample test data, we ran experiments varying position bias ($\eta = \{0.5, 1, 1.5, 2\}$) under noiseless clicks. The results on this dataset were even more promising, especially for high position bias. Figure 5 illustrates the ARRR performance of all algorithms. $Heckman^{rank}$ outperforms $PropensitySVM^{rank}$ for all position bias levels, though its strong performance decreases with increasing η. This is unlike set 2 where $Heckman^{rank}$ did not outperform $PropensitySVM^{rank}$ under high position bias. The ensemble *RankAgg* outperforms both $Heckman^{rank}$ and $PropensitySVM^{rank}$ for all position and selection bias levels, while *combinedW* outperforms $PropensitySVM^{rank}$ but does not surpass $Heckman^{rank}$. Moreover, the stronger performance of $Heckman^{rank}$ and *RankAgg* over $PropensitySVM^{rank}$ is much more pronounced compared to set 2.

6 CONCLUSION

In this work, we formalized the problem of selection bias in learning-to-rank systems and proposed $Heckman^{rank}$ as an approach for correcting for selection bias. We also presented two ensemble methods that correct for both selection and position bias by combining the rankings of $Heckman^{rank}$ and $Propensity\ SVM^{rank}$. Our extensive experiments on semi-synthetic datasets show that selection bias affects the performance of LTR systems and that $Heckman^{rank}$ performs better than existing approaches that correct for position bias but that do not address selection bias. Nonetheless, this performance decreases as the position bias increases. At the same time, $Heckman^{rank}$ is more robust to noisy clicks even with severe position bias, while $Propensity\ SVM^{rank}$ is adversely affected by noisy clicks in the presence of selection bias and even falls behind $Naive\ SVM^{rank}$. The ensemble methods, *combinedW* and *RankAgg*, outperform $Heckman^{rank}$ for severe selection bias and zero to small position bias.

Our initial study of selection bias suggests a number of promising future avenues for research. For example, our initial work considers only linear models but a Heckman-based solution to selection bias can be adapted to non-linear algorithms as well, including extensions that consider bias correction mechanisms specific to each learning-to-rank algorithm. Our experiments suggest that studying correction methods that jointly account for position bias and selection bias can potentially address the limitations of methods that only account for one. Finally, even though we specifically studied selection bias in the context of learning-to-rank systems, we expect that our methodology will have broader applications beyond LTR systems.

ACKNOWLEDGEMENTS

We thank John Fike and Chris Kanich for providing invaluable server support. We acknowledge Amazon Web Services for their support through AWS Cloud Credits for Research. This material is based on research sponsored in part by the Defense Advanced Research Projects Agency (DARPA) under contract number HR00111990114. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

REFERENCES

[1] Aman Agarwal, Kenta Takatsu, Ivan Zaitsev, and Thorsten Joachims. 2019. A General Framework for Counterfactual Learning-to-Rank. In ACM Conference on Research and Development in Information Retrieval (SIGIR).

[2] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W Bruce Croft. 2018. Unbiased Learning to Rank with Unbiased Propensity Estimation. SIGIR (2018).

[3] Elias Bareinboim and Judea Pearl. 2012. Controlling selection bias in causal inference. In AISTATS. 100–108.

[4] Elias Bareinboim and Judea Pearl. 2016. Causal inference and the data-fusion problem. Proceedings of the National Academy of Sciences 113, 27 (2016), 7345–7352.

[5] Elias Bareinboim and Jin Tian. 2015. Recovering causal effects from selection bias. In Twenty-Ninth AAAI Conference on Artificial Intelligence.

[6] Elias Bareinboim, Jin Tian, and Judea Pearl. 2014. Recovering from Selection Bias in Causal and Statistical Inference.. In AAAI. 2410–2416.

[7] Alexey Borisov, Ilya Markov, Maarten de Rijke, and Pavel Serdyukov. 2016. A neural click model for web search. In Proceedings of the 25th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 531–541.

[8] Òscar Celma and Pedro Cano. 2008. From hits to niches?: or how popular artists can bias music recommendation and discovery. In Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition. ACM, 5.

[9] Allison JB Chaney, Brandon M Stewart, and Barbara E Engelhardt. 2018. How Algorithmic Confounding in Recommendation Systems Increases Homogeneity and Decreases Utility. RecSys (2018).

[10] Olivier Chapelle and Yi Chang. 2011. Yahoo! learning to rank challenge overview. In Proceedings of the Learning to Rank Challenge. 1–24.

[11] Olivier Chapelle, Thorsten Joachims, Filip Radlinski, and Yisong Yue. 2012. Large-scale validation and analysis of interleaved search evaluation. ACM Transactions on Information Systems (TOIS) 30, 1 (2012), 6.

[12] Olivier Chapelle and Ya Zhang. 2009. A dynamic bayesian network click model for web search ranking. In Proceedings of the 18th international conference on World wide web. ACM, 1–10.

[13] Li Chen, Marco De Gemmis, Alexander Felfernig, Pasquale Lops, Francesco Ricci, and Giovanni Semeraro. 2013. Human decision making and recommender systems. ACM Transactions on Interactive Intelligent Systems (TiiS) 3, 3 (2013), 1–7.

[14] Juan D Correa and Elias Bareinboim. 2017. Causal effect identification by adjustment under confounding and selection biases. In Thirty-First AAAI Conference on Artificial Intelligence.

[15] Juan D Correa, Jin Tian, and Elias Bareinboim. 2018. Generalized adjustment under confounding and selection biases. In AAAI.

[16] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An experimental comparison of click position-bias models. In WSDM. ACM, 87–94.

[17] Zhao Dan-Dan, Zeng An, Shang Ming-Sheng, and Gao Jian. 2013. Long-term effects of recommendation on the evolution of online systems. Chinese Physics Letters 30, 11 (2013), 118901.

[18] Pranav Dandekar, Ashish Goel, and David T Lee. 2013. Biased assimilation, homophily, and the dynamics of polarization. Proceedings of the National Academy of Sciences 110, 15 (2013), 5791–5796.

[19] Cynthia Dwork, Ravi Kumar, Moni Naor, and Dandapani Sivakumar. 2001. Rank aggregation methods for the web. In Proceedings of the 10th international conference on World Wide Web. ACM, 613–622.

[20] Daniel M Fleder and Kartik Hosanagar. 2007. Recommender systems and their impact on sales diversity. In Proceedings of the 8th ACM conference on Electronic commerce. ACM, 192–199.

[21] James Heckman. 1979. Sample Selection Bias as a Specification Error. Econometrica 47, 1 (1979), 153–161.

[22] José Miguel Hernández-Lobato, Neil Houlsby, and Zoubin Ghahramani. 2014. Probabilistic matrix factorization with non-random missing data. In International Conference on Machine Learning. 1512–1520.

[23] Katja Hofmann, Anne Schuth, Shimon Whiteson, and Maarten de Rijke. 2013. Reusing historical interaction data for faster online learning to rank for IR. In Proceedings of the sixth ACM international conference on Web search and data mining. ACM, 183–192.

[24] Ziniu Hu, Yang Wang, Qu Peng, and Hang Li. 2019. Unbiased LambdaMART: An Unbiased Pairwise Learning-to-Rank Algorithm. (2019).

[25] Rolf Jagerman, Harrie Oosterhuis, and Maarten de Rijke. 2019. To Model or to Intervene: A Comparison of Counterfactual and Online Learning to Rank from User Interactions. (2019).

[26] Lilli Japec, Frauke Kreuter, Marcus Berg, Paul Biemer, Paul Decker, Cliff Lampe, Julia Lane, Cathy O'Neil, and Abe Usher. 2015. Big data in survey research: Aapor task force report. Public Opinion Quarterly 79, 4 (2015), 839–880. https://doi.org/10.1093/poq/nfv039

[27] Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 133–142.

[28] Thorsten Joachims, Laura A Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. Accurately interpreting clickthrough data as implicit feedback. In Sigir, Vol. 5. 154–161.

[29] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased learning-to-rank with biased feedback. In WSDM. ACM, 781–789.

[30] David Lazer, Ryan Kennedy, Gary King, and Alessandro Vespignani. 2014. The parable of Google Flu: traps in big data analysis. Science 343, 6176 (2014), 1203–1205.

[31] Shili Lin. 2010. Rank aggregation methods. Wiley Interdisciplinary Reviews: Computational Statistics 2, 5 (2010), 555–570.

[32] Tie-Yan Liu. 2011. Learning to rank for information retrieval. Springer Science & Business Media.

[33] Harrie Oosterhuis and Maarten de Rijke. 2018. Differentiable unbiased online learning to rank. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management. ACM, 1293–1302.

[34] Judea Pearl, Madelyn Glymour, and Nicholas P Jewell. 2016. Causal inference in statistics: A primer. John Wiley & Sons.

[35] 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.

[36] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In WWW. ACM, 521–530.

[37] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as treatments: Debiasing learning and evaluation. arXiv preprint arXiv:1602.05352 (2016).

[38] Anne Schuth, Harrie Oosterhuis, Shimon Whiteson, and Maarten de Rijke. 2016. Multileave gradient descent for fast online learning to rank. In Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. ACM, 457–466.

[39] Anne Schuth, Floor Sietsma, Shimon Whiteson, Damien Lefortier, and Maarten de Rijke. 2014. Multileaved comparisons for fast online evaluation. In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. ACM, 71–80.

[40] Andrew Smith and Charles Elkan. 2004. A Bayesian network framework for reject inference. KDD (2004). http://delivery.acm.org/10.1145/1020000/1014085/p286-smith.pdf

[41] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to rank with selection bias in personal search. In SIGIR. ACM, 115–124.

[42] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position bias estimation for unbiased learning to rank in personal search. In WSDM. ACM, 610–618.

[43] Yixin Wang, Dawen Liang, Laurent Charlin, and David M Blei. 2018. The deconfounded recommender: A causal inference approach to recommendation. arXiv preprint arXiv:1808.06581 (2018).

[44] Yisong Yue and Thorsten Joachims. 2009. Interactively optimizing information retrieval systems as a dueling bandits problem. In Proceedings of the 26th Annual International Conference on Machine Learning. ACM, 1201–1208.

[45] Bianca Zadrozny. 2004. Learning and evaluating classifiers under sample selection bias. In Proceedings of the twenty-first international conference on Machine learning. ACM, 114.