# Double Machine Learning for Selection Bias Recovery in Learning-to-rank Systems

Md Aminul Islam
University of Illinois at Chicago

## ABSTRACT

Implicit feedback from users such as click-through data in personalized recommender systems is an important resource to infer latent user interests, and improve ranking quality in learning-to-rank(LTR) systems. These implicit feedback data can be biased, and using such biased data to train LTR systems can lead to suboptimal performance in ranking relevant results. One type of bias is position bias which happens when higher ranked items get more user interactions than the items of lower ranking. Most of the recent methods for bias correction have focused on position bias. Another important type of bias is selection bias. Selection bias occurs when recommender systems choose to show a truncated list of items from all the relevant results for a specific query. There has been relatively little focus on addressing selection bias in LTR systems. Here, we propose a counterfactual-based ranking approach that leverages the principles of the Double Machine Learning(DML) econometrics method to mitigate selection bias in LTR systems. Experimental result shows that our method provides better or similar accuracy than most of the existing unbiased LTR algorithms, and performs better than the existing algorithms when selection bias severity is very high.

## 1 INTRODUCTION

In this era of digital technology, users are overwhelmed by the abundance of information. Recommender systems prioritize information that is most relevant to users based on user preferences. This alleviates information overload, and makes it easier for users to interact with items that are of interest to them. Learning-to-rank(LTR) system is a technique that is commonly used in recommender systems, news feed personalization to rank a list of items based on their relevance in the request of a specific query. The objective of the LTR system is to learn a ranking function using labeled training data with relevance labels for each document. This ranking function takes a list of documents with a specific query as input and predicts relevance scores for each document. The documents are then sorted based on their relevance where higher relevant items appear first in the ranked documents. The standard way of training the LTR systems using explicit relevance data such as rating or human-annotated labels. It is expensive, and often impossible for humans to manually annotate every possible document. Another way is to train the LTR systems by using implicit feedback data from users(e.g., clicks). This type of data is easier to collect, and most of the recommender systems use this implicit feedback data to rank relevant items for a query. However, implicit feedback data is innately biased and can be noisy too. Future training of LTR systems using these biased data results in poor performance for ranking relevant items.

The most common type of bias in the LTR systems is position bias which arises when higher-ranked items are more likely to be clicked irrespective of their relevancy. The items that appear first in the ranked documents have a higher probability of being observed and clicked. Most of the position bias correction methods assume that all the relevant items have a non-zero probability of being observed and clicked. The LTR systems get increased interactions with top-ranked items which eventually leads to the LTR systems prioritizing those items more. As a result, the choice for a diverse set of related recommendations is narrowed which can lead to user dissatisfaction. Position bias correction algorithms try to boost those items that are in relatively lower positions so that users can interact with other related items in response to a query.

If the list of relevant items is large, recommender systems sometimes show the top $k$ relevant results from all the relevant results for a query. As a result, some relevant items are being cut, and may not be observed and clicked by users even if they are relevant to that query. The items that appear below $k$ have zero probability of being observed and getting user interactions. This type of bias is called selection bias in the LTR systems. The selection bias correction algorithm assumes that the items that are within the top $k$ positions have an equal probability of observation, and items that appear below $k$ have zero probability of observation[16, 17]. Users never a get chance to observe and interact with items below the top $k$ relevant items for a query. As a consequence, the option for users to interact with related and different sets of recommendations for a specific query becomes limited. This reinforces the users to limit themselves to a limited number of items for a query, and this can cause users disappointment. It is also important to correct this selection bias so that users get a chance to interact with the relevant items that would have been cut off without selection bias correction.

The objective of this project is to propose a counterfactual-based ranking algorithm that will correct for selection bias in LTR systems so that the ranking of items in LTR systems can recover from selection bias. The goal is to find a ranking function that will rank items based on item features and the effect of selection bias on clicks. Including the effect of selection or observation in the ranking function can recover from selection bias. To determine the unbiased effect of selection on click, we will use Double Machine Learning(DML)[16] which can separate the effect of selection or observation on click output from other covariates using the concept of orthogonalization. The most prominent work for correcting selection bias has been proposed by Ovaisi et al. [16] that uses Heckman's two-stage econometrics method which considers the relationship between the item features and observation, and between item features and click output is linear. The main advantage of DML over the Heckman-based method is that DML does not need to consider these relationships to be linear for estimating causal effects. Also, DML provides a better unbiased estimate of causal effect by overcoming the problem of regularization bias than the

traditional statistical methods(e.g. OLS, LASSO). DML uses sample splitting of data to overcome the issue of overfitting for estimating the causal effects. To the best of our knowledge, no prior work has used DML in the context of bias recovery in LTR systems.

## 2 RELATED WORK

Most works of unbiased learning-to-rank have focused on position bias[1, 2, 4, 8, 9, 12–15, 19, 22, 24], where higher ranked items get more user interactions than the items of lower ranks. Most of the position bias correction approaches for LTR systems are two-staged methods. These approaches involve the computation of click bias through a propensity model, and incorporating the click bias effect into a ranking function to rank relevant items. Propensity score is the probability of observation or treatment of a particular item given its features. The most common approach for propensity estimation is Inverse Propensity Weighting(IPW)[2, 6, 8, 11, 12, 20, 24]. IPW approach uses the propensity score of each item given its position, and re-weights the relevancy of each item. Joachims et al.[12] proposed $PropensitySVM^{rank}$ to address position bias by enhancing standard $SVM^{rank}$[10] with an Inverse Propensity Score through an unbiased estimator when dealing with biased feedback data, and incorporated this estimator into $SVM^{rank}$ to rank documents. $PropensitySVM^{rank}$ surpassed the performance of a standard $SVM^{rank}$ by effectively correcting for position bias in the learning process. Agarwal et al.[1] proposed $nDCGSVM^{rank}$ that performs better than IPW approaches when position bias is severe. Ai et al.[2] presented a Dual Learning Algorithm(DLA) for position bias correction that concurrently learns an unbiased propensity model and ranking model from biased click data without any preprocessing. Vardasbi et al.[22] proposed a counterfactual-based mixture model for position bias correction where the distribution of the click-through-rate(CTR) data is a mixture of relevant and non-relevant items. Search result randomization on a small percentage of production traffic can affect users' search experience. Wang et al.[24] developed an Expectation-Maximization (EM) position bias debiasing algorithm from regular click data without relying on search result randomization. Ren et al.[19] proposed a pairwise position bias debiasing algorithm that can mitigate the position bias effect for both continuous and categorical values in LTR systems. More recently, Oosterhuis[14] introduced Doubly-Robust Estimator(DRE) to correct for position bias which is more robust than the existing IPW approaches and shows optimal performance with fewer datapoints during convergence.

Recommender systems sometimes show a truncated list of items to users if the list of relevant items for a query is very large. As a result, some items are being cut off. The items that are getting cut off are not being observed by users. These items have zero probability of being observed and clicked by users even if they are relevant to the specific query. This leads to selection bias in LTR systems[16, 16]. In personal search engines, click data is highly sparse, and most of the click models find it difficult to incorporate this sparse click data. Wang et al.[23] addressed the problem of selection bias in personal search, and proposed a query-dependent selection bias correction method from sparse click data. Hernández-Lobato et al.[7] and Schnabel et al.[21] proposed probabilistic matrix-factorization methods when data is missing not-at-random. The most recent prominent work for correcting selection bias has been proposed by Ovaisi et al. [16] through a counterfactual-based solution for estimating the probability of a document under selection bias. The counterfactual problem was framed as a document that would have been clicked if it had been observed. For estimating the unbiased click counterfactual, Ovaisi et al. used $Heckman^{rank}$ two-stage bivariate econometrics method which corrects for selection bias through inverse mills ratio(IMR), and incorporated this IMR in the ranking function to calculate the click probabilities of all documents. However, Ovaisi et al. considered the relationship between the item features and observation, and between item features and click output to be linear when calculating the IMR. The main advantage of our Double Machine Learning[5] based selection bias correction approach is that this method does not need to consider these relationships to be linear, and can capture the non-linearity in data when measuring the effect of selection on click output. Therefore, we will compare our DML-based correction method with [16].

Joint debiasing that corrects for both position and selection bias has also been studied[16, 17] to a little extent. Ovaisi et al.[16] developed an ensemble approach that incorporates $PropensitySVM^{Rank}$[12] and $Heckman^{rank}$ correction [16] method to correct for joint debiasing of position and selection bias. It is not always feasible to calculate propensity scores to estimate the position bias effect. Ovaisi et al.[17] developed a propensity-independent debiasing method that corrects for both position and selection bias by leveraging the base ranker ranking as an additional covariate while calculating IMR. This method helps in debiasing position bias without calculating the propensity scores of an item explicitly.

## 3 PROBLEM DESCRIPTION

We first review the formal definition of learning-to-rank systems following Joachims et al.[12]. In traditional LTR systems, we are given a sample $\mathbf{x}$ consisting of i.i.d. queries $\mathbf{x}_i$, sampled from the distribution $P(\mathbf{x})$ while assuming complete knowledge of the relevances $rel(\mathbf{x}, y)$ for all documents $y$ associated with these queries. This type of setting is called the Full-Information Setting as the relevances are known. If $\Delta(y|\mathbf{x}_i)$ is the loss of any ranking $y$ for query $\mathbf{x}_i$, the overall risk of ranking system $S$ that returns ranking $S(\mathbf{x})$ for queries $\mathbf{x}$ can be defined as

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

The objective of the learning is to find a ranking function $S \in \boldsymbol{S}$, which aims to minimize the *empirical risk* $\hat{R}(S)$

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

$rel(\mathbf{x}_i, y)$ denotes the true relevancy of a document $y$ for the query $\mathbf{x}_i$. This can be obtained via human-annotated labels but it is unrealistic, and often impossible for humans to annotate every possible document given the queries. One alternative way is to use implicit feedback data from users(e.g. clicks). The implicit feedback data can be used as a proxy for the relevancy of documents given a query. Although these data are easier to collect, they can be noisy and suffer from biases such as position bias and selection bias.

Recommender systems choose to show the only top $k$ results if the list of relevant results for a query is large. As a result, some

relevant items are being cut off, and users do not have a chance to observe and click on them even if they are relevant to the query. The focus of this work is to propose a ranking algorithm that will correct the selection bias in LTR systems so that future training of LTR systems can recover from selection bias.

We frame the task of ranking documents as a counterfactual problem[18]. In our case, the outcome is the probability of click *counterfactual* $C_{\boldsymbol{x},y}$ for the document $y$ under with features $\boldsymbol{x}$. The variable $C_{\boldsymbol{x},y}$ signifies the *counterfactual* observation of a click, indicating whether a document $y$ would have been clicked if it had been observed in response to query $\mathbf{x}$ with features $\boldsymbol{x}$. The intuition to use *counterfactual* is that we want to estimate the probability of clicks for all documents irrespective of their observation status. We can define the click outcome as a linear regression equation form.

$$C_{\boldsymbol{x},y} = \alpha \boldsymbol{F}_{\boldsymbol{x},y} + \varepsilon_c, \tag{3}$$

where $\boldsymbol{F}_{\boldsymbol{x},y}$ represents the features of document $y$, and $\varepsilon_c$ is a normally distributed error term. However, predicting click probability using equation (3) will be biased as it does not consider the effect of selection on click output. Therefore, we need to include the effect of document selection when calculating the click probability of a document. Let $O_{\boldsymbol{x},y} \in \{0, 1\}$ indicate the observation status of a document given the query $\mathbf{x}$ which is treatment variable indicating whether users have observed the document or not. Now, we can redefine the click outcome as a linear regression equation form considering the effect of selection on click output.

$$C_{\boldsymbol{x},y} = \theta O_{\boldsymbol{x},y} + \alpha \boldsymbol{F}_{\boldsymbol{x},y} + \varepsilon_c, \tag{4}$$

where $\theta$ is the effect of selection on click. Equation will predict the unbiased click probability as it takes into consideration the effect of selection on click output. However, one potential issue of this equation (3) is that the relationship between features and click output is assumed to be linear. In most cases, this relationship may not be linear, and our estimation of the causal effect $\theta$ will produce an incorrect result in that case. The challenge is that we need to remove this linear relationship between the features and the click output. If we can simply remove $\alpha \boldsymbol{F}_{\boldsymbol{x},y}$ this linear form with a generic functional form $g(\boldsymbol{x})$, we can redefine the (3) as

$$C_{\boldsymbol{x},y} = \theta O_{\mathbf{x},y} + g(\boldsymbol{x}) + \varepsilon_c, \tag{5}$$

where $g(\boldsymbol{x})$ represents a functional form of the features of the document $y$ with features $\boldsymbol{x}$. We do not need to assume a functional form of $g(\boldsymbol{x})$. It can be parametric, non-parametric or semi-parametric. The main challenge is now to calculate the causal effect $\theta$ with the presence of unknown functional form $g(\boldsymbol{x})$. Chernozhukov et al.[5] developed an econometric method Double Machine Learning(DML) which can estimate the causal effect with the functional form given in equation (5). Therefore, we will use DML to estimate the causal effect of selection or observation on click. After calculating the causal effect of selection on click using DML, we will incorporate this effect into the ranking function to rank relevant items which will be an unbiased ranking function in the context of selection bias.

# 4 SELECTION BIAS CORRECTION

In this section, we will describe the procedures of calculating the causal effect of observation or selection on click output, and incorporating the effect into the ranking function for an unbiased learning of the ranking function. Therefore, our tasks are twofold. First, we will calculate the causal effect of selection on click, and then leverage this effect into the ranking function.

## 4.1 Causal Effect Estimation with Double Machine Learning

Chernozhukov et al.[5] developed the Double Machine Learning(DML) framework, an econometric method to estimate unbiased causal effects. DML has two regression models: the outcome model and the treatment model. The outcome model is used to estimate the probability of the outcome variable given the treatment variables and covariates. If we model DML in the context of LTR, the outcome is the probability of click $C_{\boldsymbol{x},y}$ for the document $y$ under query $\mathbf{x}$ with feature $\boldsymbol{x}$. $O_{\boldsymbol{x},y}$ is the treatment variable which indicate whether the document $y$ have been observed or not. We can define the DML outcome model in the context of LTR as

$$C_{\boldsymbol{x},y} = \theta O_{\boldsymbol{x},y} + g(\boldsymbol{x}) + \varepsilon_c, \tag{6}$$

which is identical to the equation (5).

DML has another model called the treatment model. The treatment model is used to estimate the probability of an individual receiving the treatment based on a set of covariates. If we model the DML treatment equation with the context of LTR, the equation for the treatment model can be written as

$$O_{\boldsymbol{x},y} = m(\boldsymbol{x}) + \varepsilon_o \tag{7}$$

where $m(\boldsymbol{x})$ represents a functional form of the features of document $y$ with features $\boldsymbol{x}$ upon which the treatment assignment depends, and $\varepsilon_o$ is a normally distributed error term. If $\boldsymbol{x}$ is low dimensional, we can estimate $\theta$ by controlling for $\boldsymbol{x}$. In this case, running Ordinary Least Square(OLS) is enough to get the causal effect $\theta$. However, if $\boldsymbol{x}$ is high dimensional which is common in LTR systems, there is much possibility that there exists colinearity between features $\boldsymbol{x}$. As a result, OLS will produce an incorrect estimate of the coefficients of the feature terms. We can use machine learning techniques to overcome this by adding regularization terms. However, regularization will push some coefficients to zero or close to zero. As a result, we are eliminating some controls. This can be good for predictions. Eliminating some controls that can be correlated with $\theta$ will result in biased causal effect estimation. This introduces regularization bias in causal effect estimation. That is why machine learning is good for prediction but may not be good for causal effect estimation. To overcome the problems of colinearity in high dimensional settings and regularization bias, DML uses the following two equations to predict $C_{\boldsymbol{x},y}$ and $O_{\boldsymbol{x},y}$.

$$C_{\boldsymbol{x},y} = g(\boldsymbol{x}) + \varepsilon_c, \tag{8}$$

$$O_{\boldsymbol{x},y} = m(\boldsymbol{x}) + \varepsilon_o \tag{9}$$

We will first predict $C_{\boldsymbol{x},y}$ based on features $\boldsymbol{x}$. In equation (8), $\theta$ is not included that means we will predict $C_{\boldsymbol{x},y}$ for document $y$ without the effect of $\theta$. In equation (9), we will predict $O_{\boldsymbol{x},y}$ for document $y$ based on features $\boldsymbol{x}$. We do not need to assume any functional form of $g(\boldsymbol{x})$ and $m(\boldsymbol{x})$. They can be parametric, semi-parametric,

or non-parametric. We can train any regression-based machine learning algorithm for predicting $C_{\boldsymbol{x},y}$ and $O_{\boldsymbol{x},y}$ from equation (8) and equation (9). DML is modeled in this way to allow non-linearity between outcome and features/covariates, and between treatment and features/covariates, and allows for regularization between features/covariates.

We can calculate the residuals from the predicted output of equation (8) as

$$\widetilde{C}_{\boldsymbol{x},y} = C_{\boldsymbol{x},y} - \hat{g}(\boldsymbol{x}) \tag{10}$$

where $\widetilde{c}_{\boldsymbol{x},y}$ denotes the residual of a document $y$ with feature $\boldsymbol{x}$ between the actual click output and the predicted value. This residual is a measure of partialling out the effect of features from the outcome variable. In other words, this residual is the parts of $\widetilde{C}_{\boldsymbol{x},y}$ that are orthogonal to the features $X$. This residual is the error term $\varepsilon_c$ in (8). Similarly, we can calculate the residuals from the predicted output of equation (9) as

$$\widetilde{O}_{\boldsymbol{x},y} = O_{\boldsymbol{x},y} - \hat{m}(\boldsymbol{x}) \tag{11}$$

where $\widetilde{O}_{\boldsymbol{x},y}$ denotes the residual of a document $y$ given features $\boldsymbol{x}$ between the actual observed value and the predicted value. This residual is a measure of separation of the effect of features from the treatment variable. In other words, this residual is the parts of $\widetilde{O}_{\boldsymbol{x},y}$ that are orthogonal to the features $X$. This residual is the error term $\varepsilon_o$ in (9).

DML uses a linear regression equation to estimate the causal effect $\theta$ using the residuals from equation (10) and equation (11) as

$$\widetilde{C} = \theta\widetilde{O} + \varepsilon. \tag{12}$$

The reason why (12) works for calculating the causal effect $\theta$ is described by Frisch–Waugh–Lovell (FWL)Theorem[3]. The FWL theorem states that regressing a variable on all covariates in a multiple regression equation will yield the same result as regressing the residual variables on the residualized subset of covariates. Another important aspect of getting the unbiased causal estimate is to use sample splitting in (10) and equation (11). Sample splitting is similar to $k$-fold cross-validation. This includes dividing the dataset into $k$-folds, training of different subsets using $k-1$ folds, estimating the causal effect on the remaining portion of the data, and finally averaging the causal effect of $k$ different folds. This can be written as

$$\theta_{DML,CF} = \frac{1}{K}\sum_{k\in K}^{n}\theta_k \tag{13}$$

Chernozhukov et al.[5] shows that sample splitting and orthogonalization techniques produce an unbiased normal estimate of causal effect. Using equation (12), sample splitting, and regularization technique, Chernozhukov et al.[5] shows that $\theta$ can directly be computed as

$$\theta = \frac{\frac{1}{n}\sum_{y=1}^{n}(\widetilde{O}_y\widetilde{C}_y)}{(\frac{1}{n}\sum_{y=1}^{n}\widetilde{O}_y O_y)}. \tag{14}$$

For estimating the average treatment effect $\theta$ in equation (14), this $(\frac{1}{n}\sum_{y=1}^{n}\widetilde{O}_y O_y)^{-1}\widetilde{O}_y\widetilde{C}_y$ term for each document $y$ is contributing to measure the average treatment effect over all documents. In other words, we can say that this term is individualized treatment effect for document $y$, and averaging over the individualized treatment effect for all documents, the equation (14) can be expressed. If we denote this term as $ITE_y$ for document $y$, it can be expressed as

$$ITE_y = \frac{\widetilde{O}_y\widetilde{C}_y}{(\frac{1}{n}\sum_{y=1}^{n}\widetilde{O}_y O_y)}. \tag{15}$$

## 4.2 Unbiased Ranking Function

For unbiased ranking function estimation, we need to incorporate the effect of selection on click while training the ranking function. At this point, we can incorporate the individualized treatment effect $ITE_y$ into the ranking function for unbiased ranking function learning. This can be done either two ways. We can update the relevance label of each document by subtracting the individualized treatment effect $ITE_y$ from the relevance labels, and then train the ranking function. Also, we can add the individualized treatment effect $ITE_y$ as an additional feature for document $y$. In our implementation, we follow the latter one. Adding $ITE_y$ as an additional feature for document $y$, we can define our final unbiased ranking function as

$$C_{\boldsymbol{x},y} = f^{unbiased}(\boldsymbol{x}, ITE_y) + \varepsilon. \tag{16}$$

This final ranking function takes the features of each document and the individualized treatment effect calculated from DML for each document as input to predict the click probability of each document. It is unbiased in the context of selection bias as it takes into consideration of the causal effect of observation on click for each document.

# 5 EXPERIEMENTS

## 5.1 Dataset

To evaluate the LTR model performance, we use set2 of the C14-Yahoo! Learning to Rank Challenge dataset. Set 2 contains $1,266$ train queries and $34,815$ train documents. Each document has a 700-dimensional feature vectors whose values are normalized $\in \{0, 1\}$. The dataset contains true relevance of ranking $\in [0, 5]$ by human-annotated relevance labels where 0 means not relevant and 5 means most relevant. One of the main reasons to use this dataset is that it has true unbiased ground truth relevance labels with which we can evaluate our model performance. We binarize the relevance label $\in \{0, 1\}$ following Joachims et al. [12] where relevance labels $\in [0, 2]$ and $\in [3, 5]$ correspond to 0 and 1 respectively. We perform our experiments on the training set of set 2 dataset. Within this set, we use random sampling to designate 70% of the queries for training data and allocate the remaining 30% for testing. This partition enables us to train LTR algorithms on one segment and evaluate their performance on the other.

## 5.2 Synthetic Biased Click Data Generation

We follow the data-generation procedure outlined by Joachims et al. [12]. We train a base ranker $SVM^{rank}$ using only 1% of the training dataset, which consists of true relevances for each $< query, document >$ pair. Subsequently, we use the trained model to generate rankings for the remaining 99% of the queries within the training dataset. In the next phase, we generate biased click data for the ranked documents of the training dataset. If $C_{\boldsymbol{x},y}$ and $r_i(y)$ denote whether a document $y$ is clicked and relevant respectively, and $rank(y|\overline{y})$ represents the ranking of document $y$ for

query $\mathbf{x}$ if the user was presented with the ranking $\overline{y}$, then the click probability of document $y$ for a given query $\mathbf{x}$ is determined as $P(C_{\boldsymbol{x},y} = 1) = \frac{r_i(y)}{(rank(y|\overline{y})^{\eta}}$. Here, $\eta$ signifies the severity of position bias.

## 5.3 Synthetic Document Selection Data Generation

We need to determine whether a document has been observed by users or not. For that, we need to generate the observation status $\in \{0, 1\}$. We follow Ovaisi et al. [16] to generate the observation status either observed or not observed based on the selection bias cutoff level $k$ for each $< query, document >$ pair. For example, if the selection bias cutoff level is 10, we set the observation status 1 for the first 10 documents of each query, and the observation status for the rest of the documents of each query has been set to 0.

## 5.4 Evaluation Metrics

We evaluate our performance with three different metrics commonly used for performance evaluation on ranking algorithms in LTR systems.

*5.4.1 Normalized Discounted Cumulative Gain.* *Normalized Discounted Cumulative Gain(nDCG)* is the most common metric for the performance evaluation of the LTR systems. *nDCG* evaluates the effectiveness of a ranked list for each query by taking into account the relevance of the items and their respective positions within the list. *Normalized Discounted Cumulative Gain, nDCG@p* is defined as

$$nDCG@p = \frac{DCG@p}{IDCG@p}$$

where $p$ is the position up to which we are interested to evaluate the performance of the ranking algorithm. *DCG@p* refers to *Discounted Cumulative Gain* upto position $p$. *DCG@p* is defined as

$$DCG@p = \sum_{i=1}^{p} \frac{2^{rel(\boldsymbol{x},y)} - 1}{log_2(i+1)}$$

where $i$ is the position of the document $y$ on the ranked list and $rel(\boldsymbol{x}, y)$ is the relevance of the document $y$ on the ranked list. *IDCG@p* refers to *Idealized Discounted Cumulative Gain* upto position $p$. *IDCG@p* is defined as

$$IDCG@p = \sum_{i=1}^{REL@p} \frac{2^{rel(\boldsymbol{x},y)} - 1}{log_2(i+1)}$$

where *REL@p* represents the list of relevant documents in a query up to position $p$ in the ranking, ordered by their relevance. The range of *nDCG@p* is $[0, 1]$. A higher *nDCG* value indicates that the algorithm is more effective in ranking the relevance of the items. In our experiments, we are interested in measuring the performance up to the first 10 elements of each query. Therefore, we use $p = 10$.

*5.4.2 Mean Reciprocal Rank.* *Mean Reciprocal Rank(MRR)* is another metric for evaluating the performance of the ranking algorithm. *MRR* is the average of the reciprocal ranks of the first relevant item of each query. *MRR* can be defined as

$$MRR = \frac{1}{|\mathbf{x}|} \sum_{i=1}^{|\mathbf{x}|} \frac{1}{rank_i}$$

where $rank_i$ refers to the rank position of the first relevant document for the $i$-th query. The range of *MRR* is $[0, 1]$. A higher *MRR* value means that the algorithm is more effective in ranking the first relevant items in the queries.

*5.4.3 Average Rank of Relevant Results.* *Average Rank of Relevant Results(ARRR)* provides a measure of how quickly relevant items are found on average in a set of queries. *ARRR* is defined as:

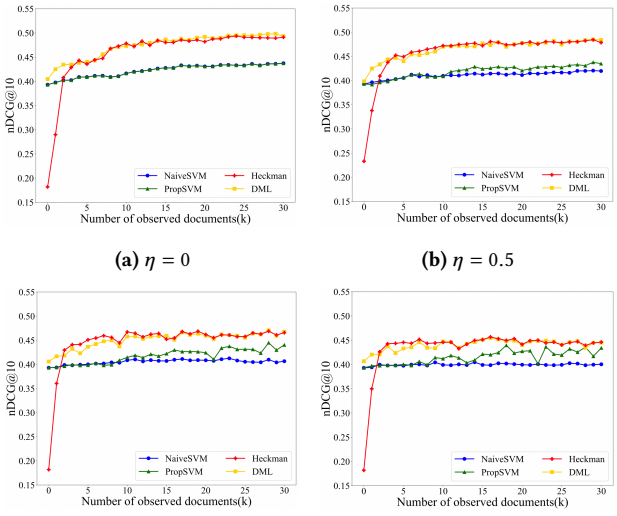
$$ARRR = \frac{\sum_{y:o_i=1 \wedge r_i=1} rank(y|\overline{y})}{|\mathbf{x}|}.$$

A lower *ARRR* indicates that, on average, relevant items are positioned closer to the top of the ranked lists which means more effective performance of the ranking algorithm in LTR systems.

## 5.5 Evaluation

We compare our DML-based LTR selection bias correction method with $SVM^{rank}$ [10], $PropensitySVM^{rank}$ [12], and $Heckman^{rank}$ [16] LTR algorithms. $SVM^{rank}$ is the standard ranking algorithm without bias correction for ranking relevant items. $PropensitySVM^{rank}$ deals with position bias correction and $Heckman^{rank}$ deals with selection bias correction in LTR systems. For calculating the individualized treatment effect using DML, we use the Support Vector Machine with Gaussian kernel, and for ranking documents using features of the documents and individualized treatment effect, we use Logistic Regression. We evaluate performance on the *Normalized Discounted Cumulative Gain nDCG@p* metric where $p$ is the rank position up to which we are interested in evaluating. We also evaluate the performance on *Mean Reciprocal Rank(MRR)* and *Average Rank of Relevant Results(ARRR)* metrics. The experimental result shows that our method outperforms $SVM^{rank}$ and $PropensitySVM^{rank}$ for different position bias and selection bias severity. It also outperforms $Heckman^{rank}$ when selection bias severity is high($1 <= k <= 5$) and position bias severity is low($\eta = 0, \eta = 0.5$). This can be particularly useful when recommender systems show only a limited number of relevant results or on the landing page of a recommender system where only a limited number of items are shown. In the case of higher selection bias and high position bias, both the $Heckman^{rank}$ and DML-based method perform almost similarly. We notice that the performance of $Heckman^{rank}$ and our DML-based method begins to decrease with the increase of the position bias severity. We can summarize our main takeaways as follows:

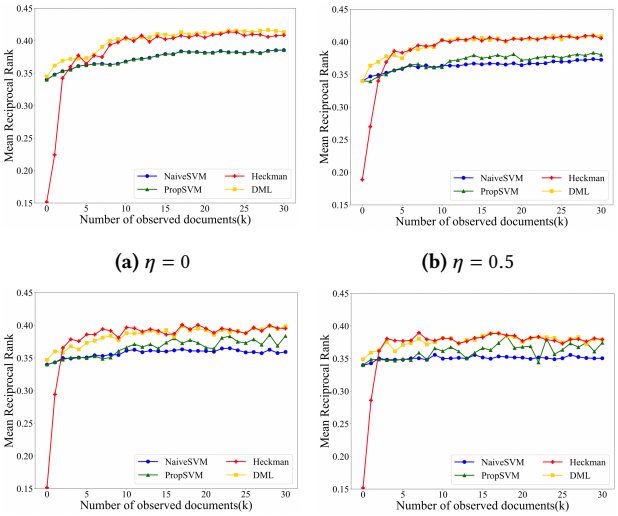
- DML-based method performs better than $SVM^{rank}$ and $PropensitySVM^{rank}$.
- DML-based method outperforms when high selection bias is high($1 <= k <= 5$) and position bias is low($\eta = 0, \eta = 0.5$).
- DML-based method and $Heckman^{rank}$ perform almost identically with high selection bias($6 <= k <= 30$) or high position bias($\eta = 1, \eta = 1.5$).
- The performance of $Heckman^{rank}$ and the DML-based method decreases with the increase of the position bias severity.




(a) $\eta = 0$ (b) $\eta = 0.5$

(c) $\eta = 1$ (d) $\eta = 1.5$

**Figure 1: The performance of LTR algorithm on set 2 for nDCG@10 metric(higher is better)**



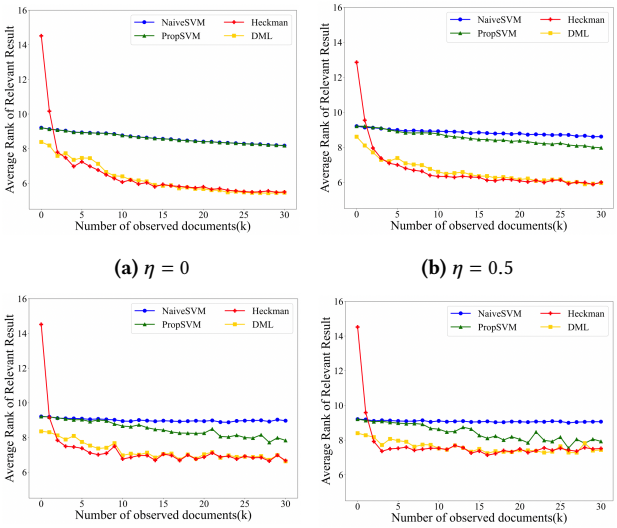

(a) $\eta = 0$ (b) $\eta = 0.5$

(c) $\eta = 1$ (d) $\eta = 1.5$

**Figure 2: The performance of LTR algorithm on set 2 for MRR metric(higher is better)**

## 6 DISCUSSION

We propose a selection bias correction algorithm that aims to correct for selection bias in learning-to-rank systems. We use Double Machine Learning, an econometric method, which can estimate causal effect in high dimensional settings, and when the relationships between the treatment and covariates, and between the covariates and outcome are not linear. We use double machine learning to estimate the effects of selection on click output, and incorporate this selection bias effect into the ranking function to




(a) $\eta = 0$ (b) $\eta = 0.5$

(c) $\eta = 1$ (d) $\eta = 1.5$

**Figure 3: The performance of LTR algorithm on set 2 for ARRR metric(lower is better)**

learn an unbiased ranking function in the context of selection bias. Our experimental results show that it performs better than $Heckman^{rank}$ selection bias correction algorithm when the selection bias severity is very high and position bias severity is low. The DML-based approach performs similarly with $Heckman^{rank}$ in case of low selection bias or high position bias. Our code is available at https://github.com/edgeslab/dml_ltr.git. Some promising future research directions can be adapted for unbiased learning-to-rank using Double Machine Learning. Position bias correction and joint debiasing of position and selection bias correction using DML can be taken as a promising future research direction. Adapting these bias correction methods to more advanced pairwise and listwise ranking algorithms such as *LambdaMART*, *ListNet* can also be taken as a future research direction.

## 7 IDEAS FOR NEXT STEPS

We notice that our DML-based selection method does not outperform significantly than the $Heckman^{rank}$, especially when selection bias is low. The next step will be to further improve the performance in this scenario. I have tried different ways to improve the performance such as using different machine learning models in DML, using different numbers of folds in cross-validation in DML, and using higher order terms of the features and the interaction terms in the ranking function. I also tried to use *LambdaMart* pairwise ranking algorithm as the final ranking function but the performance does not improve significantly for biased *LambdaMart* and DML based unbiased *LambdaMart*. At this point, we can take some alternative steps such as:

- We have used the individual treatment effect as an additional covariate in the ranking function. The thing is that each document has 700 dimensional feature vector. Just adding the selection bias effect as one additional feature

may not correct the selection bias to a great extent. We may try to increase the weight of the selection bias effect to give more importance to this feature.

- Alternatively, we can try to adjust the relevance label of each document based on the estimated causal effect such as subtracting the causal effect from the relevance labels. Using the adjusted relevance labels, we can train our ranking function.
- We may change the synthetic click data generation process. Our synthetic click data generation process generates biased click data based on the position of the items but we are correcting for selection bias. This may be one of the reasons why our method is not improving much than the state-of-the-art methods.
- For synthetic click data generation for selection bias, we can generate click labels based on whether users have observed a document or not. For example, if the cutoff level is 10, we can set click labels as 1 randomly from that 10 documents, and 0 for the rest of the documents.

Some other steps can also be performed after improving the performance of the current approach.

- We can our run experiments with some noisy click data to check how robust our method is in the presence of noisy click data.
- We can evaluate the model performance on a larger set of data such as the Set 1 dataset of Yahoo! Learning to rank challenge or the MSLR-WEB30K dataset from Microsoft.
- We can change our baseline methods with more recent position and selection bias correction algorithms in LTR systems though I have not found any recent selection bias correction algorithm in LTR systems other than the $Heckman^r ank$.

## REFERENCES


[1] Aman Agarwal, Kenta Takatsu, Ivan Zaitsev, and Thorsten Joachims. 2019. A General Framework for Counterfactual Learning-to-Rank. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 5–14. https://doi.org/10.1145/3331184.3331202

[2] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W. Bruce Croft. 2018. Unbiased Learning to Rank with Unbiased Propensity Estimation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (Ann Arbor, MI, USA) *(SIGIR '18)*. Association for Computing Machinery, New York, NY, USA, 385–394. https://doi.org/10.1145/3209978.3209986

[3] Deepankar Basu. 2023. The Yule-Frisch-Waugh-Lovell Theorem. arXiv:2307.00369 [econ.EM]

[4] Mouxiang Chen, Chenghao Liu, Zemin Liu, and Jianling Sun. 2022. Scalar is Not Enough: Vectorization-based Unbiased Learning to Rank. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 136–145.

[5] Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. 2018. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal* 21, 1 (01 2018), C1–C68. https://doi.org/10.1111/ectj.12097 arXiv:https://academic.oup.com/ectj/article-pdf/21/1/C1/27684918/ectj00c1.pdf

[6] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An Experimental Comparison of Click Position-Bias Models. In *Proceedings of the 2008 International Conference on Web Search and Data Mining* (Palo Alto, California, USA) *(WSDM '08)*. Association for Computing Machinery, New York, NY, USA, 87–94. https://doi.org/10.1145/1341531.1341545

[7] José Miguel Hernández-Lobato, Neil Houlsby, and Zoubin Ghahramani. 2014. Probabilistic Matrix Factorization with Non-Random Missing Data. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32* (Beijing, China) *(ICML'14)*. JMLR.org, II–1512–II–1520.

[8] Ziniu Hu, Yang Wang, Qu Peng, and Hang Li. 2019. Unbiased LambdaMART: An Unbiased Pairwise Learning-to-Rank Algorithm. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 2830–2836. https://doi.org/10.1145/3308558.3313447

[9] Rolf Jagerman, Harrie Oosterhuis, and Maarten de Rijke. 2019. To Model or to Intervene: A Comparison of Counterfactual and Online Learning to Rank from User Interactions. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Paris, France) *(SIGIR'19)*. Association for Computing Machinery, New York, NY, USA, 15–24. https://doi.org/10.1145/3331184.3331269

[10] Thorsten Joachims. 2002. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Edmonton, Alberta, Canada) *(KDD '02)*. Association for Computing Machinery, New York, NY, USA, 133–142. https://doi.org/10.1145/775047.775067

[11] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2017. Accurately Interpreting Clickthrough Data as Implicit Feedback. *SIGIR Forum* 51, 1 (aug 2017), 4–11. https://doi.org/10.1145/3130332.3130334

[12] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased Learning-to-Rank with Biased Feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining* (Cambridge, United Kingdom) *(WSDM '17)*. Association for Computing Machinery, New York, NY, USA, 781–789. https://doi.org/10.1145/3018661.3018699

[13] Dan Luo, Lixin Zou, Qingyao Ai, Zhiyu Chen, Dawei Yin, and Brian D. Davison. 2023. Model-Based Unbiased Learning to Rank. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining* (Singapore, Singapore) *(WSDM '23)*. Association for Computing Machinery, New York, NY, USA, 895–903. https://doi.org/10.1145/3539597.3570395

[14] Harrie Oosterhuis. 2023. Doubly Robust Estimation for Correcting Position Bias in Click Feedback for Unbiased Learning to Rank. *ACM Trans. Inf. Syst.* 41, 3, Article 61 (feb 2023), 33 pages. https://doi.org/10.1145/3569453

[15] Harrie Oosterhuis and Maarten de Rijke. 2020. Taking the Counterfactual Online: Efficient and Unbiased Online Evaluation for Ranking. In *Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval* (Virtual Event, Norway) *(ICTIR '20)*. Association for Computing Machinery, New York, NY, USA, 137–144. https://doi.org/10.1145/3409256.3409820

[16] 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* (Taipei, Taiwan) *(WWW '20)*. Association for Computing Machinery, New York, NY, USA, 1863–1873. https://doi.org/10.1145/3366423.3380255

[17] Zohreh Ovaisi, Kathryn Vasilaky, and Elena Zheleva. 2021. Propensity-Independent Bias Recovery in Offline Learning-to-Rank Systems. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 1763–1767. https://doi.org/10.1145/3404835.3463097

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

[19] Yi Ren, Hongyan Tang, and Siwen Zhu. 2022. Unbiased Learning to Rank with Biased Continuous Feedback. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 1716–1725.

[20] Yuta Saito. 2020. Unbiased Pairwise Learning from Biased Implicit Feedback. In *Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval* (Virtual Event, Norway) *(ICTIR '20)*. Association for Computing Machinery, New York, NY, USA, 5–12. https://doi.org/10.1145/3409256.3409812

[21] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as Treatments: Debiasing Learning and Evaluation. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48* (New York, NY, USA) *(ICML'16)*. JMLR.org, 1670–1679.

[22] Ali Vardasbi, Maarten de Rijke, and Ilya Markov. 2021. Mixture-Based Correction for Position and Trust Bias in Counterfactual Learning to Rank. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (Virtual Event, Queensland, Australia) *(CIKM '21)*. Association for Computing Machinery, New York, NY, USA, 1869–1878. https://doi.org/10.1145/3459637.3482275

[23] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to Rank with Selection Bias in Personal Search. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Pisa, Italy) *(SIGIR '16)*. Association for Computing Machinery, New York, NY, USA, 115–124. https://doi.org/10.1145/2911451.2911537

[24] 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 Eleventh ACM International Conference on Web Search and Data Mining* (Marina Del Rey, CA, USA) *(WSDM '18)*. Association for Computing Machinery, New York, NY, USA, 610–618. https://doi.org/10.1145/3159652.3159732