UNBIASED LEARNING-TO-RANK FROM LOGGED IMPLICIT FEEDBACK

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Aman Agarwal

May 2020

UNBIASED LEARNING-TO-RANK FROM LOGGED IMPLICIT FEEDBACK

Aman Agarwal, Ph.D.

Cornell University 2020

Learning-to-rank (LTR) search results in large scale industrial information retrieval settings, such as personal email and e-commerce, directly from logged implicit user feedback such as clicks is highly attractive since such feedback is ubiquitous, routinely collected, user-focused and time-sensitive unlike manual relevance annotations or slow, disruptive A/B testing protocols. However, LTR from such feedback is challenging since it can be very partial and biased as signals of relevance. In particular, position bias must be addressed since higher ranks are more likely to be examined and clicked, and thus naively interpreting clicks as relevance labels leads to undesirable feedback loops and sub-optimal ranking quality. Towards this end, we develop a theoretical framework based on counterfactual reasoning that systematically deals with the various forms of position bias inherent in user behavior, and demonstrate its effectiveness in several real-world settings including Gmail and Arxiv search. While the framework can be adapted for any form of implicit feedback, we primarily focus on click data since they are routinely logged and reliable indicators of user intent.

We present our key contributions within this framework, especially Intervention Harvesting, the first method for consistent position-bias estimation without additional online interventions or relevance modeling using logs from multiple rankers. The general unbiased LTR framework, and addressing position-dependent trust bias in relevance evaluation (in addition to examination bias) are also described in detail.

BIOGRAPHICAL SKETCH

Aman Agarwal was born and raised in India. He first came to the United States in 2012 for his undergraduate studies at Caltech, where he majored in Computer Science, while also completing significant coursework in Mathematics. At Caltech, Aman developed an appreciation for novel research through various projects with faculty members in areas ranging from complexity theory to graphics and machine learning. These formative experiences led him subsequently to the doctoral program at Cornell in 2016, where he was advised by Prof. Thorsten Joachims.

Aman has also had the opportunity to delve into the real-world applications of his work through internships at Two Sigma, Google AI and Bloomberg over the years. Upon graduation, he is looking forward to join Facebook in 2020 at Menlo Park as a Research Scientist in machine learning.

ACKNOWLEDGEMENTS

This thesis is a product of my doctoral studies undertaken under the direct supervision of my advisor Professor Thorsten Joachims. Our projects also greatly benefited from the inputs of our various collaborators, especially Xuanhui Wang and Marc Najork, who hosted me as an intern in their team at Google. Furthermore, I am grateful to Professors Karthik Sridharan and Andrew Wilson for their guidance as members of my academic committee.

Beyond the main focus of my thesis, I also had the great opportunity to collaborate with Professor Shimon Edelman on a journal article titled *Functionally Effective Conscious AI Without Suffering*. I truly enjoyed my discussions with Prof. Edelman regarding AI, philosophy of mind and contemplative practice.

The research presented in this thesis was supported in part by NSF Awards IIS-1615706, IIS-1901168 and IIS-1513692, an Amazon Research Award, and the Criteo Faculty Research Award program. All contents represent the opinions of the author, which are not necessarily shared or endorsed by their respective employers and/or sponsors.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

In the Learning-to-Rank (LTR) setting, the target is a ranking function that ranks candidate documents (or results) for a given query such that some desired utility measure (e.g. user engagement, ad revenue etc.) is maximized over the run of the system. Learning such ranking functions via supervised batch training requires collecting relevance-label annotations for domain-specific query-document pairs. This can be highly infeasible, if not impossible, for large scale applications, especially in user-specific domains such as personal email search. In such a scenario, directly learning from the feedback (e.g. clicks, hover times etc.) implicit in historical logs of user interactions with the ranking system is highly attractive since such feedback is abundant, user-specific and easily logged for typical industrial information retrieval (IR) applications (e.g. Gmail, e-commerce etc.).

Effectively harnessing the already available logged data is also more preferable than performing online interventions such as A/B testing which can be slow, expensive and disruptive. However, LTR from logged implicit feedback is challenging since such feedback can be very partial and biased as signals of relevance. In particular, position bias must be addressed since higher ranks are more likely to be examined and clicked, and thus naively interpreting clicks as relevance labels leads to undesirable feedback loops and sub-optimal ranking quality. Towards this end, we develop a theoretical framework based on counterfactual reasoning that systematically deals with the various forms of position bias inherent in user behavior, and demonstrate its effectiveness in several real-world settings. While the framework can be adapted for any form of implicit feedback, we primarily focus on click data since they are routinely logged and reliable indicators of user intent.

Our work takes the unbiased LTR approach introduced in [33] as a starting point. This approach has three broad components, and we make fundamental contributions to each. First, a model for user behavior must be hypothesized to explain the observed logged implicit feedback (such as clicks) in terms of the underlying unobserved document relevance, position, and other potential contextual information available about the query and user. Next, the parameters of the chosen model must be estimated using the logged data. Specifically, this step quantifies the position bias of various forms, and thus determines the relationship between the observed feedback and the true unobserved relevance of the ranked documents. This step may require a relatively small set of further online interventions, but can also benefit from novel techniques that are largely, if not entirely, offline, including key contributions of our work. Finally, these bias estimates can be employed in conjunction with the observed feedback to learn a new ranking function by indirectly optimizing a desired relevance-based ranking metric (such as Discounted Cumulative Gain, or DCG). This entails designing a training objective that accounts for the various biases via causal inference techniques such as inverse propensity scoring (IPS), so that the objective matches the unobserved relevance-based metric in expectation over the stochastic user behavior.

In the following sections, we provide an overview of the subsequent chapters by outlining our key contributions in the context of the above approach.

1.1 Generalized Counterfactual LTR Framework

The counterfactual inference approach developed in [33] provides a rigorous strategy for unbiased LTR despite biased feedback and overcomes the drawbacks of alternative bias-mitigation strategies. In particular, it does not require the same query to be seen multiple times as necessary for most generative click models, and it does not introduce alternate biases like treating clicks as preferences between clicked and skipped documents. However, [33] is limited to linear ranking functions and optimizing the Average Rank metric. Consequently, we address these limitations in [2] by designing a counterfactual LTR framework that generalizes to a broad class of additive IR metrics as well as non-linear deep models.

The key technique underlying the counterfactual framework is inverse propensity scoring (IPS) which accounts for the differential propensity of users to examine documents across different positions. In particular, documents clicked at lower ranks are given proportionately higher weights during training by incorporating the estimated propensity of observing the clicks into a provably unbiased Propensity-Weighted Empirical Risk Minimization (ERM) training objective [50]. Our generalized framework allows any IR metric that is the sum of individual document relevances weighted by some function of document rank to be directly optimized via such a Propensity-Weighted ERM. Moreover, if an IR metric meets the mild requirement that the rank weighting function is monotone, then a hinge-loss upper-bounding technique can be used to learn a broad class of differentiable models (e.g. deep networks) as scoring functions for ranking documents.

To demonstrate the effectiveness of the general framework, we fully develop two learning-to-rank methods that optimize the Discounted Cumulative Gain (DCG) metric commonly employed for various industrial IR applications. The first is SVM PropDCG, which generalizes a Ranking SVM to directly optimize a bound on DCG from biased click data. The resulting optimization problem is no longer convex, and we show how to find a local optimum using the Convex Concave Procedure (CCP). In CCP, several iterations of convex sub-problems are solved. In the case of SVM PropDCG, these convex sub-problems have the convenient property of being a Quadratic Program analogous to a generalized Ranking SVM. This allows the CCP to work by invoking an existing and fast SVM solver in each iteration until convergence. The second method we develop is Deep PropDCG, which further generalizes the approach to deep networks as non-linear ranking functions. Deep PropDCG also optimizes a bound on the DCG, and we show how the resulting optimization problem can be solved via stochastic gradient descent for any network architecture that shares neural network weights across candidate documents for the same query.

In addition to the theoretical derivation and the justification, we also empirically evaluate the effectiveness of both SVM PropDCG and Deep PropDCG, especially in comparison to the basic SVM PropRank method introduced in [33]. We find that SVM PropDCG performs significantly better than SVM PropRank in terms of DCG, and that it is robust to varying degrees of bias, noise and propensity-model misspecification. In our experiments, CCP convergence was typically achieved quickly within three to five iterations. For Deep PropDCG, the results show that DCG performance is further improved compared to SVM PropDCG when using a neural network, thus demonstrating that the counterfactual learning approach can effectively train non-linear ranking functions.

SVM PropDCG and Deep PropDCG are also seen to outperform LambdaRank, the industry LTR standard, in terms of DCG.

1.2 Position Bias Estimation via Intervention Harvesting

As discussed, the key prerequisite for counterfactual LTR is estimation of the propensity of obtaining a particular feedback signal, which enables Propensity-Weighted ERM. Towards this end, we propose the first approach for producing consistent propensity estimates without manual relevance judgments, disruptive interventions, or restrictive relevance-modeling assumptions in [5].

We primarily focus on propensity estimation under the Position-Based Propensity Model (PBM), which hypothesizes that the probability of clicking a relevant document is equal to the purely rank-dependent probability of examining it. The PBM has been seen to be simple yet effective in [33], but previous propensity estimation methods for it have substantial drawbacks. In particular, the conventional estimator for the PBM takes a generative approach [16] and requires individual queries to repeat many times. This is unrealistic for most ranking applications. To avoid this requirement of repeating queries, Wang et al. [55] included a relevance model that is jointly estimated via an Expectation-Maximization (EM) procedure. Unfortunately, defining an accurate relevance model is just as difficult as the learning-to-rank problem itself, and a misspecified relevance model can lead to biased propensity estimates. An alternative to generative click modeling on observational data are estimators that rely on specific randomized interventions – for example, randomly swapping the result at rank 1 to any rank k [33]. While this provides provably consistent propensity estimates for the PBM, it degrades retrieval performance and user experience [55] and is thus costly. In addition, we find in our work that swap-interventions are statistically rather inefficient. Our proposed approach overcomes the disadvantages of the existing methods, as it does not require repeated queries, a relevance model, or additional interventions.

The key idea behind our estimation technique is to exploit data from a natural intervention that is readily available in virtually any operational system – namely that we have implicit feedback data from more than one ranking function. We call this Intervention Harvesting. Since click behavior depends jointly on examination and relevance, we show how to exploit this intervention to control for any difference in overall relevance of results at different positions under the PBM. This makes our approach fundamentally interventional and it is thus consistent analogous to using explicit swap interventions under mild assumptions. However, by leveraging existing data that is readily available in most systems, it does not require additional online interventions and the resulting decrease in user experience. To make efficient use of the interventional data we harvest, we propose a specific extremum estimator – called AllPairs – that combines all available data for improved statistical efficiency without the need for a relevance model that could introduce bias. We find that this estimator works well even if the rankers we harvest interventions from are quite similar in their result placements and overall performance, and that it is able to recover the relative propensities globally even if most of the changes in rank are small. Furthermore, we find empirically that the new estimator provides superior propensity estimates in two real-world systems – Arxiv Full-text Search and Google Drive Search.

The Position-Based Model (PBM) however only models how examination changes with the rank of the result. In follow-up work in [20], we show how Intervention Harvesting can also be adapted to a more expressive Contextual Position-Based Model (CPBM) in which the examination propensity can vary based on query, user and other context observable information in addition to position. This is a significant improvement since, for example, how users examine results in navigational queries (i.e., search queries entered with the intention of finding a particular website or webpage) is likely to differ from that of informational queries (i.e., search queries for a broad topic for which there could be thousands of relevant results), a phenomenon also observed in Arxiv Search experiments.

1.3 Addressing Trust Bias

Our discussion so far has been based on the PBM, and its extension to the Contextual PBM, both of which model the differential propensity of examining documents at different results, but ignore the possibility of position-dependent noise in relevance evaluation upon examination, which turns out to be another important source of position bias. The robustness of the purely IPS-based approach when position-independent constant noise is present in click data is shown in [33]. However, as the user study in [30] showed qualitatively, there is a position-dependent trust bias after examining results, capturing the intuition that users may overestimate (or underestimate) the relevance of a result given its high (or low) position, due to their trust in the effectiveness of the search application to rank relevant documents higher. Thus a non-relevant result may be clicked and a relevant result may not be clicked, and the probabilities are not constant but depend on the position of the result.

To address this issue, we introduce a noise-aware PBM, named TrustPBM, to model trust bias in user clicks in [3]. The standard PBM model assumes that an examined and relevant document is always clicked. We extend it to capture the following uncertainty: after examination, a relevant document can be missed, and meanwhile a non-relevant document can be clicked. TrustPBM is a click model that gives a rigorous mathematical formulation of the trust bias. However, its bias parameters cannot be estimated based on intervention like the result randomization used for PBM. Even if we can estimate bias parameters in TrustPBM, it is still unclear how to use them for unbiased LTR. We show that TrustPBM can be effectively estimated using an EM algorithm, inspired by the previous work [55]. Since TrustPBM is no longer a factorization model, a purely IPS method for unbiased LTR is not feasible. We thus derive a novel and mathematically principled Bayes-IPS correction that uses a Bayesian approach for trust bias and IPS for examination bias. Conceptually, the Bayes-IPS method addresses both click incompleteness bias (using IPS) and click noise (using Bayes rule) for unbiased LTR.

Next, we empirically evaluate our method on click logs from search services for Gmail, Google Drive, and Gmail Expert users. Our TrustPBM estimates obtained via the EM algorithm reveal novel insights into the nature of relevance noise present in real settings. Most interestingly, we observe that most of the noise comes from higher ranked irrelevant documents being judged relevant, while relevant documents tend to be judged accurately even at lower positions. Moreover, our results demonstrate that the TrustPBM fits the click data better than the plain PBM. Furthermore, ranking models trained with the Bayes-IPS correction have superior live experiment metrics than those trained with only the IPS correction based on PBM, estimated either via EM or result randomization.

1.4 Bibliographic Remarks

The work presented in this thesis is based on several conference papers which were published during my doctoral studies under the supervision of my advisor Professor Thorsten Joachims. While I was the lead author of these papers, the projects greatly benefited from the inputs of all our collaborators.

Specifically, Chapter 3 focuses on the work in [2] which appeared at SIGIR 2019. Ivan and Kenta did substantial work for the empirical analyses, especially the engineering behind the Arxiv real-world experiments. Chapter 4 focuses on [5] which appeared at WSDM 2019. This was a collaboration with my internship team at Google, who enabled the experiments on Google search data using their proprietary infrastructure. Ivan led the real-world experimentation on Arxiv. Finally, Chapter 5 focuses on [3] which appeared at WWW 2019. This work was primarily done during my internship at Google, especially in collaboration with Xuanhui who initially observed the possibility of position-dependent relevance noise in the data. The rest of the team members, Cheng, Mike and Marc, helped with the real-world experiments on Google's systems.

CHAPTER 2
LITERATURE REVIEW

We focus on learning-to-rank (LTR) using implicit user feedback such as click logs in search applications. In this setting, clicks are an incomplete and biased signal of result relevance since higher results are more likely to be clicked. Leveraging user interaction data such as clicks has been shown to be quite promising in improving search quality. A simple approach is to use click and non-click information as relevance judgment of results. This can be used for evaluation or to train a new ranking function in the LTR setting. However, such an approach can lead to misleading evaluation results or sub-optimal ranking functions due to various types of bias in click data, e.g., position bias [30], presentation bias [59], and trust bias [41, 30]. In the past, a large amount of research has been devoted to extracting more accurate signals from click data. For example, some heuristic methods have been proposed to address the position bias by utilizing the pairwise preferences between clicked and skipped documents (e.g., SkipAbove) [30, 28, 31]. Though these methods have been found to provide more accurate relevance assessments, their data is fundamentally biased. For example, a ranking function trained based on the SkipAbove heuristic tends to reverse the presented order due to its sampling bias [28].

The key ingredient in the counterfactual framework is bias-correcion via inverse propensity scoring (IPS) which was developed in causal inference [47] and is a widely employed technique for handling sampling bias. IPS is often used for unbiased evaluation and learning [1, 18, 36, 37, 48, 50]. The common assumption in most of these studies is that the propensities are under the system's control and are thus known. In the unbiased LTR setting, however, propensities arise from user behavior and thus need to be estimated. While IPS can be used to address the sample bias or click incompleteness, it is not straightforward for click noise and we address this challenge in [3] using a Bayes correction term.

2.1 Click Modeling and Bias Estimation

Click modeling, especially the position-based model (PBM), has been used for estimating propensities. Under this model, [54] proposed a result randomization method that randomly shuffles the top n results and uses the average CTR at each position as the propensity. [33] proposed a pivot-based randomization where position 1 and k are randomly shuffled. The relative propensity between these two positions p_k/p_1 can be derived and this is sufficient for LTR. [55] proposed a RandPair approach where adjacent positions are randomly flipped to obtain p_{k+1}/p_k and a chain rule is used to estimate the ratio p_k/p_1. While these approaches require interventions, more recently, methods for propensity estimation have been developed that do not require online interventions. [55] proposed a regression-based EM algorithm to fit the PBM model to the regular click logs without interventions. Our method in [5] achieves consistent propensity estimation without interventions or relevance modeling by harvesting naturally randomized pairs from logs of multiple rankers in operational systems. While the above approaches focus on the simple PBM, we also extend the intervention harvesting approach in [5] to Contextual Position-Based Model (CPBM) in [20], where the context information is also incorporated into the propensity model, to provide more accurate estimates. Unlike traditional click modeling, we do not use the PBM as a generative model to infer relevance, but instead use interventional techniques to infer propensities even without repeat queries.

With the propensities estimated, the IPS-based approach is also used for search quality evaluations, in addition to learning a ranking function. For example, [11] extends the counterfactual inference framework by considering the case of evaluating new rankers that can retrieve previously unseen documents. [12] also studies the click-through bias in the cascade model for evaluation.

Besides PBM, there are many other click models for extracting relevance signals from click data. Click models are parameterized generative models [16] and their parameters are typically estimated via generative maximum likelihood under specific modeling assumptions about user behavior. There are two classic click models: the PBM model [45] and the Cascade model [17]. Based on these two models, more advanced models have been developed. For example, the user browsing model (UBM) [19] extends PBM to condition the examination on a previously clicked position, in addition to the position of the current result. The dynamic Bayesian network model (DBN) [15] and the click chain model (CCM) [23] extend the Cascade model to handle multiple clicks. Both models assume a sequential user behavior over the result list. They focus on modeling the probability that document gets examined depending on the previous documents and differ in their modeling formulation. DBN and CCM are different from PBM and UBM in that the examination of a later document depends on the relevance of previous documents. However, most of these models assume clicks are noise-free. Our work in [3] addresses the issue of position-dependent click noise, conceptualized as trust bias, extending the PBM. While the estimated examination probability in any click model can be used in unbiased LTR, the PBM model is more robust given that the examination does not depend on the relevance of previous documents since estimating relevance is as hard as learning a ranking function and thus leads to high variance of the estimated examination probability. Our TrustPBM model enjoys the same advantage where the trust bias is dependent only on position.

The alternate approach of directly using position as a feature is generally not effective for learning a ranking function. This is because the position of a result is highly predictive of its click rate, however, the position feature cannot be used during inference time when a collection of documents must be scored to determine their positions in the presented ranking. Using the position feature in training and then unplugging it during inference leads to worse results [55].

2.2 Training Methods

Beyond click modeling and propensity estimation, a ranking function must be learned from the click logs using an appropriate unbiased LTR algorithm. Recently, some algorithms ([6, 25]) have been proposed that jointly estimate the propensities and learn the ranking function. However, this requires an accurate relevance model to succeed, which is at least as hard as the LTR from biased feedback problem in question. Moreover, the two-step approach of propensity estimation followed by training an unbiased ranker allows direct optimization of any chosen target ranking metric independent of the propensity estimation step. Towards this end, in [2], we develop a general framework for counterfactual LTR that enables unbiased training for a broad class of additive rank metrics (e.g., Discounted Cumulative Gain (DCG), Precision@k) as well as a broad class of models (e.g., deep networks)

While our focus is on directly optimizing ranking performance in the implicit feedback partial-information setting, several approaches have been proposed for the same task in the full-information supervised setting, i.e. when the relevances of all the documents in the training set are known. A common strategy is to use some smoothed version of the ranking metric for optimization, as seen in SoftRank [52] and others [14, 57, 58, 29]. In particular, SoftRank optimizes the expected performance metric over the distribution of rankings induced by smoothed scores, which come from a normal distribution centered at the query-document mean scores predicted by a neural net. This procedure is computationally expensive with an $O(n^3)$ dependence on the number of documents for a query. In contrast, our approach in [2] employs an upper bound on the performance metric, whose structure makes it amenable to the Convex Concave Procedure for efficient optimization, as well as adaptable to non-linear ranking functions via deep networks.

Finally, several works exist [52, 10, 8, 46] that have proposed neural network architectures for learning-to-rank. We do not focus on a specific network architecture in our work ([2]), but instead propose a new training criterion for learning-to-rank from implicit feedback that in principle allows unbiased network training for a large class of architectures.

CHAPTER 3

A GENERAL FRAMEWORK FOR COUNTERFACTUAL LEARNING-TO-RANK

We begin by developing a counterfactual learning framework that covers the full class of linearly decomposable metrics as defined below (e.g. DCG). This extends [33] which was limited to the Average Rank metric. Suppose we are given a sample $\boldsymbol{X}$ of i.i.d. query instances $\boldsymbol{x}_i \sim \mathrm{P}(\boldsymbol{x})$, $i \in [N]$. A query instance can include personalized and contextual information about the user in addition to the query string. For each query instance $\boldsymbol{x}_i$, let $\mathrm{r}_i(y)$ denote the user-specific relevance of result y for instance $\boldsymbol{x}_i$. For simplicity, assume that relevances are binary, $\mathrm{r}_i(y) \in \{0, 1\}$. In the following, we consider the class of additive ranking performance metrics, which includes any metric that can be expressed as

$$\Delta(\boldsymbol{y}|\boldsymbol{x}_i, \mathrm{r}_i) \quad = \quad \sum_{y \in \boldsymbol{y}} \lambda\left(\mathrm{rank}(y|\boldsymbol{y})\right) \cdot \mathrm{r}_i(y). \tag{3.1}$$

$\boldsymbol{y}$ denotes a ranking of results, and $\lambda\left(\right)$ can be any weighting function that depends on the rank $\mathrm{rank}(y|\boldsymbol{y})$ of document y in ranking $\boldsymbol{y}$. A broad range of commonly used ranking metrics falls into this class, and Table 3.1 lists some of them. For instance, setting $\lambda\left(\mathrm{rank}\right) = \mathrm{rank}$ gives the sum of relevant ranks metric (also called average rank when normalized) considered in [33], and $\lambda\left(\mathrm{rank}\right) = \frac{-1}{\log(1+\mathrm{rank})}$ gives the DCG metric. Note that we consider negative values wherever necessary to make the notation consistent with risk minimization.

Metric	$\lambda\left(\mathrm{rank}\right)$
AvgRank	rank
DCG	$-1/\log(1 + \mathrm{rank})$
Prec@k	$-\mathbf{1}_{\mathrm{rank} \leq k}/k$
RBP-p [40]	$-(1-p)/p^{\mathrm{rank}}$

Table 3.1: Some popular linearly decomposable IR metrics that can be directly optimized by Propensity-Weighted ERM. $\lambda\left(r\right)$ is the rank weighting function.

A ranking system S maps a query instance $\boldsymbol{x}_i$ to a ranking $\boldsymbol{y}$. Aggregating the losses of individual rankings over the query distribution, we can define the overall *risk* (e.g., the expected DCG) of a system as

$$R(S) \quad = \quad \int \Delta(S(\boldsymbol{x})|\boldsymbol{x}, \mathrm{r})\, d\mathrm{P}(\boldsymbol{x}, \mathrm{r}). \tag{3.2}$$

A key problem when working with implicit feedback data is that we cannot assume that all relevances r_i are observed. In particular, while a click (or a sufficiently long dwell time) provides a noisy indicator of positive relevance in the presented ranking $\bar{\boldsymbol{y}}_i$, a missing click does not necessarily indicate lack of relevance as the user may not have observed that result. From a machine learning perspective, this implies that we are in a partial-information setting, which we will deal with by explicitly modeling missingness in addition to relevance. Let $o_i \sim Q(o|\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i, \mathrm{r}_i)$ denote the 0/1 vector indicating which relevance values are revealed. While o_i is not necessarily fully observed either, we can now model its distribution, which we will find below is sufficient for unbiased learning despite the missing data. In particular, the *propensity* of observing $\mathrm{r}_i(y)$ for query instance x_i given presented ranking $\bar{\boldsymbol{y}}$ is then defined as $Q(o_i(y) = 1|\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i, \mathrm{r}_i)$.

Using this counterfactual setup, an unbiased estimate of $\Delta(\boldsymbol{y}|\boldsymbol{x}_i, \mathrm{r}_i)$ for any ranking $\boldsymbol{y}$ can be obtained via IPS weighting

$$\hat{\Delta}_{IPS}(\boldsymbol{y}|\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i, o_i) = \sum_{\substack{y: o_i(y)=1 \\ \wedge\, \mathrm{r}_i(y)=1}} \frac{\lambda(\mathrm{rank}(y|\boldsymbol{y}))}{Q(o_i(y)=1|\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i, \mathrm{r}_i)}. \tag{3.3}$$

This is an unbiased estimator since,

$$\begin{aligned}
&\mathbb{E}_{o_i}[\hat{\Delta}_{IPS}(\boldsymbol{y}|\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i, o_i)] \qquad (3.4)\\
&= \mathbb{E}_{o_i}\left[\sum_{y:o_i(y)=1} \frac{\lambda\,(\mathrm{rank}(y|\boldsymbol{y}))\cdot \mathrm{r}_i(y)}{Q(o_i(y)=1|\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i, \mathrm{r}_i)}\right]\\
&= \sum_{y\in\boldsymbol{y}} \mathbb{E}_{o_i}\left[\frac{o_i(y)\cdot\lambda\,(\mathrm{rank}(y|\boldsymbol{y}))\cdot \mathrm{r}_i(y)}{Q(o_i(y)=1|\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i, \mathrm{r}_i)}\right]\\
&= \sum_{y\in\boldsymbol{y}} \frac{Q(o_i(y)=1|\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i, \mathrm{r}_i)\cdot\lambda\,(\mathrm{rank}(y|\boldsymbol{y}))\cdot \mathrm{r}_i(y)}{Q(o_i(y)=1|\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i, \mathrm{r}_i)}\\
&= \sum_{y\in\boldsymbol{y}} \lambda\,(\mathrm{rank}(y|\boldsymbol{y}))\,\mathrm{r}_i(y)\\
&= \Delta(\boldsymbol{y}|\boldsymbol{x}_i, \mathrm{r}_i),
\end{aligned}$$

assuming $Q(o_i(y) = 1|\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i, \mathrm{r}_i) > 0$ for all y that are relevant $\mathrm{r}_i(y) = 1$. The above proof is a generalized version of the one in [33] for the Average Rank metric. Note that the estimator in Equation (3.3) sums only over the results where the feedback is observed (i.e., $o_i(y) = 1$) and positive (i.e., $\mathrm{r}_i(y) = 1$), which means that we do not have to disambiguate whether lack of positive feedback (e.g., the lack of a click) is due to a lack of relevance or due to missing the observation (e.g., result not relevant vs. not viewed).

Using this unbiased estimate of the loss function, we get an unbiased estimate of the risk of a ranking system S

$$\hat{R}_{IPS}(S) = \frac{1}{N}\sum_{i=1}^{N} \sum_{\substack{y:o_i(y)=1\\ \wedge\, \mathrm{r}_i(y)=1}} \frac{\lambda\,(\mathrm{rank}(y|S(\boldsymbol{x}_i)))}{Q(o_i(y)=1|\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i, \mathrm{r}_i)}. \qquad (3.5)$$

Note that the propensities $Q(o_i(y) = 1|\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i, \mathrm{r}_i)$ are generally unknown, and must be estimated based on some model of user behavior. Practical approaches to estimating the propensities are given in [33, 55, 2, 20], and also form the central focus of Chapter 4.

3.1 Unbiased Empirical Risk Minimization for LTR

The propensity-weighted empirical risk from Equation (3.5) can be used to perform Empirical Risk Minimization (ERM)

$$\hat{S} \quad = \quad \operatorname{argmin}_{S \in \mathcal{S}} \left\{ \hat{R}_{IPS}(S) \right\}.$$

Under the standard uniform convergence conditions [53], the unbiasedness of the risk estimate implies consistency in the sense that given enough training data, the learning algorithm is guaranteed to find the best system in $\mathcal{S}$. We have thus obtained a theoretically justified training objective for learning-to-rank with additive metrics like DCG. However, it remains to be shown that this training objective can be implemented in efficient and practical learning methods. This section shows that this is indeed possible for a generalization of Ranking SVMs and for deep networks as ranking functions.

Consider a dataset of n examples of the following form. For each query-result pair $(\boldsymbol{x}_i, y_i)$ that is clicked, let $q_i = Q(o_i(y) = 1 | \boldsymbol{x}_i, \bar{\boldsymbol{y}}_i, \mathrm{r}_i)$ be the propensity of the click according to a click propensity model such as the Position-Based Model [33, 55]. We also record the candidate set Y_i of all results for query $\boldsymbol{x}_i$. Note that each click generates a separate training example, even if multiple clicks occur for the same query.

Given this propensity-scored click data, we would like to learn a scoring function $f(\boldsymbol{x}, y)$. Such a scoring function f naturally specifies a ranking system S by sorting candidate results Y for a given query $\boldsymbol{x}$ by their scores.

$$S_f(\boldsymbol{x}) \equiv \operatorname{argsort}_Y \{f(\boldsymbol{x}, y)\} \tag{3.6}$$

Since $\operatorname{rank}(y | S_f(\boldsymbol{x}))$ of a result is a discontinuous step function of the score, tractable learning algorithms typically optimize a substitute loss that is (sub-)differentiable [28, 58, 52]. Following this route, we now derive a tractable substitute for the empirical risk of (3.5) in terms of the scoring function. This is achieved by the following hinge-loss upper bound [33] on the rank

$$\begin{aligned}\mathrm{rank}(y_i|\boldsymbol{y}) - 1 &= \sum_{\substack{y \in Y_i \\ y \neq y_i}} \mathbb{1}_{f(\boldsymbol{x}_i, y) - f(\boldsymbol{x}_i, y_i) > 0} \\ &\leq \sum_{\substack{y \in Y_i \\ y \neq y_i}} \max(1 - (f(\boldsymbol{x}_i, y_i) - f(\boldsymbol{x}_i, y)), 0).\end{aligned}$$

Using this upper bound, we can also get a bound for any IR metric that can be expressed through a monotonically increasing weighting function $\lambda(r)$ of the rank. Note that this monotonicity condition is satisfied by all the metrics in Table 3.1. By rearranging terms and applying the weighting function $\lambda(r)$, we have

$$\lambda(\mathrm{rank}(y_i|\boldsymbol{y})) \leq \lambda\left(1 + \sum_{\substack{y \in Y_i \\ y \neq y_i}} \max(1 - (f(\boldsymbol{x}_i, y_i) - f(\boldsymbol{x}_i, y)), 0)\right).$$

This provides the following continuous and subdifferentiable upper bound $\hat{R}_{IPS}^{hinge}(f)$ on the propensity-weighted risk estimate of (3.5).

$$\begin{aligned}\hat{R}_{IPS}(S_f) &\leq \hat{R}_{IPS}^{hinge}(f) \\ &= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{q_i}\lambda\left(1 + \sum_{\substack{y \in Y_i \\ y \neq y_i}} \max(1 - (f(\boldsymbol{x}_i, y_i) - f(\boldsymbol{x}_i, y)), 0)\right) \qquad (3.7)\end{aligned}$$

Focusing on the DCG metric, we show in the following how this upper bound can be optimized for linear as well as non-linear neural network scoring functions. For the general class of additive IR metrics, the optimization depends on the properties of the weighting function $\lambda(r)$, and we highlight them wherever appropriate.

3.1.1 SVM PropDCG

The following derives an SVM-style method, called SVM PropDCG, for learning a linear scoring function $f(\boldsymbol{x}, y) = w \cdot \phi(\boldsymbol{x}, y)$, where w is a weight vector and $\phi(\boldsymbol{x}, y)$ is a feature vector describing the match between query $\boldsymbol{x}$ and result y. For such linear ranking functions – which are widely used in Ranking SVMs [28] and many other learning-to-rank methods [39] –, the propensity-weighted ERM bound from Equation (3.7) can be expressed as the following SVM-type optimization problem.

$$\begin{aligned}
\hat{w} \;=\; & \operatorname{argmin}_{w,\xi} \frac{1}{2} w \cdot w + \frac{C}{n} \sum_{i=1}^{n} \frac{1}{q_i} \lambda \left(\sum_{y \in Y_i} \xi_{iy} + 1 \right) \\
s.t. \quad & \forall y \in Y_1 \setminus \{y_1\} : w \cdot [\phi(\boldsymbol{x}_1, y_1) - \phi(\boldsymbol{x}_1, y)] \geq 1 - \xi_{1y} \\
& \vdots \\
& \forall y \in Y_n \setminus \{y_n\} : w \cdot [\phi(\boldsymbol{x}_n, y_n) - \phi(\boldsymbol{x}_n, y)] \geq 1 - \xi_{ny} \\
& \forall i \forall y : \xi_{iy} \geq 0
\end{aligned}$$

C is a regularization parameter. The training objective optimizes the $\mathcal{L}_2$-regularized hinge-loss upper bound on the empirical risk estimate (3.7). This upper bound holds since for any feasible (w, ξ) and any monotonically increasing weighting function $\lambda(r)$

$$\begin{aligned}
& \lambda \left(1 + \sum_{\substack{y \in Y_i \\ y \neq y_i}} \max(1 - (f(\boldsymbol{x}_i, y_i) - f(\boldsymbol{x}_i, y)), 0) \right) \\
& = \lambda \left(1 + \sum_{\substack{y \in Y_i \\ y \neq y_i}} \max(1 - w \cdot [\phi(\boldsymbol{x}_i, y_i) - \phi(\boldsymbol{x}_i, y)], 0) \right) \leq \lambda \left(1 + \sum_{y \in Y_i} \xi_{iy} \right).
\end{aligned}$$

As shown in [33], for the special case of using the sum of relevant ranks as the metric to optimize, i.e. $\lambda(r) = r$, this SVM optimization problem is a convex

Quadratic Program which can be solved efficiently using standard SVM solvers, like SVM-rank [29], via a one-slack formulation.

Moving to the case of DCG as the training metric via the weighting function $\lambda(r) = \frac{-1}{\log(1+r)}$, we get the following optimization problem for SVM PropDCG

$$\begin{aligned}\hat{w} &= \text{argmin}_{w,\xi}\ \frac{1}{2}w\cdot w - \frac{C}{n}\sum_{i=1}^{n}\frac{1}{q_i}\frac{1}{\log(\sum_{y\in Y_i}\xi_{iy}+2)}\\ s.t. &\quad \forall j\forall y\in Y_i\backslash\{y_i\}: w\cdot[\phi(\boldsymbol{x}_i,y_i)-\phi(\boldsymbol{x}_i,y)]\geq 1-\xi_{iy}\\ &\quad \forall j\forall y:\xi_{iy}\geq 0.\end{aligned}$$

This optimization problem is no longer a convex Quadratic Program. However, all constraints are still linear inequalities in the variables w and ξ, and the objective can be expressed as the difference of two convex functions h an g. Let $h(w) = \frac{1}{2}\|w\|^2$ and $g(\xi) = \frac{C}{n}\sum_{j=1}^{n}\frac{1}{q_i}\frac{1}{\log(\sum_{y\in\mathbf{Y}_i}\xi_{iy}+2)}$. Then the function h is the $\mathcal{L}_2$ norm of the vector w and is thus a convex function. As for the function g, the function $k : x \mapsto \frac{1}{\log x}$ is convex as it is the composition of a a convex decreasing function ($x \mapsto \frac{1}{x}$) with a concave function ($x \mapsto \log x$). So, since the sum of affine transformations of a convex function is convex, g is convex.

Such an optimization problem is called a convex-concave problem[1] and a local optimum can be obtained efficiently via the Convex-Concave Procedure (CCP) [38]. At a high level, the procedure works by repeatedly approximating the second convex function with its first order Taylor expansion which makes the optimization problem convex in each iteration. The Taylor expansion is first done at some chosen initial point in the feasible region, and then the solution of the convex problem in a particular iteration is used as the Taylor approximation point for the next iteration. It can be shown that this procedure converges to a local optimum [38].

[1]More generally, the inequality constraints can also be convex-concave and not just convex

Concretely, let w^k, ξ^k be the solution in the k^{th} iteration. Then, we have the Taylor approximation

$$\begin{aligned}\hat{g}(\xi;\xi^k) &= g(\xi^k) + \nabla g(\xi^k)^T(\xi - \xi^k) \\ = \quad & g(\xi^k) - \frac{C}{n}\sum_{j=1}^{n}\frac{1}{q_i}\frac{\sum_{y\in\mathbf{Y}_i}\xi_{iy} - \xi_{iy}^k}{\left(\sum_{y\in\mathbf{Y}_i}\xi_{iy}^k + 2\right)\log^2\left(\sum_{y\in\mathbf{Y}_i}\xi_{iy}^k + 2\right)}\end{aligned}$$

Letting $q_i' = q_i\left(\sum_{y\in\mathbf{Y}_i}\xi_{iy}^k + 2\right)\log^2\left(\sum_{y\in\mathbf{Y}_i}\xi_{iy}^k + 2\right)$, and dropping the additive constant terms from $\hat{g}$, we get the following convex program that needs to be solved in each CCP iteration.

$$\begin{aligned} & \operatorname{argmin}_{w,\xi}\ \frac{1}{2}w\cdot w + \frac{C}{n}\sum_{i=1}^{n}\frac{1}{q_i'}\sum_{y\in Y_i}\xi_{iy} \\ s.t. \quad & \forall i \forall y \in Y_i\backslash\{y_i\} : w\cdot[\phi(\boldsymbol{x}_i, y_i) - \phi(\boldsymbol{x}_i, y)] \geq 1 - \xi_{iy} \\ & \forall i \forall y : \xi_{iy} \geq 0\end{aligned}$$

Observe that this problem is of the same form as SVM PropRank, the Propensity Ranking SVM for the average rank metric, i.e. $\lambda(r) = r$ (with the caveat that q_i' are not propensities). This nifty feature allows us to solve the convex problem in each iteration of the CCP using the fast solver for SVM PropRank provided in [33]. In our experiments, CCP convergence was achieved within a few iterations – as detailed in the empirical section. For other IR metrics, the complexity and feasibility of the above Ranking SVM optimization procedure will depend on the form of the target IR metric. In particular, if the rank weighting function $\lambda(r)$ is convex, it may be solved directly as a convex program. If $\lambda(r)$ is concave, then the CCP may be employed as shown for the DCG metric above.

An attractive theoretical property of SVM-style methods is the ability to switch from linear to non-linear functions via the Kernel trick. In principle, kernelization can be applied to SVM PropDCG as is evident from the representer theorem [49]. Specifically, by taking the Lagrange dual, the problem can be kernelized analogous to [28]. While it can be shown that the dual is convex and strong duality holds, it is not clear that the optimization problem has a convenient and compact form that can be efficiently solved in practice. Even for the special case of the average rank metric, $\lambda(r) = r$, the associated kernel matrix $K_{iy,jy'}$ has a size equal to the total number of candidates $\sum_{i=1}^{n} |Y_i|$ squared, making the kernelization approach computationally infeasible or challenging at best. We therefore explore a different route for extending our approach to non-linear scoring functions in the following.

3.1.2 Deep PropDCG

Since moving to non-linear ranking functions through SVM kernelization is challenging, we instead explore deep networks as a class of non-linear scoring functions. Specifically, we replace the linear scoring function $f(\boldsymbol{x}, y) = w \cdot \phi(\boldsymbol{x}, y)$ with a neural network

$$f(\boldsymbol{x}, y) = NN_w[\phi(\boldsymbol{x}, y)] \tag{3.8}$$

This network is generally non-linear in both the weights w and the features $\phi(\boldsymbol{x}, y)$. However, this does not affect the validity of the hinge-loss upper bound from Equation (3.7), which now takes the form

$$\frac{1}{n}\sum_{j=1}^{n}\frac{1}{q_i}\lambda\left(1+\sum_{\substack{y\in Y_i\\ y\neq y_i}}\max(1-(NN_w[\phi(\boldsymbol{x}_i, y_i)]-NN_w[\phi(\boldsymbol{x}_i, y)]), 0)\right)$$

During training, we need to minimize this function with respect to the network parameters w. Unlike in the case of SVM PropDCG, this function can no longer be expressed as the difference of a convex and a concave function, since $NN_w[\phi(\boldsymbol{x}_i, y_i)]$ is neither convex nor concave in general. Nevertheless, the empirical success of optimizing non-convex $NN_w[\phi(\boldsymbol{x}_i, y_i)]$ via gradient descent to a local optimum is well documented, and we will use this approach in the following. This is possible since the training objective is subdifferentiable as long as the weighting function $\lambda(r)$ is differentiable. However, the non-linearity of $\lambda(r)$ adds a challenge in applying *stochastic* gradient descent methods to our training objective, since the objective no longer decomposes into a sum over all $(\boldsymbol{x}_i, y)$ as in standard network training. We discuss in the following how to handle this situation to arrive at an efficient stochastic-gradient procedure.

For concreteness, we again focus on the case of optimizing DCG via $\lambda(r) = \frac{-1}{\log(1+r)}$. In particular, plugging in the weighting function for DCG, we get the Deep PropDCG minimization objective

$$\frac{1}{n}\sum_{j=1}^{n}\frac{-1}{q_i}\log^{-1}\left(2+\sum_{\substack{y\in Y_i\\ y\neq y_i}}\max(1-(NN_w[\phi(\boldsymbol{x}_i, y_i)]-NN_w[\phi(\boldsymbol{x}_i, y)]), 0)\right)$$

to which a regularization term can be added (our implementation uses weight decay).

Since the weighting function ties together the hinge losses from pairs of documents in a non-linear way, stochastic gradient descent (SGD) is not directly feasible at the level of individual documents. In the case of DCG, since the rank weighting function is concave, one possible workaround is a Majorization-Minimization scheme [50] (akin to CCP): upper bound the loss function with a linear Taylor approximation at the current neural net weights, perform SGD at the level of document pairs (y_i, y) to update the weights, and repeat until convergence.

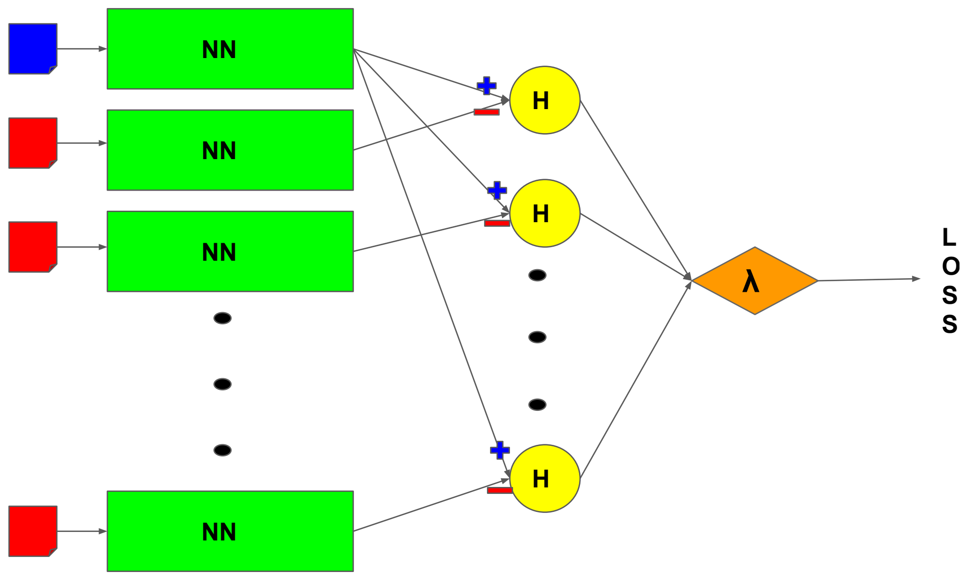


Figure 3.1: Deep PropDCG schema for computing the loss from one query instance. The blue document is the positive (clicked) result, and the red documents are the other candidates. The neural net NN is used to compute document scores for each set of candidate features. Pairs of scores are passed through the hinge node, and then finally the weighting function is applied as shown.

While this Majorization-Minimization scheme in analogy to the SVM approach is possible also for deep networks, we chose a different approach for the reasons given below. In particular, given the success of stochastic-gradient training of deep networks in other settings, we directly perform stochastic-gradient updates at the level of query instances, not individual $(\boldsymbol{x}_i, y)$. At the level of query instances, the objective does decompose linearly such that any subsample of query instances can provide an unbiased gradient estimate. Note that this approach works for any differentiable weighting function $\lambda(r)$, does not require any alternating approximations as in Majorization-Minimization, and processes each candidate document y including the clicked document y_i only once in one SGD step.

For SGD at the level of query instances, a forward pass of the neural network

– with the current weights fixed – must be performed on each document y in candidate set Y_i in order to compute the loss from training instance $(\boldsymbol{x}_i, y_i)$. Since the number of documents in each candidate set varies, this is best achieved by processing each input instance (including the corresponding candidate set) as a (variable-length) sequence so that the neural net weights are effectively shared across candidate documents for the same query instance.

This process is most easily understood via the network architecture illustrated in Figure 3.1. The scoring function $NN_w[\phi(\boldsymbol{x}_i, y_i)]$ is replicated for each result in the candidate set using shared weights w. In addition there is a hinge-loss node $H(u, v) = \max(1 - (u - v), 0)$ that combines the score of the clicked result with each other result in the candidate set Y_i. For each such pair (y_i, y), the corresponding hinge-loss node computes its contribution h_j to the upper bound on the rank. The result of the hinge-loss nodes then feeds into a single weighting node $\Lambda(\vec{h}) = \lambda\left(1 + \sum_j h_j\right)$ that computes the overall bound on the rank and applies the weighting function. The result is the loss of that particular query instance.

Note that we have outlined a very general method which is agnostic about the size and architecture of the neural network. As a proof-of-concept, we achieved superior empirical results over a linear scoring function even with a simple two layer neural network, as seen in Section 3.2.8. We conjecture that DCG performance may be enhanced further with deeper, more specialized networks. Moreover, in principle, the hinge-loss nodes can be replaced with nodes that compute any other differentiable loss function that provides an upper bound on the rank without fundamental changes to the SGD algorithm.

Dataset	# Avg. train clicks	# Train queries	# Features
Yahoo	173,986	20,274	699
LETOR4.0	25,870	1,484	46

Table 3.2: Properties of the two benchmark datasets.

3.2 Empirical Evaluation

While the derivation of SVM PropDCG and Deep PropDCG has provided a theoretical justification for both methods, it still remains to show whether this theoretical argument translates to improved empirical performance. To this effect, the following empirical evaluation addresses three key questions.

First, we investigate whether directly optimizing DCG improves performance as compared to baseline methods, in particular, SVM PropRank as the most relevant method for unbiased LTR from implicit feedback, as well as LambdaRank, a common strong non-linear LTR method. Comparing SVM PropDCG to SVM PropRank is particularly revealing about the importance of direct DCG optimization, since both methods are linear SVMs and employ the same software machinery for the Quadratic Programs involved, thus eliminating any confounding factors. We also experimentally analyze the CCP optimization procedure to see whether SVM PropDCG is practical and efficient. Second, we explore the robustness of the generalized counterfactual LTR approach to noisy feedback, the severity of the presentation bias, and misspecification of the propensity model. And, finally, we compare the DCG performance of Deep PropDCG with a simple two layer neural network against the linear SVM PropDCG to understand to what extent non-linear models can be trained effectively using the generalized counterfactual LTR approach.

Model	Avg. DCG (Yahoo)	Avg. DCG (LETOR4.0)
SVM Rank	0.6223 ± 8e-4	0.6841 ± 2e-3
LambdaRank	0.6435 ± 4e-4	0.6915 ± 4e-3
SVM PropRank	0.6410± 1e-3	0.7004 ± 1e-2
SVM PropDCG	0.6468± 2e-3	0.7043 ± 1e-2
Deep PropDCG	**0.6517 ± 4e-4**	**0.7244 ± 4e-3**

Table 3.3: Performance comparison of different methods on two benchmark datasets ($\eta = 1$, $\epsilon_- = 0.1$, $\epsilon_+ = 1$).

3.2.1 Setup

We conducted experiments on synthetic click data derived from two major LTR datasets, the Yahoo Learning to Rank Challenge corpus and LETOR4.0 [44]. LETOR4.0 contains two separate corpora: MQ2007 and MQ2008. Since MQ2008 is significantly smaller than Yahoo Learning to Rank Challenge, with only 784 queries, we follow the data augmentation approach proposed in [42], combining the MQ2007 and MQ2008 train sets for training and using the MQ2008 validation and test sets for validation and testing respectively.

Our experiment setup matches [33] for the sake of consistency and reproducibility. Briefly, the training and validation click data were generated from the respective full-information datasets (with relevances binarized) by simulating the position-based click model. Following [33], we use propensities that decay with presented rank of the result as $p_r = (\frac{1}{r})^\eta$. The rankings that generate the clicks are given by a "production ranker" which was a conventional Ranking SVM trained on 1 percent of the full-information training data. The parameter η controls the severity of bias, with higher values causing greater position bias.

We also introduced noise into the clicks by allowing some irrelevant documents to be clicked. Specifically, an irrelevant document ranked at position r by the production ranker is clicked with probability p_r times ϵ_-. When not mentioned otherwise, we used the parameters $\eta = 1$, $\epsilon_- = 0.1$ and $\epsilon_+ = 1$, which is consistent with the setup used in [33]. Other bias profiles are also explored in the following.

Both the SVM PropRank and SVM PropDCG models were trained and cross-validated to pick the regularization constant C. For cross-validation, we use the partial feedback data in the validation set and select based on the IPS estimate of the DCG [51]. The performance of the models is reported on the binarized fully labeled test set which is never used for training or validation.

3.2.2 How do SVM PropDCG and Deep PropDCG compare against baselines?

We begin the empirical evaluation by comparing our counterfactual LTR methods again standard methods that follow a conventional ERM approach, namely LambdaRank and SVM-Rank. We generate synthetic click data using the procedure describe above, iterating over the training set 10 times for the Yahoo dataset and 100 times for MQ2008. This process was repeated over 6 independent runs, and we report the average performance along with the standard deviation over these runs. The regularization constant C for all SVM methods was picked based on the average DCG performance across the validation click data sampled over the 6 runs. Table 3.2 shows the average number of clicks along with other information about the training sets.

As a representative for non-linear LTR methods that use a conventional ERM approach, we also conducted experiments with LambdaRank as one of the most

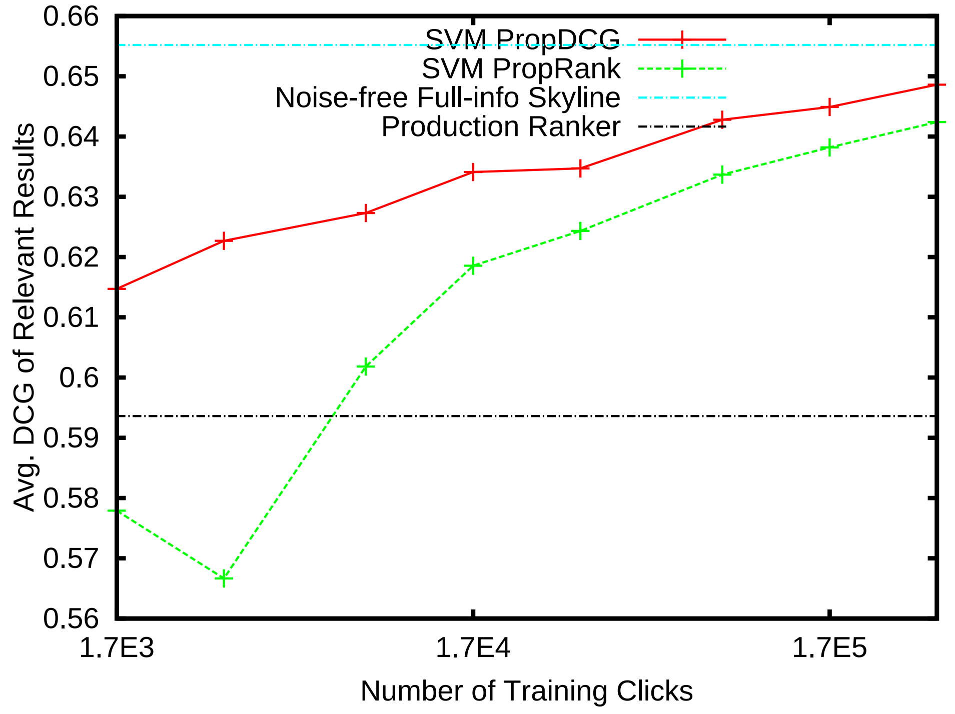


Figure 3.2: Test set Avg DCG performance for SVM PropDCG and SVM PropRank ($\eta = 1$, $\epsilon_{-} = 0.1$)

popular tree-based rankers. We use the LightGBM implementation [34]. During training, LambdaRank optimizes Normalized Discounted Cumulative Gain (NDCG). Since LambdaRank is a full-information method, we used clicks as relevance labels, i.e. all clicked documents as relevant and all non-clicked documents as irrelevant. The hyperparameters for LambdaRank, namely learning rate and the number of leaves were tuned based on the average DCG of clicked documents in the validation sets. More specifically, we performed a grid search to finetune learning rate from 0.001 to 0.1 and the number of leaves from 2 to 256. After tuning, we selected the learning rate to be 0.1, and the number of leaves to be 64 for the Yahoo dataset and 4 for MQ2008. We also made sure each split does not use more than 50% of the input features.

As shown in in Table 3.3, the counterfactual ERM approach via IPS weighting and directly optimizing for the target metric DCG yield superior results for

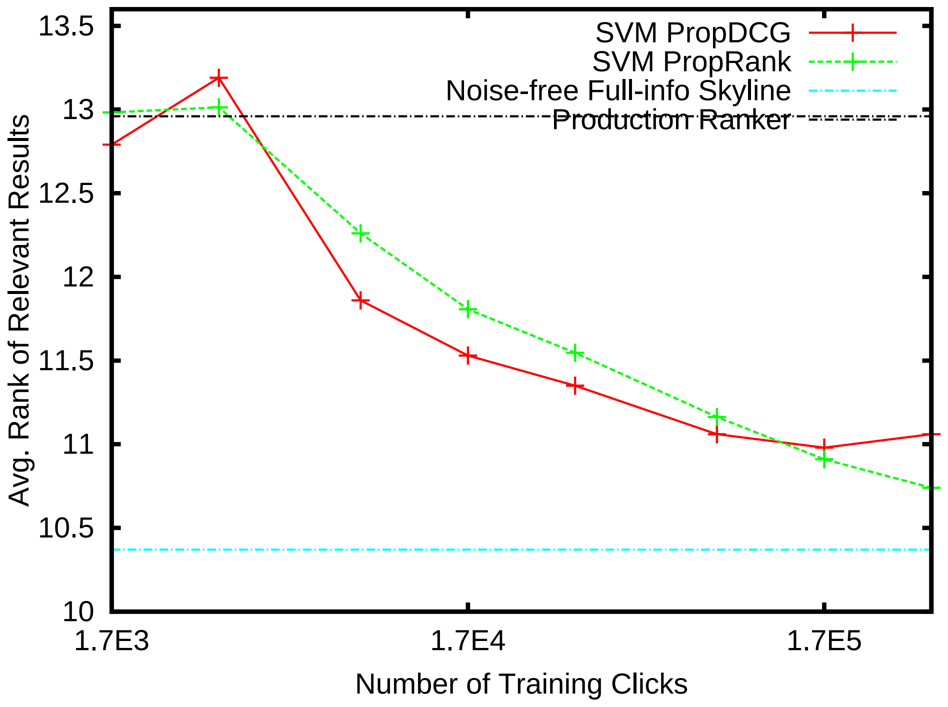


Figure 3.3: Test set Avg Rank performance for SVM PropDCG and SVM PropRank ($\eta = 1$, $\epsilon_{-} = 0.1$)

SVM PropDCG and Deep PropDCG. The best results on both benchmarks are achieved by Deep PropDCG, which learns a two-layer neural network ranker. We conjecture that more sophisticated network architectures can further improve performance.

3.2.3 How does ranking performance scale with training set size?

Next, we explore how the test-set ranking performance changes as the learning algorithm is given more and more click data. The resulting learning curves are given in Figures 3.2 and 3.3. The click data has presentation bias with $\eta = 1$ and noise with $\epsilon_{-} = 0.1$. For small datasets, results are averaged over 3 draws

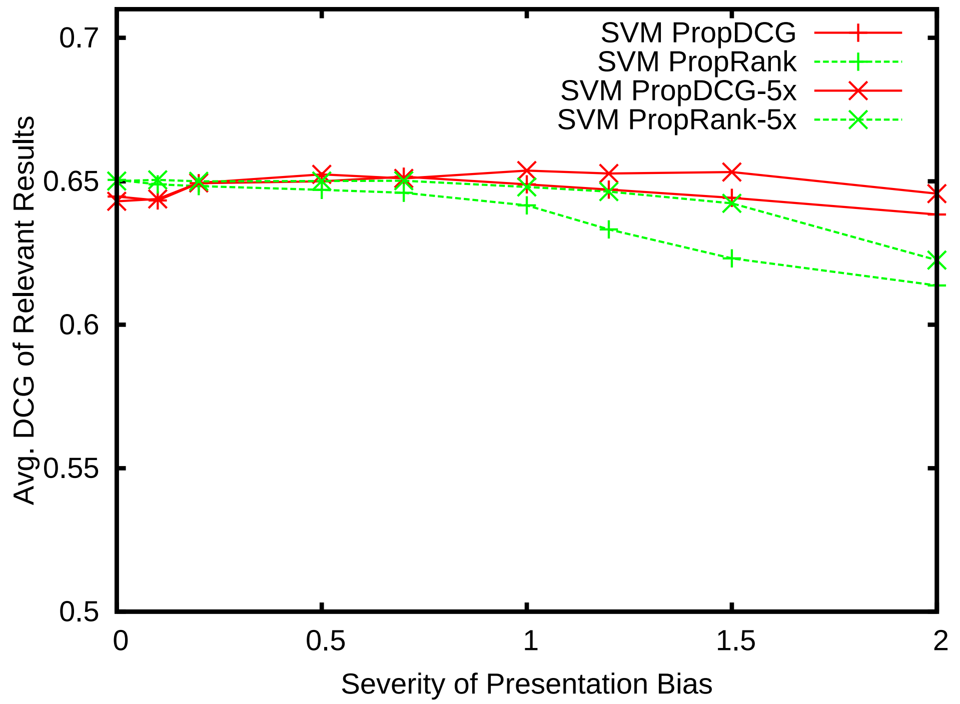


Figure 3.4: Test set Avg DCG performance for SVM PropDCG and SVM PropRank as presentation bias becomes more severe in terms of η ($n = 45K$ and $n = 225K$, $\epsilon_{-} = 0$).

of the click data. Both curves show the performance of the Production Ranker used to generate the click data, and the SVM skyline performance trained on the full-information training set. Ideally, rankers trained on click data should outperform the production ranker and approach the skyline performance.

Figure 3.2 shows that the DCG performance of both SVM PropDCG and SVM PropRank. As expected, both improve with increasing amounts of click data. Moreover, SVM PropDCG performs substantially better than the baseline SVM PropRank in maximizing test set DCG.

More surprisingly, Figure 3.3 shows both methods perform comparably in minimizing the average rank metric, with SVM PropDCG slightly better at smaller amounts of data and SVM PropRank better at larger amounts. We conjecture that this is due the variance-limiting effect of the DCG weights in SVM

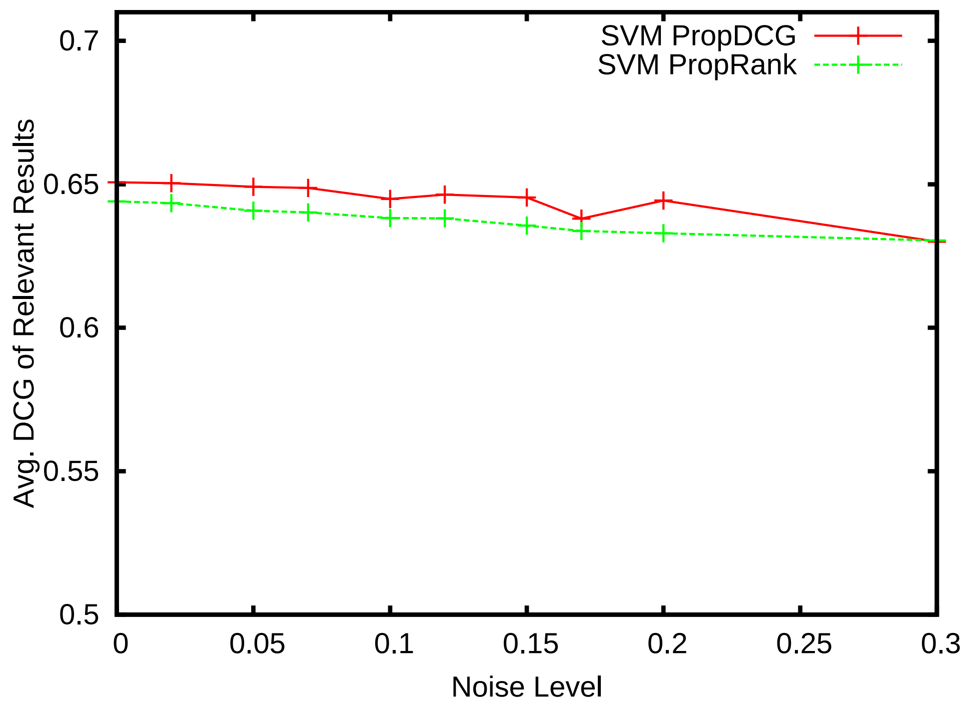


Figure 3.5: Test set Avg DCG performance for SVM PropDCG and SVM PropRank as the noise level increases in terms of ϵ ($n = 170K, \eta = 1$).

PropDCG when substituting the propensity weights q_i with the new constants q_i' in the SVM PropDCG CCP iterations. This serves as implicit variance control in the IPS estimator similar to clipping [33] by preventing propensity weights from getting too big. Since variance dominates estimation error at small amounts of data and bias dominates at large amounts, our conjecture is consistent with the observed trend.

3.2.4 How much presentation bias can be tolerated?

We now vary the severity of the presentation bias via η – higher values leading to click propensities more skewed to the top positions – to understand its impact on the learning algorithm. Figure 3.4 shows the impact on DCG performance for both methods. We report performance for two training set sizes that differ

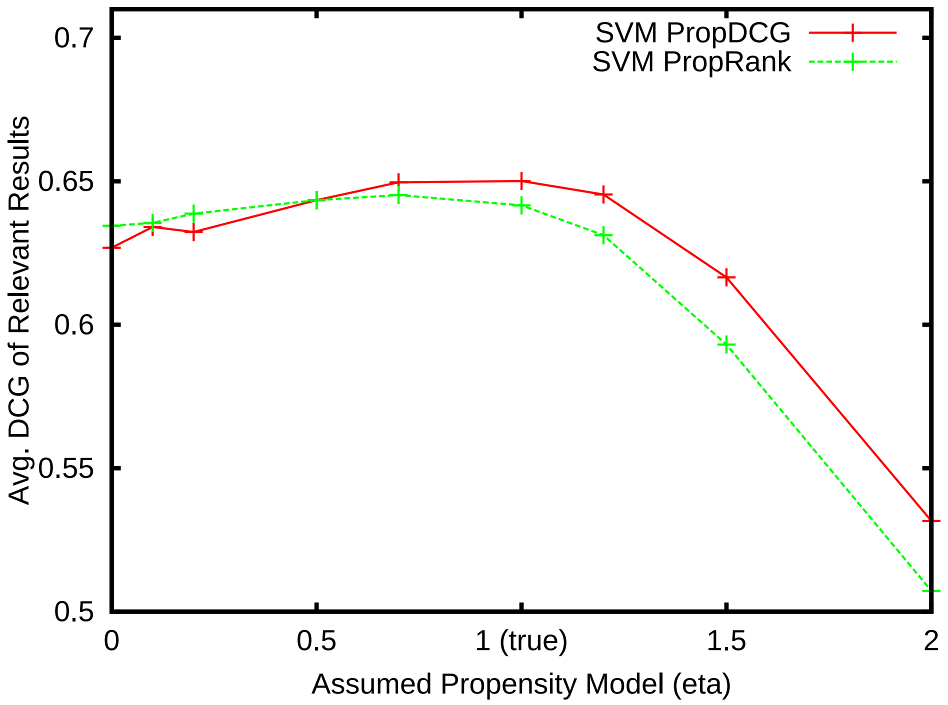


Figure 3.6: Test set Avg DCG performance for SVM PropDCG and SVM PropRank as propensities are misspecified (true $eta = 1, n = 170K, \epsilon_{-} = 0.1$).

by a factor of 5 (noise $\epsilon_{-} = 0$). We see that SVM PropDCG is at least as robust to the severity of bias as SVM PropRank. In fact, SVM PropRank's performance degrades more at high bias than that of SVM PropDCG, further supporting the conjecture that the DCG weighting in SVM PropDCG provides improved variance control which is especially beneficial when propensity weights are large. Furthermore, as also noted for SVM PropRank in [33], increasing the amount of training data by a factor of 5 improves performance of both methods due to variance reduction, which is an advantage that unbiased learning methods have over those that optimize a biased objective.

3.2.5 How robust is SVM PropDCG to noise?

Figure 3.5 shows the impact of noise on DCG performance, as noise levels in terms of ϵ_{-} increase from 0 to 0.3. The latter results in click data where 59.8% of all clicks are on irrelevant documents. As expected, performance degrades for both methods as noise increases. However, there is no evidence that SVM PropDCG is less robust to noise than the baseline SVM PropRank.

3.2.6 How robust is SVM PropDCG to misspecified propensities?

So far all experiments have had access to the true propensities that generated the synthetic click data. However, in real-world settings propensities need to be estimated and are necessarily subject to modeling assumptions. So, we evaluate the robustness of the learning algorithm to propensity misspecification.

Figure 3.6 shows the performance of SVM PropDCG and SVM PropRank when the training data is generated with $\eta = 1$, but the propensities used in learning are misspecified according to the η on the x-axis. The results show that SVM PropDCG is at least as robust to misspecified propensities as SVM PropRank. Both methods degrade considerably in the high bias regime when small propensities are underestimated – this is often tackled by clipping [33]. It is worth noting that SVM PropDCG performs better than SVM PropRank when misspecification leads to propensities that are underestimated, further strengthening the implicit variance control conjecture for SVM PropDCG discussed above.

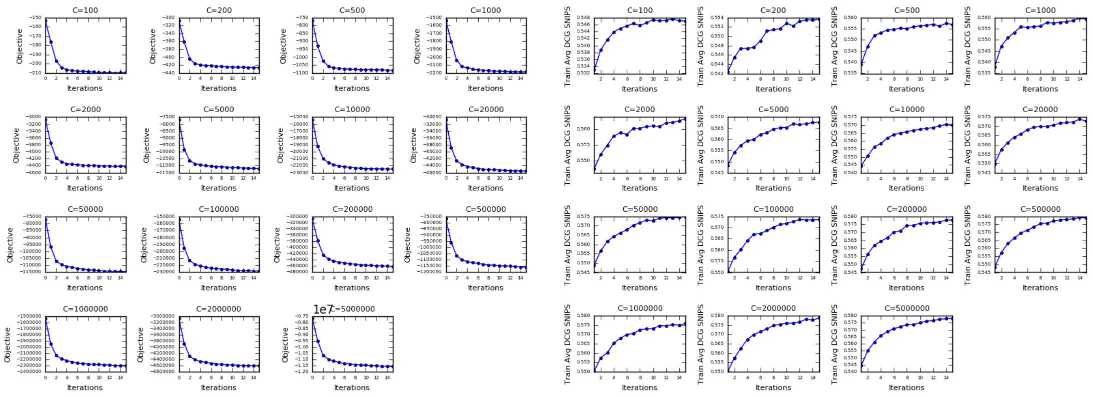


Figure 3.7: Optimization progress with respect to the number of CCP iterations. The objective value is shown in the left plots, and the training set DCG estimate on the right plots. Each plot corresponds to a particular value of regularization constant C ($n = 17K$, $\eta = 1$, $\epsilon_{-} = 0.1$).

3.2.7 How well does the CCP converge?

Next, we consider the computational efficiency of employing the CCP optimization procedure for training SVM PropDCG. Recall that the SVM PropDCG objective is an upper bound on the regularized (negative) DCG IPS estimate. It is optimized via CCP which repeatedly solves convex subproblems using the SVM PropRank solver until the objective value converges.

In Figure 3.7, optimization progress vs number of iterations as indicated by the change in objective value as well as the training DCG SNIPS estimate [51] is shown for $17K$ training clicks and the full range of regularization parameter C used in validation. The figure shows that the objective value usually converges in 3-5 iterations, a phenomenon observed in our experiments for other amounts of training data as well. In fact, the convergence tends to take slightly fewer iterations for larger amounts of data. The figure also shows that progress in objective is well-tracked with progress in the training DCG estimate, which suggests that the objective is a suitable upper bound for DCG optimization.

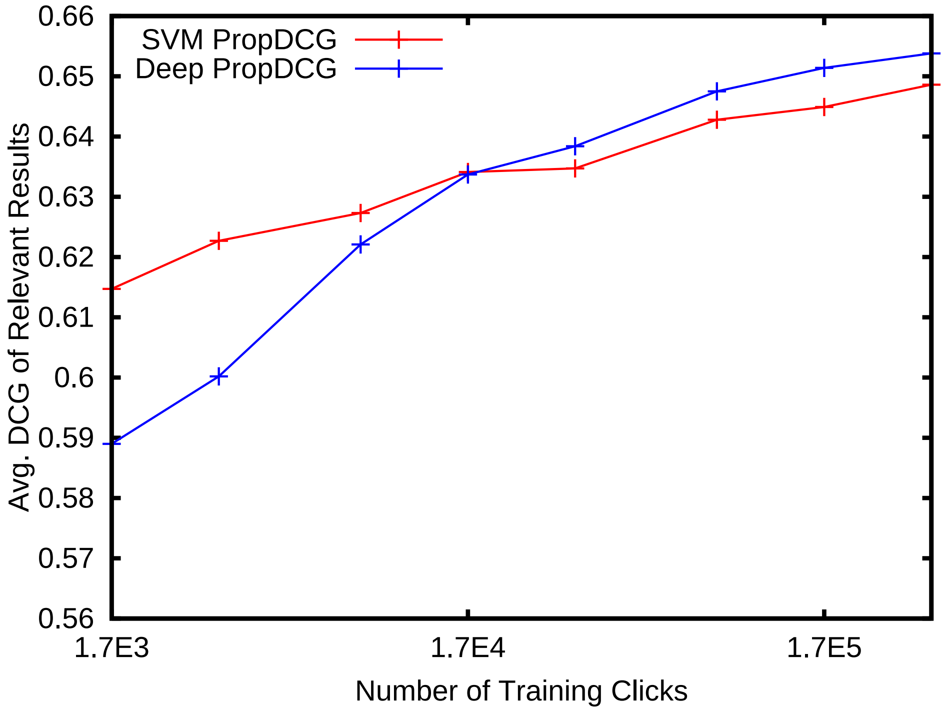


Figure 3.8: Test set Avg DCG performance for SVM PropDCG and Deep PropDCG ($\eta = 1$, $\epsilon_{-} = 0.1$)

It is worth noting that restarting the optimizer across multiple CCP iterations can be substantially less time consuming than the initial solution that SVM PropRank computes. Since only the coefficients of the Quadratic Program change, the data does not need to be reloaded and the optimizer can be warm-started for quicker convergence in subsequent CCP iterations.

3.2.8 When does the non-linear model improve over the linear model?

We have seen that SVM PropDCG optimizes DCG better than SVM PropRank, and that it is a robust method across a wide range of biases and noise levels. Now we explore if performance can be improved further by introducing non-linearity via neural networks. Since the point of this chapter is not a specific deep architecture but a novel training objective, we used a simple two-layer neural network with 200 hidden units and sigmoid activation. We expect that specialized deep architectures will further improve performance.

Figure 3.8 shows that Deep PropDCG achieves improved DCG compared to the linear SVM PropDCG given enough training data. For small amounts of training data, the linear model performs better, which is to be expected given the greater robustness to overfitting of linear models.

We also expect improved performance from tuning the hyperparameters of Deep PropDCG. In fact, we only used default parameters for Deep PropDCG, while we optimized the hyperparameters of SVM PropDCG on the validation set. In particular, Adam was used for stochastic gradient descent with weight decay regularizer at 10^{-6}, minibatch size of 1000 documents and 750 epochs. The learning rate began at 10^{-6} for the first 300 epochs, dropping by one order of magnitude in the next 200 epochs and another order of magnitude in the remaining epochs. We did not try any other hyperparameter settings and these settings were held fixed across varying amounts of training data.

3.3 Conclusion

In this chapter, we proposed a counterfactual learning-to-rank framework that is broad enough to cover a broad class of additive IR metrics as well as non-linear deep network models. Based on the generalized framework, we developed the SVM PropDCG and Deep PropDCG methods that optimize DCG via the Convex-Concave Procedure (CCP) and stochastic gradient descent respectively. We found empirically that SVM PropDCG performs better than SVM PropRank in terms of DCG, that it is robust to a substantial amount of presentation bias, noise and propensity misspecification, and that it can be optimized efficiently. DCG was improved further by using a neural network in Deep PropDCG.

CHAPTER 4

ESTIMATING POSITION BIAS WITHOUT INTRUSIVE INTERVENTIONS

The key prerequisite for the unbiased LTR framework presented in the previous chapter is estimation of the propensity of obtaining a particular feedback signal, which enables Propensity-Weighted ERM. Towards this end, we propose the first approach for producing consistent propensity estimates without manual relevance judgments, disruptive interventions, or restrictive relevance-modeling assumptions in [5].

4.1 Harvesting Interventions

We approach the propensity estimation task by harvesting implicit interventions from already logged data. In particular, we make use of the fact that we typically have data from multiple historic ranking functions, and we will identify the conditions under which these provide unconfounded intervention data. We focus on the Position-Based Propensity Model (PBM), which we review before defining interventional sets and analyzing their properties.

4.1.1 Position-Based Propensity Model

The position-based model recognizes that higher-ranked results are more likely to be considered (i.e. discovered and viewed) by the user than results further down the ranking. Suppose that for a particular query q, result d is displayed at position k. Let C be the random variable corresponding to a user clicking on d, and let E be the random variable denoting whether the user examines d. In our notation, q represents all information about the users, the query context, and which documents the user considers relevant or not. This mean we can denote the relevance of an individual document as a non-random function $\mathrm{rel}(q, d)$ of q, where $\mathrm{rel}(q, d) = 1$ and $\mathrm{rel}(q, d) = 0$ indicates relevant and non-relevant respectively. Then according to the Position-Based Propensity Model (PBM) [16],

$$\begin{aligned}\Pr(\mathrm{C} = 1|q, d, k) &= \Pr(E = 1|k)\mathrm{rel}(q, d)\\ &= p_k \mathrm{rel}(q, d).\end{aligned}$$

In this model, the examination probability $p_k := \Pr(E = 1|k)$ depends only on the position, and it is identical to the observation propensity [33]. For learning, it is sufficient to estimate relative propensities p_k/p_1 for each k [33], since multiplicative scaling does not change the training objective of counterfactual learning methods (e.g. [32, 33, 4, 6, 50, 51]). Estimating these relative propensities is the goal in this chapter.

Note that one can train multiple PBM models to account for changes in the propensity curve due to context. In this way, the expressiveness of the PBM model can be substantially extended. For example, one can train separate PBM models for navigational vs. informational queries simply by partitioning the data. While training multiple such models is prohibitively expensive when intrusive interventions are required, the intervention harvesting approach we describe below makes training such contextual PBMs feasible since it does not require costly data, is both statistically and computationally efficient, and does not have any parameters that require manual tuning.

4.1.2 Controlling for Relevance through Swap Interventions

The key problem in both propensity estimation and unbiased learning to rank is that we only observe clicks C, but we never get to observe E (whether a user examined a result) and $\mathrm{rel}(q, d)$ (whether the user found d relevant for q) individually. This makes it difficult to attribute the lack of a click to a lack of examination or a lack of relevance. A key idea for overcoming this dilemma without explicit relevance judgments or cumbersome instrumentation (e.g. eye tracking) was proposed in [33], namely to control for relevance through randomized interventions.

The intervention proposed in [33] is to randomly swap the result in position 1 with the result in position k. We call these $\mathit{Swap}(1, k)$-interventions. Such swaps provide a completely randomized experiment [26], meaning that the assignment of the document to a position does not depend on relevance or any covariates (e.g. abstract length, document language). Under these swap interventions, we now get to observe how many clicks position 1 results get when they stay in position 1 vs. when they get swapped into position k. Since the expected relevance in either condition (i.e. swap vs. not swapped) is the same, any change in clickthrough rate must be proportional to a drop in examination.

More formally, we model user queries as sampled i.i.d. $q \sim \Pr(Q)$. Whenever a query is sampled, the ranker $f(q)$ sorts the candidate results d for the query and we apply the randomized swap intervention between positions 1 and some fixed k before the ranking is displayed to the user. As a precursor to the later exposition, suppose that the random swap occurs with a fixed probability p (not necessarily 0.5), yielding the logged datasets $D_1^{1,k} = (q_1^i, d_1^i, C_1^i)^{n_1}$ (result stayed in position 1) and $D_k^{1,k} = (q_k^j, d_k^j, C_k^j)^{n_2}$ (result was swapped into position k) of sizes n_1

and n_2 respectively. Here C_1^i denotes whether the document d_1^i placed at position 1 was clicked or not, and similarly for C_k^j. Denote with $\hat{c}_1^{1,k} = \frac{1}{n_1}\sum C_1^i$ the rate of clicks that documents get when they remain in position 1 and let $\hat{c}_k^{1,k} = \frac{1}{n_2}\sum C_k^j$ be the rate of clicks when they get swapped to position k. Then, under the PBM, the relative propensity is equal to the ratio of expected click rates:

$$\begin{aligned}\frac{p_k}{p_1} &= \frac{p_k E_{D_k^{1,k}}\left[\sum \mathrm{rel}(q_k^j, d_k^j)/n_2\right]}{p_1 E_{D_1^{1,k}}\left[\sum \mathrm{rel}(q_1^i, d_1^i)/n_1\right]} \\ &= \frac{E_{D_k^{1,k}}\left[\sum p_k \mathrm{rel}(q_k^j, d_k^j)/n_2\right]}{E_{D_1^{1,k}}\left[\sum p_1 \mathrm{rel}(q_1^i, d_1^i)/n_1\right]} \\ &= \frac{E_{D_k^{1,k}}\left[\sum \Pr(C_k^j = 1 | q_k^j, d_k^j, k)/n_2\right]}{E_{D_1^{1,k}}\left[\sum \Pr(C_1^i = 1 | q_1^i, d_1^i, 1)/n_1\right]} \\ &= \frac{E_{D_k^{1,k}}\left[\hat{c}_k^{1,k}\right]}{E_{D_1^{1,k}}\left[\hat{c}_1^{1,k}\right]}\end{aligned}$$

The first equality holds since swaps are completely at random, such that the expected average relevance under each condition is equal and their ratio is one (we need not consider n_1 and n_2 as random variables). It is thus reasonable to estimate the relative propensity as

$$\frac{\hat{p}_k}{\hat{p}_1} = \frac{\hat{c}_k^{1,k}}{\hat{c}_1^{1,k}},$$

which is a statistically consistent estimate as the sample size n grows [33]. Analogously, relative propensities can be estimated between any pairs of positions k and k' with *Swap*(k, k') interventions [55].

4.1.3 Interventional Sets from Multiple Rankers

A key shortcoming of the swap experiment is its impact on user experience, since retrieval performance can be degraded quite substantially. This is especially true for swaps between position 1 and k for large k, even though this particular swap experiment makes it easy to estimate propensities relative to position 1 directly. To overcome this impact on user experience, we now show how to harvest interventions from already existing data under mild assumptions, so that no additional swap experiments are needed. As part of this, we may never see a direct swap between position 1 and k, and we tackle the problem of how to aggregate many local swaps into an overall consistent estimate in Section 4.2.

Our key idea in harvesting swap interventions lies in the use of data from multiple historic rankers. Logs from multiple rankers are typically available in operational systems, since multiple rankers may be fielded at the same time in A/B tests, or when the production ranker is updated frequently. Consider the case where we have data from m historic rankers $F = \{f_1, ..., f_m\}$. Furthermore, a crucial condition is that the query distribution must not depend on the choice of ranker f_i (where the query q includes the user's relevance vector rel in our notation),

$$\forall f_i : \Pr(Q|f_i) = \Pr(Q) \Rightarrow \forall q \in Q : \Pr(f_i|q) = \Pr(f_i) \tag{4.1}$$

Note that the condition on the left implies that $\Pr(f_i|q) = \Pr(f_i)$ on the right by Bayes rule, which is related to exploration scavenging [35]. Intuitively, this condition avoids that different rankers f_i get different types of queries. For rankers that are compared in an A/B test, this condition is typically fulfilled by construction since the assignment of queries to rankers is randomized. For data from a sequence of production rankers, one needs to be mindful that any temporal covariate shift in the query and user distribution is acceptably small.

We denote the click log of each ranker f_i with $\mathcal{D}_i = (q_i^j, y_i^j, \mathrm{c}_i^j)^{n_i}$, where n_i is the number of queries that f_i processed. Here $j \in [n_i]$, q_i^j is a query, $y_i^j = f_i(q_i^j)$ is the presented ranking, and c_i^j is a vector that indicates for each document click or no click. Since most retrieval systems only rerank a candidate set of documents in their final stage, we denote $\Omega(q)$ as the candidate set of results for query q. We furthermore denote the rank of candidate result d in ranking y_i^j as $\text{rank}(d|y_i^j)$, and we use $c_i^j(d) \in [0, 1]$ to denote whether result d was clicked or not.

We begin by defining *interventional sets* $S_{k,k'}$ of query-document pairs, where one ranking function $f \in F$ puts document d at position k and another ranker $f' \in F$ puts the same document d at position k' for the same query q. Let M be some fixed number of top positions for which propensity estimates are desired (e.g. $M = 10$). Then, for each two ranks $k \neq k' \in [M]$, we define the interventional set as

$$\begin{aligned} S_{k,k'} := \{(q, d) : q \in Q, d \in \Omega(q), \\ \exists f, f' \ \text{rank}(d|f(q)) = k \wedge \text{rank}(d|f'(q)) = k'\} \end{aligned}$$

Intuitively, the pairs in these sets are informative because they receive different treatments or interventions based on the choice of different rankers. But note that we are *not* requiring any query to occur multiple times. An interventional set merely reflects that two potential outcomes (i.e. document d either in position k or in position k' for query q) were possible. In fact, we only ever observe one factual outcome, while the other outcome remains counterfactual and unobserved.

The key insight is that for any pair of ranks (k, k'), the interventional set contains the query-document pairs (q, d) for which the rank of d was randomly assigned to either k or k' via the choice of ranking function f_i. Specifically, there are three possible outcomes depending on the choice of ranking function $f_i \in F$: (a) f_i puts d at rank k, (b) f_i puts d at rank k', or (c) f_i puts d at some other rank.

In the latter case, the instance is not included in the interventional set $S_{k,k'}$, but the former two cases can be seen as the two conditions of a swap experiment between positions k and k'. In this way, we can think about these as virtual swap interventions between ranks k and k' where the randomization comes from the randomized choice of ranking function according to (4.1).

While this swap experiment is completely randomized under condition (4.1) (i.e. the choice of f_i does not depend on q), the assignment is generally not uniform (i.e. d could have a higher probability to be presented in position k than in position k'). The weights

$$w(q, d, k) := \sum_{i=1}^{m} n_i \mathbb{1}[\text{rank}(d|f_i(q)) = k].$$

are then used to account for this non-uniformity. Specifically, the weight $w(q, d, k)$ reflects how often document d is ranked at position k given that we have employed each ranking function f_i exactly n_i times. Therefore, the probability of assigning d to k as opposed to k' for query q can be estimated as

$$P(rank = k | rank = k \text{ or } rank = k', q, d) = \frac{w(q, d, k)}{w(q, d, k) + w(q, d, k')}$$

We will show in the following how interventional sets can be used to control for unobserved relevance information when estimating propensities.

4.1.4 Controlling for Unobserved Relevance through Interventional Sets

As we had already seen in Section 4.1.2, we need to disentangle two unobserved quantities – relevance and examination – when analyzing observed clicks in the

PBM. This can be achieved by controlling for relevance through randomization, which was done explicitly in Section 4.1.2 via *Swap*$(1, k)$-interventions. The following shows that interventional sets $S_{k,k'}$ provide analogous control for any pair of ranks (k, k') under condition (4.1).

For each interventional set $S_{k,k'}$, with $k \neq k' \in [M]$, we can now define the quantities $\hat{c}_k^{k,k'}$ (and $\hat{c}_{k'}^{k,k'}$) which can be thought of as the rate of clicks in position k (and in position k'):

$$\hat{c}_k^{k,k'} := \sum_{i=1}^{m} \sum_{j=1}^{n_i} \sum_{d \in \Omega(q_i^j)} \mathbb{1}_{[(q_i^j, d) \in S_{k,k'}]} \mathbb{1}_{[\mathrm{rank}(d|\mathrm{y}_i^j) = k]} \frac{\mathrm{c}_i^j(d)}{w(q_i^j, d, k)}.$$

$$\hat{c}_{k'}^{k,k'} := \sum_{i=1}^{m} \sum_{j=1}^{n_i} \sum_{d \in \Omega(q_i^j)} \mathbb{1}_{[(q_i^j, d) \in S_{k,k'}]} \mathbb{1}_{[\mathrm{rank}(d|\mathrm{y}_i^j) = k']} \frac{\mathrm{c}_i^j(d)}{w(q_i^j, d, k')}.$$

The definition normalizes the observed clicks by dividing with $w(q_i^j, d, k)$, which accounts for non-uniform assignment probabilities. Note that $w(q_i^j, d, k)$ is non-zero whenever the first indicator is true, such that we never divide a non-zero quantity by zero (and we define $0/0 := 0$). Intuitively, $\hat{c}_k^{k,k'}$ and $\hat{c}_{k'}^{k,k'}$ capture the weighted click-through rate at position k and k' restricted to (k, k')-interventional (query, document) pairs, where the weights $w(q, d, k)$ account for the imbalance in applying the intervention of putting document d at position k vs k' for query q.

The following shows that $\hat{c}_k^{k,k'}$ and $\hat{c}_{k'}^{k,k'}$ are proportional to the true click-through rate at positions k and k' in expectation, conditioned on the number of relevant documents in the interventional set $S_{k,k'}$.

Proposition 1. *For the PBM model, i.i.d. queries $q \sim \Pr(Q)$, and under the condition* *in* (4.1), *the expectations of* $\hat{c}_k^{k,k'}$ *and* $\hat{c}_{k'}^{k,k'}$ *are*

$$\mathbb{E}_{q,\mathrm{c}}[\hat{c}_k^{k,k'}] = p_k r_{k,k'}$$
$$\mathbb{E}_{q,\mathrm{c}}[\hat{c}_{k'}^{k,k'}] = p_{k'} r_{k,k'}$$

where $r_{k,k'} = \mathbb{E}_q\left[\sum_{d\in\Omega(q)} \mathbb{1}_{[(q,d)\in S_{k,k'}]}\mathrm{rel}(q,d)\right]$.

Proof. We only detail the proof for $\hat{c}_k^{k,k'}$, since the proof for $\hat{c}_{k'}^{k,k'}$ is analogous.

$$\begin{aligned}
&\mathbb{E}_{q,\mathrm{c}}[\hat{c}_k^{k,k'}] \\
&= \sum_{i=1}^{m}\sum_{j=1}^{n_i}\sum_{q\in\mathrm{Q}}\Pr(q)\sum_{d\in\Omega(q)}\mathbb{1}_{[(q,d)\in S_{k,k'}]}\mathbb{1}_{[\mathrm{rank}(d|f_i(q))=k]}\frac{\mathbb{E}_{\mathrm{c}}[\mathrm{c}(d)]}{w(q,d,k)} \\
&= \sum_{i=1}^{m}\sum_{j=1}^{n_i}\sum_{q\in\mathrm{Q}}\Pr(q)\sum_{d\in\Omega(q)}\mathbb{1}_{[(q,d)\in S_{k,k'}]}\mathbb{1}_{[\mathrm{rank}(d|f_i(q))=k]}\frac{p_k\mathrm{rel}(q,d)}{w(q,d,k)} \\
&= p_k\sum_{q\in\mathrm{Q}}\Pr(q)\sum_{d\in\Omega(q)}\mathbb{1}_{[(q,d)\in S_{k,k'}]}\mathrm{rel}(q,d)\frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i}\mathbb{1}_{[\mathrm{rank}(d|f_i(q))=k]}}{w(q,d,k)} \\
&= p_k\mathbb{E}_q[\sum_{d\in\Omega(q)}\mathbb{1}_{[(q,d)\in S_{k,k'}]}\mathrm{rel}(q,d)\frac{\sum_{i=1}^{m}n_i\mathbb{1}_{[\mathrm{rk}(d|f_i(q))=k]}}{w(q,d,k)}] \\
&= p_k\mathbb{E}_q[\sum_{d\in\Omega(q)}\mathbb{1}_{[(q,d)\in S_{k,k'}]}\mathrm{rel}(q,d)]
\end{aligned}$$

The first equality follows from the i.i.d. assumption of condition (4.1) and that $y_i^j = f_i(q_i^j)$ by definition, the second from the PBM definition, and the third by taking the inner terms common. □

The quantity $r_{k,k'}$ is related to the average relevance of the documents in the interventional set $S_{k,k'}$ (also see Section 4.2.2). While $r_{k,k'}$ is unobserved, it is shared between both $\hat{c}_k^{k,k'}$ and $\hat{c}_{k'}^{k,k'}$. We can thus get the relative propensity between positions k and k' as

$$\frac{p_k}{p_k'} = \frac{p_k r_{k,k'}}{p_{k'} r_{k,k'}} = \frac{\mathbb{E}_{q,\mathrm{c}}[\hat{c}_k^{k,k'}]}{\mathbb{E}_{q,\mathrm{c}}[\hat{c}_{k'}^{k,k'}]}.$$

We are now in a position to define specific estimators for the relative propensities based on the interventional sets.

4.2 Propensity Estimators for Interventional Sets

This section defines relative propensity estimators that use interventional sets. The first set of estimators, which we call local estimators, are straightforward adaptations of estimators that had been proposed for data from explicit interventions [33, 55]. However, these local estimators ignore much of the available information when harvesting interventions, and we thus develop a new global estimator that exploits information from all interventional sets $S_{k,k'}$.

4.2.1 Local Estimators

Since the click counts $\hat{c}_k^{k,k'}$ and $\hat{c}_{k'}^{k,k'}$ can be treated just like data from an explicit intervention, the same estimators apply. In particular, we observe the interventional sets $S_{1,k}$ and can thus use the same estimator that we previously used for the explicit *Swap*$(1, k)$ experiment [33]. We call this the **PivotOne** estimator

$$\frac{\hat{p}_k}{\hat{p}_1} = \frac{\hat{c}_k^{1,k}}{\hat{c}_1^{1,k}}$$

We can similarly adapt the estimator used in [55], which uses a chain of swaps between adjacent positions in the ranking. We call this the **AdjacentChain** estimator

$$\frac{\hat{p}_k}{\hat{p}_1} = \frac{\hat{c}_2^{1,2}}{\hat{c}_1^{1,2}} \cdot \frac{\hat{c}_3^{2,3}}{\hat{c}_2^{2,3}} \cdot \ldots \cdot \frac{\hat{c}_k^{k-1,k}}{\hat{c}_{k-1}^{k-1,k}}$$

It is easy to see that both estimators are statistically consistent under mild conditions, most importantly that each relevant interventional set has non-zero support such that $\hat{c}_k^{k,k'}$ and $\hat{c}_{k'}^{k,k'}$ concentrate to their expectations.

4.2.2 Global AllPairs Estimator

A key shortcoming of the local estimators is that they only use a small part of the available information, and that they ignore the data from most interventional sets $S_{k,k'}$. To overcome this shortcoming, we developed a new extremum estimator called **AllPairs** that resembles a maximum likelihood objective over all interventional sets. In order to formulate the training objective of the AllPairs estimator, we first need to define a quantity that is analogous to $\hat{c}_k^{k,k'}$ and $\hat{c}_{k'}^{k,k'}$, but counting the non-click events:

$$\neg\hat{c}_k^{k,k'} := \sum_{i=1}^{m}\sum_{j=1}^{n_i}\sum_{d\in\Omega(q_i^j)} \mathbb{1}_{[(q_i^j,d)\in S_{k,k'}]}\mathbb{1}_{[\mathrm{rank}(d|\mathrm{y}_i^j)=k]}\frac{1-\mathrm{c}_i^j(d)}{w(q_i^j,d,k)}$$

$$\neg\hat{c}_{k'}^{k,k'} := \sum_{i=1}^{m}\sum_{j=1}^{n_i}\sum_{d\in\Omega(q_i^j)} \mathbb{1}_{[(q_i^j,d)\in S_{k,k'}]}\mathbb{1}_{[\mathrm{rank}(d|\mathrm{y}_i^j)=k']}\frac{1-\mathrm{c}_i^j(d)}{w(q_i^j,d,k')}.$$

Analogous to $\hat{c}_k^{k,k'}$ and $\hat{c}_{k'}^{k,k'}$, both quantities have the desired expectation.

Proposition 2. *For the PBM model, i.i.d. queries $q \sim \Pr(Q)$, and under the condition in (4.1), the expectations of $\neg\hat{c}_k^{k,k'}$ and $\neg\hat{c}_{k'}^{k,k'}$ are*

$$\mathbb{E}_{q,\mathrm{c}}[\neg\hat{c}_k^{k,k'}] = N_{k,k'} - p_k r_{k,k'}$$

$$\mathbb{E}_{q,\mathrm{c}}[\neg\hat{c}_{k'}^{k,k'}] = N_{k,k'} - p_k r_{k,k'}$$

where $r_{k,k'} = \mathbb{E}_q\left[\sum_{d\in\Omega(q)}\mathbb{1}_{[(q,d)\in S_{k,k'}]}\mathrm{rel}(q,d)\right]$ and $N_{k,k'} = \mathbb{E}_q[\sum_{d\in\Omega(q)}\mathbb{1}_{[(q,d)\in S_{k,k'}]}]$.

Proof. The proof is analogous to Proposition 1 and thus omitted. □

The way we constructed the interventional sets, the expected relevances $r_{k,k'}$ are not necessarily normalized to be within the interval $[0,1]$. It is therefore more convenient to instead consider the normalized version of $r_{k,k'}$

$$\bar{r}_{k,k'} \equiv \frac{r_{k,k'}}{N_{k,k'}} \in [0,1],$$

using the definitions from Propositions 1 and 2. Under this normalization, we have that $\mathbb{E}[\hat{c}_k^{k,k'}] = p_k\bar{r}_{k,k'}N_{k,k'}$ and $\mathbb{E}[\hat{\neg c}_k^{k,k'}] = (1 - p_k\bar{r}_{k,k'})N_{k,k'}$ and similarly for $\hat{c}_k'^{k,k'}$ and $\hat{\neg c}_k^{k,k'}$. We model this normalized $\bar{r}_{k,k'}$ in the AllPairs estimator.

We can now formulate the training objective of the AllPairs estimator. The following objective needs to be maximized with respect to $\hat{p}_k \in [0, 1]$ and $\hat{r}_{k,k'} = \hat{r}_{k',k} \in [0, 1]$ (since $\bar{r}_{k,k'} = \bar{r}_{k',k}$ by definition)

$$(\hat{p}, \hat{r}) =_{p,r} \sum_{k \neq k' \in [M]} \hat{c}_k^{k,k'} \log(\hat{p}_k\hat{r}_{k,k'}) + \hat{\neg c}_k^{k,k'} \log(1 - \hat{p}_k\hat{r}_{k,k'}).$$

This optimization problem can be interpreted as Weighted Cross-Entropy Maximization for estimating the distribution $p_k\bar{r}_{k,k'}$ using weighted samples $\hat{c}_k^{k,k'}$ and $\hat{\neg c}_k^{k,k'}$. The weighting by $N_{k,k'}$ ensures that the contribution of each aggregated click-through sample is proportional to the size of its interventional set.

From the solution of this optimization problem, $(\hat{p}, \hat{r})$, we only need $\hat{p}$ while the matrix $\hat{r}$ can be discarded. To get normalized propensities relative to rank 1, we compute $\hat{p}_k/\hat{p}_1$. Note that the optimization problem is quite small, as it uses only $O(M^2)$ variables and $2 * M * (M - 1)$ terms in the objective.

Unlike the local estimators, the AllPairs approach integrates all data from every interventional set. We will evaluate empirically how much statistical efficiency is gained by taking this global approach compared to the local estimators.

4.3 Empirical Evaluation

We take a two-pronged approach to evaluating our intervention-harvesting technique and the estimators. First, we fielded them on two real-world systems – the Arxiv Full-Text Search Engine and the Google Drive Search – to gain insight into their practical effectiveness. Second, we augment these real-world experiments with simulation studies using synthetically generated click data, where we can explore the behavior of our method over the whole spectrum of settings (e.g. varying ranker similarity, presentation-bias severity, and click noise).

4.3.1 Real-World Evaluation: Arxiv Full-Text Search

We conducted a controlled experiment on the Arxiv Full-Text Search Engine, where we compare the results from the AllPairs estimator using Intervention Harvesting with the results from a gold-standard intervention experiment as described below. We used three ranking functions $\{f_1, f_2, f_3\}$ for defining the interventional sets, which were generated by using different learning methods and datasets.

Gold Standard. Since we do not know the true propensities for the Arxiv Full-Text Search Engine, we use the *Swap*$(1, k)$ interventions and estimator described in Section 4.1.2 as our gold standard. With probability 0.5, an incoming query is assigned to generating data towards a swap experiment. For each assigned query, a ranking function f_i is chosen uniformly at random, and its rank 1 is swapped uniformly with rank $k \in \{1, .., 21\}$ before it is presented to the user. The $\hat{c}_1^{1,k}$ and $\hat{c}_k^{1,k}$ are computed over all three f_i.

Intervention Harvesting. For the other half of the incoming queries, we uniformly pick a ranking function f_i and present its results without further intervention. From the data in these conditions, we then compute the interventional sets $S_{k,k'}$ for $k \neq k' \in \{1, .., 21\}$ and their resulting $\hat{c}_k^{k,k'}$, $\hat{c}_{k'}^{k,k'}$, $\neg\hat{c}_k^{k,k'}$ and $\neg\hat{c}_{k'}^{k,k'}$. These are then used in the AllPairs estimator from Section 4.2.2.

Results.

Data for all six conditions was collected simultaneously between May 14, 2018 and August 1, 2018 to avoid confounding due to shift in the query distribution for maximum validity of the experiment. About 53,000 queries and 25,600 clicks were collected, about half for the Swap Experiment and half for the Interventional Set method.

The estimated propensity curves for the gold-standard swap experiment and for the AllPairs estimator are shown in Figure 4.1. The shaded region for each curve corresponds to a 95% confidence interval calculated from 1000 bootstrap samples. As we can see, the curves follow a similar trend for each position and AllPairs mostly lies within the confidence interval of the gold-standard curve. However, the confidence interval for the AllPairs method is substantially tighter than for the swap experiment, indicating that AllPairs is not only less intrusive (no swap interventions) but also statistically more efficient given the same number of queries.

The following provide further insight into this gain in efficiency. While we conducted interleaving experiments [13] to confirm that the three ranking functions $\{f_1, f_2, f_3\}$ provide similar ranking accuracy, we found that the three rankers tend to assign documents to different ranks. In fact, only 7.39% of the documents were ranked at the same rank by all three rankers, such that about 93% of

Table 4.1: Size of the interventional sets $S_{k,k'}$ for Arxiv (showing only top 10 positions).

rank k									rank
1	2	3	4	5	6	7	8	9	k'
13625	8340	6723	5231	3966	3051	2656	2274	2015	2
-	9039	7692	5994	5053	3588	2675	2861	2244	3
-	-	8555	6994	5573	4117	3368	3321	2361	4
-	-	-	6783	5345	5040	3849	3427	3614	5
-	-	-	-	6290	4809	4058	4126	3489	6
-	-	-	-	-	5466	4746	3935	3294	7
-	-	-	-	-	-	5425	4092	3692	8
-	-	-	-	-	-	-	4258	3930	9
-	-	-	-	-	-	-	-	3719	10

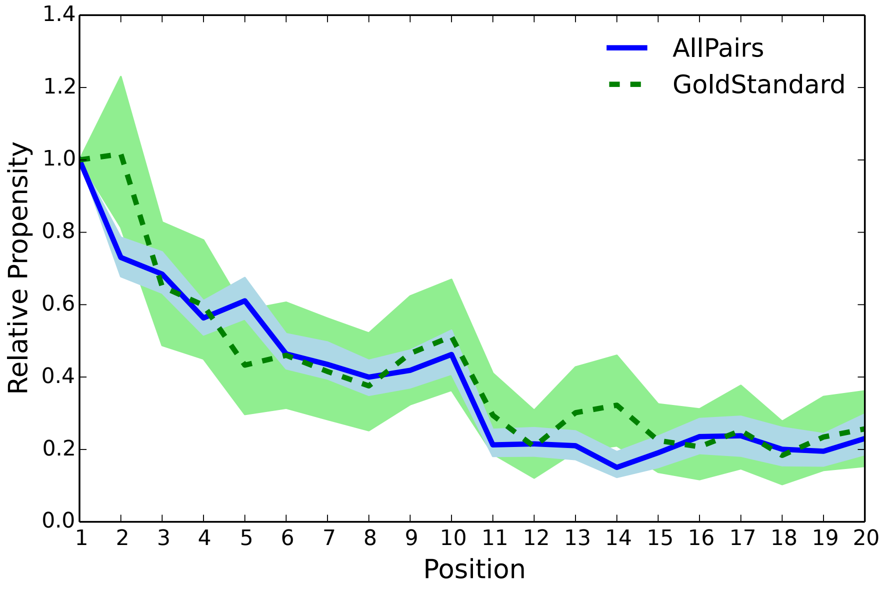


Figure 4.1: Estimated propensity curves for Arxiv. Shaded regions correspond to a 95% confidence interval.

the documents contributed to an interventional set and thus became meaningful training data for AllPairs. Table 4.1 shows the size of the interventional sets $S_{k,k'}$ for the top 10 positions. While local swaps are most common, the rankers frequently rank the same document at substantially different ranks.

4.3.2 Real-World Evaluation: Google Drive Search

We conducted a second real-world experiment on the search for Google Drive. The service uses an overlay to show results as users type. The overlay disappears when a click on the overlay happens. Thus, each query has at most a single click. The overlay displays at most 5 results for each query and all the displayed results are logged with their position information. Again, we compare the results from the AllPairs estimator against a gold-standard intervention experiment.

Gold Standard. We follow [55] and use $Swap(k, k+1)$ interventions along with the AdjacentChain estimator as our gold standard.

Intervention Harvesting. In an A/B test, users were randomly assigned either to the production ranker f_1 or a new ranker f_2. Based on these two rankers the interventional sets $S_{k,k'}$ with $k \neq k' \in \{1, .., 5\}$ were computed and provided to the AllPairs estimator.

Results. During two weeks in April 2018, a total of 877,689 queries were collected. Note that data for the gold-standard swap experiment was collected at a different time and for a different ranking function, but is is reasonable to assume that the propensity curve has not changed. The estimated propensity curves for the AllPairs estimator and the gold-standard swap experiment are shown in Figure 4.2. The AllPairs estimates closely resemble the gold standard as desired. For comparison, we also computed propensity estimates via the regression EM from [55] on the same data set as AllPairs. Those estimates are

Table 4.2: Size of the interventional sets $S_{k,k'}$ for Google Drive.

rank k				rank
1	2	3	4	k'
19,516	2,203	579	244	2
-	48,576	10,630	3,780	3
-	-	73,170	21,037	4
-	-	-	92,343	5

substantially off from the gold standard. Even further off is the naive method of using the empirical clickthrough rate (CTR) as an estimate for p_k without any experimental control for relevance. Overall, we find that Intervention Harvesting with AllPairs provides superior results that are close to the gold standard.

In some ways, the Arxiv experiment and the Google Drive experiment covered two substantially different use cases. For Arxiv, the rankings were substantially different, while for Google Drive the two ranking functions were typically quite close. In fact, the two rankers for Google Drive provided identical top-5 rankings for 75.26% of the queries, which thus did not contribute any useful data to the interventional sets. And even among the queries that lead to different rankings, 74.5% of the results were at the same rank in both ranking functions. This means much less interventional-set data was generated per query, and Table 4.2 further shows that most of the swaps were quite local. It is therefore reassuring that AllPairs nevertheless provides accurate estimates, and that it works well for both Arxiv and Google Drive despite these differences.

4.3.3 Robustness Analysis: Yahoo LTR Challenge

While the two real-world experiments provide validation for the applicability of the method, these are just two data-points in a large spectrum of possible

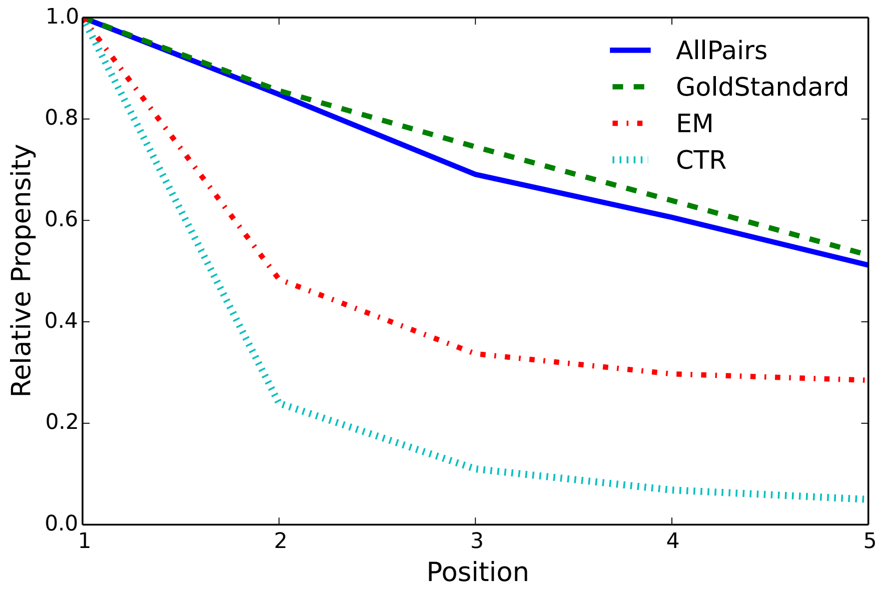


Figure 4.2: Estimated propensity curves for Google Drive.

settings. We therefore now evaluate the robustness of the method on synthetic click data, where we know the true propensity curve by construction and can control the properties of the data with respect to all relevant parameters (e.g. noise, ranker similarity, data-set size).

Experiment Setup. We generated synthetic click data according to the following methodology that closely matches that in [33]. In particular, we use the Yahoo LTR Challenge data, which comes with manual relevance judgments. Using these relevance judgments, clicks were generated by simulating the Position-Based Model with propensities that decay with the presented rank via $p_r = (\frac{1}{r})^\eta$. The parameter η controls the severity of bias, with higher values causing greater position bias. We also introduced noise into the clicks by allowing some irrelevant documents to be clicked. Specifically, an irrelevant document ranked at position r by the production ranker is clicked with probability p_r times ϵ_- whereas a relevant document is clicked with probability p_r. For simplicity (and with-

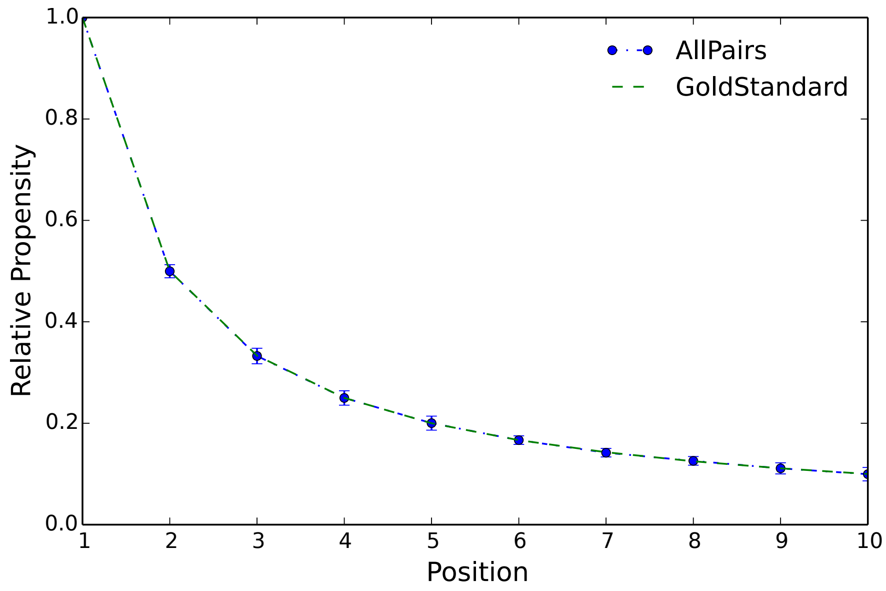
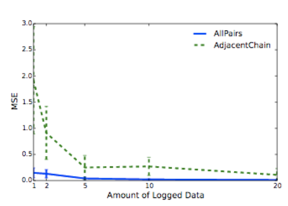
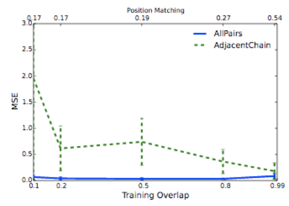


Figure 4.3: Estimated propensity curve for synthetic click data derived from the Yahoo LTR dataset ($\eta = 1$, $\epsilon_{-} = 0.1$, $n_i = 99720$, frac = 0.02, overlap = 0.8).

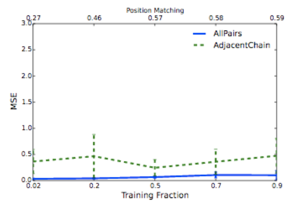
out loss of generality), we used click logs from two rankers in each experiment setting. Rankers were obtained by training Ranking SVMs on random samples of queries with their manual relevance judgments. The "similarity" of the two rankers was controlled by varying the degree of overlap in their respective training sets. We evaluate the estimation accuracy via the Mean Squared Error (MSE) of the estimated inverse relative propensity weights, $MSE = \frac{\sum_{i=1}^{M}(\hat{p}_1/\hat{p}_i - p_1/p_i)^2}{M}$, since these inverse propensity weights better reflect how inaccurate propensity estimates impact the IPS estimator [33]. We estimate propensities up to rank $M = 10$. Error bars indicate the standard deviation over 6 independent runs (except in Figure 4.3 as described below). If not mentioned otherwise, the number of simulated queries per ranker is $n_i = 99720$ (obtained by 5 sweeps of the 19944 queries in the Yahoo LTR training set), and we use $\eta = 1$ and $\epsilon_{-} = 0.1$.



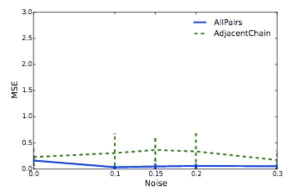
(a) Estimation error with increasing amount of log data for each ranker where every 1 **sweep has 19944 queries.** ($\eta = 1$, $\epsilon_- = 0.1$, frac $= 0.02$, overlap $= 0.8$)



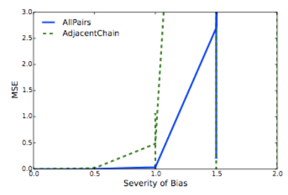
(b) Estimation error with increasing fraction of overlap in the training data for the rankers. ($\eta = 1$, $\epsilon_- = 0.1$, $n_i = 99720$, frac $= 0.02$)



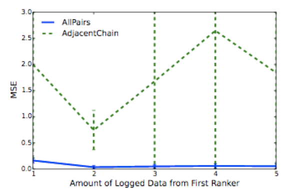
(c) Estimation error with increasing fraction of the training data used for both rankers. ($\eta = 1$, $\epsilon_- = 0.1$, $n_i = 99720$, overlap $= 0.8$)



(d) Estimation error with increasing amount of noise ϵ_- in the training data. ($\eta = 1$, $n_i = 99720$, frac $= 0.5$, overlap $= 0.5$)



(e) Estimation error with increasing severity of bias η in the training data. ($\epsilon_- = 0.1$, $n_i = 99720$, frac $= 0.5$, overlap $= 0.5$)



(f) Estimation error with increasingly unbalanced amounts of log data from f_1 vs. f_2 where every 1 **sweep has 19944 queries.** ($\eta = 1$, $\epsilon_- = 0.1$, $n_1 + n_2 = 119664$, frac $= 0.02$, overlap $= 0.5$)

Figure 4.4: Robustness experiments on the synthetic click data derived from the Yahoo LTR dataset.

Does AllPairs recover the true propensity curve?

Figure 4.3 shows the estimated propensities of AllPairs in comparison to the true propensity curve of ($\eta = 1$) that is known by construction. All parameters of the simulation experiment are kept at the defaults as stated above. As expected, AllPairs perfectly recovers the true propensities. The error bars show the 99% confidence intervals that are computed via the standard deviation estimated from re-running the simulation experiment 20 times.

How much data is needed?

While the previous section showed that Intervention Harvesting with the AllPairs estimator converges to the true propensities, we now analyze the speed of convergence. Figure 4.4 (a) shows that AllPairs provides good estimates even for modest amounts of data. As a baseline for comparison, we also show the MSE of the AdjacentChain estimator, which is substantially worse and requires at least one order of magnitude more data to achieve the same MSE as AllPairs. We also explored the use of the PivotOne estimator, but do not report its results since they are typically worse than those of AdjacentChain.

How different should the ranking functions be?

If the ranking functions f_i are all identical, then Intervention Harvesting cannot produce any data. So, how different do the rankers f_i need to be? To vary ranker similarity, we trained pairs of rankers with increasing overlap (from 1% to 99%) in their training sets. In Figure 4.4 (b), we see that the estimation accuracy remains quite robust even as the rankers become increasingly similar due to the overlap in the data they are trained on. The top of the plot shows the similarity in terms of the fraction of documents at the same rank in both rankers averaged across all the queries. As expected, the error goes up when the rankers are very similar since then they tend to put documents at the same position, leading to fewer interventional pairs. Interestingly, the error is also relatively higher when the rankers are too dissimilar. This is because when the candidate sets are larger than 10, the dissimilarity in the rankers causes many interventions to be discarded since they often go beyond rank 10. Note that AdjacentChain benefits from ranker similarity, since it focuses the interventional set data on $S_{k,k+1}$.

However, AdjacentChain at best matches the performance of AllPairs, but never outperforms it.

How important is the quality of the rankers?

Another way of controlling ranker similarity is to increase the total number of training examples for both. Figure 4.4 (c) shows the result of this experiment. Again, AllPairs shows robust performance over the whole spectrum of settings and substantially improves over AdjacentChain.

How does click noise impact estimation accuracy?

Clicks in any real-world setting will be noisy, and we thus want to explore the robustness of AllPairs with respect to noise. Note that noise (which can be seen simply as an alternative vector of relevances $rel(q, d)$ in our model) should have no influence on the propensity estimates via Information Harvesting, as long as the noise is not confounded by rank (which is guaranteed in the PBM). Figure 4.4 (d) verifies that Information Harvesting estimates are indeed stable over different levels of ϵ_- noise.

How does bias severity impact estimation accuracy?

Figure 4.4 (e) explores the behavior of the estimators when we vary the steepness of the propensity curve via η. The AllPairs method performs better than AdjacentChain, but the MSE of both methods increases sharply when the propensity curve gets steep. The explanation for this is twofold. First, the increase is partly an artifact of the MSE error measure on the inverse propensities. Steep curves lead to small propensities, which in turns generates large inverse propensities that provide opportunity for large MSE. Second, for steep propensity curves the bottom ranks receive only few clicks such that there is not enough data for estimating their propensities reliably.

How robust is the method to imbalanced datasets?

Finally, we explore the situation where we may have more log data from one ranker than from the other. Figure 4.4 (f) shows MSE when we vary the amount of log data for f_1 while keeping the total amount of data constant at $119,664$ queries (i.e. 6 sweeps). The plot shows that AllPairs is robust to such imbalances.

4.4 Conclusion

We presented the idea of Intervention Harvesting, allowing the use of multiple historic loggers for generating interventional data under mild assumptions. We showed how this idea can be used to control for relevance, providing the first method for propensity estimation in the PBM that does not require intrusive interventions, a relevance model, or repeat queries. In particular, we propose the AllPairs estimator for combining all intervention data, which we find to provide superior propensity estimation accuracy compared to existing local estimators over a wide spectrum of settings.

CHAPTER 5

ADDRESSING TRUST BIAS FOR UNBIASED LEARNING-TO-RANK

So far, we have focused on the Position-Based Model (PBM) which models the differential propensity of examining documents at different results, but ignores the possibility of position-dependent noise in relevance evaluation upon examination. To address this limitation, we introduce a noise-aware PBM, named TrustPBM, to model trust bias in user clicks in [3], and show how to modify the unbiased LTR framework to address both examination and trust biases simultaneously.

5.1 Trust Enhanced Position-Based Model

The Position-Based Model (PBM) is a simple generative model which has been extensively used in unbiased LTR. Most existing click models including the PBM adhere to the *examination hypothesis*. Suppose we have N positions, and for a query q, document d is displayed at position $k \in [1, N]$. A click can be generated only after the user examines the result. In the PBM model, the user clicks the document if and only if they examine the document and the document is relevant. The examination only depends on the position k, but not on q or d. Such a model factors out q and d, and makes the estimation of the probability of examination simpler and more robust, compared to other click models.

Formally, let C be a binary random variable signifying whether the user clicks document d, E whether the user examines the document, R for the true relevance and $\tilde{R}$ to represent the perceived or judged relevance. The PBM makes the following assumptions

$$C = 1 \Leftrightarrow E = 1, \tilde{R} = 1$$

$$R \Leftrightarrow \tilde{R}$$

$$\Pr(E = 1|q, d, k) = \Pr(E = 1|k)$$

$$\Pr(R|q, d, k) = \Pr(R|q, d)$$

Thus, the probability of examining a document only depends on the position k and the perceived relevance is the same as the true relevance. Based on these assumptions, the PBM has the following probability for a click.

$$\begin{aligned}\Pr(C = 1|q, d, k) &= \Pr(E = 1, \tilde{R} = 1|q, d, k)\\ &= \Pr(E = 1, R = 1|q, d, k)\\ &= \Pr(E = 1|k) \Pr(R = 1|q, d)\\ &= \theta_k \gamma_{q,d}\end{aligned}$$

where we use θ_k and $\gamma_{q,d}$ as short-hands for the probabilities.

5.1.1 Trust Bias

The PBM model and its extensions [54] have a noise-free assumption that perceived relevance is the same as the true relevance. This is not the case in general. How to model the relation between perceived relevance and true relevance is relatively less studied, even in the click models. DBN [15] is one of the few models that incorporates this relation. In DBN, a set of document-dependent parameters α_d are introduced to model the relation between perceived relevance and true relevance:

$$\Pr(R = 1|q, d) = \alpha_d \Pr(\tilde{R} = 1|q, d)$$

where a parameter α_d depend on the document d. The intuition is that the probability of true relevance is a discounted version of perceived relevance and that the discount is document-dependent. As a result, the model has a large number of additional parameters that equals to the number of unique documents in the data.

In this chapter, we model the relation between perceived relevance and true relevance based on trust bias introduced in [30]. In this user study, eye tracking was used to monitor examination. The study showed that, after examination, users tend to trust the results presented higher on the result page more, demonstrated by higher click ratio despite the fact that the documents are both examined and their expected relevance is the same. This suggests a position-dependent trust bias in addition to examination bias, due to which higher ranked non-relevant documents may be clicked and lower ranked relevant documents not clicked. However, this has not been mathematically formulated in the past. We model the trust bias as follows,

$$\Pr(\tilde{R} = 1 | R = 1, E = 1, k) = \epsilon_k^+$$
$$\Pr(\tilde{R} = 1 | R = 0, E = 1, k) = \epsilon_k^-$$

Since $C = 1 \Leftrightarrow E = 1, \tilde{R} = 1$, we have $\Pr(C = 1 | R = 1, E = 1, k) = \epsilon_k^+$ and $\Pr(C = 1 | R = 0, E = 1, k) = \epsilon_k^-$. Intuitively, after examination, a relevant document at position k can be missed with probability $1 - \epsilon_k^+$, but a non-relevant document can be clicked mistakenly with probability ϵ_k^-. Notice that the noise is position-dependent. A similar formulation has been studied in [33] where the noise is constant across positions.

5.1.2 TrustPBM

We enhance the PBM model by explicitly modeling the trust bias introduced above. We have

$$\begin{aligned}
&\Pr(\tilde{R} = 1 | E = 1, q, d, k) \\
=&\Pr(\tilde{R} = 1 | R = 1, E = 1, k) \Pr(R = 1 | q, d) \\
+&\Pr(\tilde{R} = 1 | R = 0, E = 1, k) \Pr(R = 0 | q, d) \\
=&\epsilon_k^+ \gamma_{q,d} + \epsilon_k^- (1 - \gamma_{q,d})
\end{aligned}$$

Thus, in TrustPBM, the probability of a click is given as

$$\begin{aligned}
\Pr(C = 1 | q, d, k) &= \Pr(E = 1 | k) \Pr(\tilde{R} = 1 | E = 1, q, d, k) \\
&= \theta_k (\epsilon_k^+ \gamma_{q,d} + \epsilon_k^- (1 - \gamma_{q,d}))
\end{aligned}$$

PBM can be viewed as a special case of TrustPBM, where $\epsilon_k^+ = 1$ and $\epsilon_k^- = 0$ to reflect the noise-free assumption. Both PBM and TrustPBM model examination bias in the same way.

On the other hand, TrustPBM is different from DBN. The noise depends on position only, not on the document. Compared to PBM, we introduce $2N$ more parameters into the model where N is usually small (e.g., 10), so the model complexity of TrustPBM is much smaller than DBN.

The TrustPBM parameters can be estimated using logged click data. Let $\mathcal{L}$ be the logged data consisting of tuples (q, d, k, c) for each received user query q, each document d displayed at any position k and a binary indicator c for whether the document was clicked or not (i.e. the realized value of the random variable C). Then, the log-likelihood of generating $\mathcal{L}$ (over the randomness in user click behavior) is,

$$\log \Pr(\mathcal{L}) = \sum_{(q,d,k,c)\in\mathcal{L}} c \log \theta_k(\epsilon_k^+ \gamma_{q,d} + \epsilon_k^-(1-\gamma_{q,d})) \\ + (1-c)\log(1-\theta_k(\epsilon_k^+\gamma_{q,d} + \epsilon_k^-(1-\gamma_{q,d})).$$

In order to find the parameters that maximize the above log-likelihood objective, we use Expectation-Maximization, extending the Regression-based EM method for the plain PBM in [55].

5.2 Estimation via Expectation-Maximization

As we will show in Section 5.3, we only need the parameters θ_k, ϵ_k^+, and ϵ_k^- for unbiased LTR. However, unlike the standard PBM model whose parameters can be estimated based on result randomization, TrustPBM does not allow estimation of these parameters using a randomized data set. We extend the previously proposed regression-based EM algorithm [55] to estimate the parameters.

In the standard EM procedure, parameters are estimated by alternating Expectation and Maximization steps until convergence. Suppose $\{\theta_k^{(t)}, \epsilon_k^{+(t)}, \epsilon_k^{-(t)} : k \in [N]\}$ and $\{\gamma_{q,d}^{(t)} : q, d \in \mathcal{L}\}$ are the parameter estimates at iteration t (initialized appropriately at the first iteration). We will get a new set of parameter estimates after iteration $t+1$.

5.2.1 Expectation Step

In the Expectation step at iteration $t+1$, the posterior distribution of the hidden variables of examination E and true relevance R given the observed click data $\mathcal{L}$

$$\begin{aligned}
Pr(E=1,R=1|q,d,k,c=1) &= \frac{\epsilon_k^+\gamma_{q,d}}{\epsilon_k^+\gamma_{q,d}+\epsilon_k^-(1-\gamma_{q,d})}\\
\Pr(E=0,R=1|q,d,k,c=1) &= 0\\
Pr(E=1,R=0|q,d,k,c=1) &= \frac{\epsilon_k^-(1-\gamma_{q,d})}{\epsilon_k^+\gamma_{q,d}+\epsilon_k^-(1-\gamma_{q,d})}\\
\Pr(E=0,R=0|q,d,k,c=1) &= 0\\
Pr(E=1,R=1|q,d,k,c=0) &= \frac{\theta_k(1-\epsilon_k^+)\gamma_{q,d}}{1-\theta_k(\epsilon_k^+\gamma_{q,d}+\epsilon_k^-(1-\gamma_{q,d}))}\\
\Pr(E=0,R=1|q,d,k,c=0) &= \frac{(1-\theta_k)\gamma_{q,d}}{1-\theta_k(\epsilon_k^+\gamma_{q,d}+\epsilon_k^-(1-\gamma_{q,d}))}\\
\Pr(E=1,R=0|q,d,k,c=0) &= \frac{\theta_k(1-\epsilon_k^-)(1-\gamma_{q,d})}{1-\theta_k(\epsilon_k^+\gamma_{q,d}+\epsilon_k^-(1-\gamma_{q,d}))}\\
\Pr(E=0,R=0|q,d,k,c=0) &= \frac{(1-\theta_k)(1-\gamma_{q,d})}{1-\theta_k(\epsilon_k^+\gamma_{q,d}+\epsilon_k^-(1-\gamma_{q,d}))}
\end{aligned}$$

Figure 5.1: Formulas for computing probabilities of hidden variables in the E-step of the EM algorithm for TrustPBM.

is expressed in terms of the parameter estimates at iteration t, as shown in Figure 5.1 (we omit superscript (t) for readability). All the formulas follow directly from Bayes rules. For instance, in the third from last equation, we have

$$\begin{aligned}
&\Pr(E=0,R=1|q,d,k,c=0)\\
&=\frac{\Pr(C=0|E=0,R=1,q,d,k)\Pr(E=0,R=1|q,d,k)}{\Pr(C=0|q,d,k)}\\
&=\frac{\Pr(E=0,R=1|q,d,k)}{\Pr(C=0|q,d,k)}\\
&=\frac{\Pr(E=0|k)\Pr(R=1|q,d)}{\Pr(C=0|q,d,k)}
\end{aligned}$$

since $\Pr(C=0|E=0,R=1,q,d,k)=1$, and examination and relevance are independent events. Also, a click means that the document must have been examined and thus $\Pr(E=0,R=1|q,d,k,c=1)=0$ in the second equation.

$$\theta_k = \frac{\sum_{(q,d,k',c)\in\mathcal{L}} 1\{k' = k\}(c + (1-c)P(E=1|q,d,k,C=0))}{\sum_{(q,d,k',c)\in\mathcal{L}} 1\{k' = k\}}$$

$$\gamma_{q,d} = \frac{\sum_{(q',d',k,c)\in\mathcal{L}} 1\{q' = q, d' = d\}(c \cdot P(R=1|q,d,k,C=1) + (1-c)P(R=1|q,d,k,C=0))}{\sum_{(q',d',k,c)\in\mathcal{L}} 1\{q' = q, d' = d\}}$$

$$\epsilon_k^+ = \frac{\sum_{(q,d,k',c)\in\mathcal{L}} 1\{k' = k\} \cdot c \cdot P(E=1,R=1|,q,d,k,C=1)}{\sum_{(q,d,k',c)\in\mathcal{L}} 1\{k' = k\}(c \cdot P(E=1,R=1|q,d,k,C=1) + (1-c)P(E=1,R=1|q,d,k,C=0))}$$

$$\epsilon_k^- = \frac{\sum_{(q,d,k',c)\in\mathcal{L}} 1\{k' = k\} \cdot c \cdot P(E=1,R=0|,q,d,k,C=1)}{\sum_{(q,d,k',c)\in\mathcal{L}} 1\{k' = k\}(c \cdot P(E=1,R=0|q,d,k,C=1) + (1-c)P(E=1,R=0|q,d,k,C=0))}$$

Figure 5.2: Formulas for updating parameters in the M-step of the EM algorithm for TrustPBM, where 1 is the indicator function.

5.2.2 Maximization Step

In the Maximization step at iteration $t+1$, the parameters are updated to their maximum likelihood values given the click data and the posterior probabilities from the Expectation step as shown in Figure 5.2. Once again, we drop the superscript $(t+1)$ for clarity in these formulas. Different from PBM model, we have additional ϵ_k^+ and ϵ_k^- parameters updated in this step.

Regression-based EM. For the M-step to work with query-document pairs (q,d), a requirement is that (q,d) should repeat and also be shown in different positions. Since the query-document pairs (q,d) vary arbitrarily, using free parameters $\gamma_{q,d}$ makes the objective very sparse and difficult to learn. Moreover, exposing individual (q,d) identifiers can also pose a privacy or security risk. Therefore, as in [55], we propose to estimate the $\gamma_{q,d}$ parameters in each Maximization step via regression, i.e. fit a function (such as a Gradient Boosted Decision Tree) from features of (q,d) to the derived target value for $P(R=1|c,q,d,k)$

at iteration $t + 1$.

More specifically, we assume that there is a feature vector $\mathbf{x}_{q,d}$ representing them and use a function to compute the relevance $\gamma_{q,d} = f(\mathbf{x}_{q,d})$. The Maximization step is then to find a regression function $f(\mathbf{x})$ to maximize the likelihood given the estimation from the Expectation step. Specifically, for each $(q, d, k, c) \in \mathcal{L}$, the expectation step gives a probability $P(R = 1|q, d, k, c)$. Intuitively, we can regress the feature vector $\mathbf{x}_{q,d}$ to the probability $P(R = 1|c, q, d, k)$. Similar to [55], we convert such a regression problem to a classification problem based on sampling: we sample a binary relevance label $r \in \{0, 1\}$ according to $P(R = 1|c, q, d, k)$. This conversion allows us to use widely available classification tools to solve our problem. After sampling, we have a training set $\{(\mathbf{x}, r)\}$ for $f(\mathbf{x})$. The objective function is the log likelihood:

$$\sum_{\{(\mathbf{x},r)\}} r \log(f(\mathbf{x})) + (1 - r) \log(1 - f(\mathbf{x})),$$

where we use the sigmoid in the objective function

$$f(\mathbf{x}) = \frac{1}{1 + e^{-F(\mathbf{x})}}$$

$F(x)$ is the log odd of function $f(x)$ and is learned by Gradient Boosted Decision Tree (GBDT) method [21].

5.3 Unbiased LTR via Bayes-IPS Correction

Now we turn to the question of using the parameter estimates of the TrustPBM to learn a new ranking function from the click data in an unbiased way. Specifically, we need to adjust each click signal in the logged data such that it reflects the actual underlying relevance signal in expectation, unconfounded by the inherent examination and trust biases captured by TrustPBM.

In existing work [33, 55] which deals with the standard PBM, this is achieved via Inverse Propensity Scoring (IPS) which corrects for the examination bias factor. Since TrustPBM does not factorize into purely bias (from examination and trust) and relevance terms, a direct IPS correction is not feasible. In the following, we first review the existing unbiased LTR and then present our two-step approach to deal with noise: correct for examination bias via IPS, and then de-noise for trust bias via Bayes law.

5.3.1 LTR

We begin with a brief overview of the LTR setting. In the standard LTR, a set of labeled data for query and documents are given $\mathcal{L}_U = \{(q, d, r)\}$ where r represents the relevance labels for the (q, d) pair. We also use $\Omega_q = \{d | (q, d) \in \mathcal{L}_U\}$ to represent all documents that are associated with query q. Collection $\mathcal{L}_U$ is *unbiased* and a uniform sample from the space of queries, documents, and relevance. In general, fully supervised LTR algorithms such as LambdaMART and RankingSVM are used to directly or indirectly optimize a performance metric which decomposes as a sum over the (q, d) samples in the training set, as such

$$M = \sum_{(q,d,r) \in \mathcal{L}_U} r \cdot f(q, d, \Omega_q)$$

where f is usually a rank-related function.

For instance, Average Relevance Position (ARP) [56, 33, 55] and Discounted

Cumulative Gain (DCG) like metrics [27, 55] are shown as below,

$$ARP = \sum_{(q,d,r)\in\mathcal{L}_U} r\,\mathrm{rank}(q,d,\Omega_q)$$
$$DCG = \sum_{(q,d,r)\in\mathcal{L}_U} r\,\frac{1}{\mathrm{rank}(q,d,\Omega_q)}.$$

Note that lower *ACP* and higher *DCG* indicate better ranking performance with more relevant documents at higher ranks. We use a simple DCG definition that uses a binary relevance and a linear function for position discount to be consistent with the rest of the chapter. But the methodology is applicable to any DCG definitions.

In pairwise learning-to-rank algorithms, scores s are learned for (q,d) pairs by expressing rank as a pairwise objective

$$\begin{aligned} ARP &= \sum_{(q,d,r)\in\mathcal{L}_U} r\,\mathrm{rank}(q,d,\Omega_q) \\ &= \sum_{(q,d,r)\in\mathcal{L}_U} r\,(\sum_{d'\in\Omega_q} \mathbb{I}_{s_{(q,d)}<s_{(q,d')}} + 1) \\ &= \sum_{(q,d,r)\in\mathcal{L}_U} \sum_{d'\in\Omega_q} r\,\mathbb{I}_{s_{(q,d)}<s_{(q,d')}} + const \end{aligned} \tag{5.1}$$

The loss above can then be upper-bounded by common classification loss functions (e.g., the hinge loss or the logistic loss [24]) as in the RankingSVM method. Moreover, rank-based metrics such as *ARP* and *DCG* can also be optimized in the LambdaLoss framework [56] where LambdaRank and LambdaMART [9] are special cases in the framework.

5.3.2 Unbiased LTR

The challenge in learning from implicit feedback such as click data is that only the click information c is observed, not the true relevance r for each (q, d) pair. The relation between r and c is captured by click models. The Inverse Propensity Scoring (IPS) technique is used to tackle this issue by de-biasing click data because the examination bias in PBM is equivalent to the propensity of observing relevance labels in the counterfactual framework [33].

Recall that we have logged data $\mathcal{L}$ consisting of tuples (q, d, k, c),

$$\mathbb{E}[C|q, d, k] = \Pr(C = 1|q, d, k) = \theta_k \Pr(\tilde{R} = 1|E = 1, q, d, k)$$

where the expectation is over the randomness in the user click behavior. As in previous work, we can employ IPS to correct for the examination bias in the following way,

$$\begin{aligned} M_{IPS} &= \sum_{(q,d,k,c)\in\mathcal{L}} \frac{c}{\theta_k} f(q, d, \Omega_q) \\ &= \sum_{(q,d,k,c=1)\in\mathcal{L}} \frac{1}{\theta_k} f(q, d, \Omega_q) \end{aligned}$$

M_{IPS} is an unbiased estimation of metric M because

$$\begin{aligned} \mathbb{E}[M_{IPS}] &= \sum_{(q,d,k,c)\in\mathcal{L}} \frac{\mathbb{E}[C]}{\theta_k} f(q, d, \Omega_q) \\ &= \sum_{(q,d,k,c)\in\mathcal{L}} \Pr(\tilde{R} = 1|E = 1, q, d, k) f(q, d, \Omega_q) \end{aligned}$$

In the plain PBM which assumes that $\Pr(\tilde{R} = 1|E = 1, q, d, k) = \Pr(R = 1|q, d)$, i.e. judgment upon examination reflects actual relevance free from noise, interpreting r as $\Pr(R = 1|q, d)$ (the degree of true relevance), M_{IPS} equals M in expectation and thus can be used as a direct unbiased substitute in the learning-to-rank algorithm since it is simply a linear weight modification.

$$\begin{aligned}
ARP_{IPS} &= \sum_{(q,d,k,c=1)\in\mathcal{L}} \frac{1}{\theta_k}\,\mathrm{rank}(q,d,\Omega_q) \\
DCG_{IPS} &= \sum_{(q,d,k,c=1)\in\mathcal{L}} \frac{1}{\theta_k}\,\frac{1}{\mathrm{rank}(q,d,\Omega_q)}.
\end{aligned}$$

For instance, the upper-bounding technique shown in Eq 5.1 still applies with ARP_{IPS} and an IPS-based RankingSVM was derived in [33] and the LambdaMART algorithm can be used to optimize DCG_{IPS}.

5.3.3 Bayes-IPS Correction

Now, for the TrustPBM, we do not have the equivalence between perceived relevance $\tilde{R}$ and R. The standard IPS approach gives an unbiased estimation of $\tilde{R}$. Note that we only care about the clicked documents in M_{IPS}. To get the estimate of R, we only need to estimate it for $C = 1$, i.e., $\Pr(R = 1|C = 1, q, d, k)$. Such an estimation can involve many parameters due to the dependency on (q, d) pairs. To make it tractable, we thus approximate it by the average relevance at position k: $\Pr(R = 1|C = 1, k)$, i.e., $\Pr(R = 1|\tilde{R} = 1, E = 1, k)$.

By Bayes law, we have

$$\begin{aligned}
&\Pr(R = 1|\tilde{R} = 1, E = 1, k) \\
&= \frac{\Pr(\tilde{R} = 1|R = 1, E = 1, k)\Pr(R = 1|E = 1, k)}{\Pr(\tilde{R} = 1|E = 1, k)} \\
&= \frac{\epsilon_k^+ \Pr(R = 1|E = 1, k)}{\epsilon_k^+ \Pr(R = 1|E = 1, k) + \epsilon_k^- \Pr(R = 0|E = 1, k)} \\
&= \frac{\epsilon_k^+ \Pr(R = 1)}{\epsilon_k^+ \Pr(R = 1) + \epsilon_k^- \Pr(R = 0)} \\
&= \frac{\epsilon_k^+}{\epsilon_k^+ + \epsilon_k^-}
\end{aligned}$$

The first step is based on Bayes law. The second is the definition of ϵ_k^+ and ϵ_k^-. The third step is because R is independent of E and k. The last step is based on a weak uninformative *prior* on actual relevance that $\Pr(R = 1) = \Pr(R = 0) = 0.5$. Putting it all together, we have

$$
\begin{aligned}
M_{Bayes-IPS} &= \sum_{(q,d,k,c=1)\in\mathcal{L}} \frac{1}{\theta_k} \Pr(R = 1|C = 1, k) f(q, d, \Omega_q) \\
&= \sum_{(q,d,k,c=1)\in\mathcal{L}} \frac{1}{\theta_k} \frac{\epsilon_k^+}{\epsilon_k^+ + \epsilon_k^-} f(q, d, \Omega_q).
\end{aligned}
$$

Informally, the Bayes-IPS estimate corrects for the examination bias and the likelihood that the result was actually relevant given that it was judged so upon examination. As a sanity check, note that in the noise-free case of the plain PBM, i.e. when $\epsilon_k^+ = 1$ and $\epsilon_k^- = 0$, $M_{Bayes-IPS}$ reduces to M_{IPS}. Moreover, since the proposed correction is simply a change of weights, a learning-to-rank framework with IPS can be easily adapted to use Bayes-IPS weights to replace IPS weights.

5.4 Empirical Evaluation

In this section, we evaluate our proposed TrustPBM click model and the Bayes-IPS correction for unbiased LTR. We first describe our experiment setup and then present our experimental results.

5.4.1 Experiment Setup

Our experiments are based on both offline evaluation and online experiments. For offline evaluation, we use the standard framework for supervised LTR evaluation [39] by splitting the data into training and test sets. For online experiments, we evaluate our proposed methods through online A/B testing. In this section, we describe the data sets and metrics used in offline evaluation as well as the metrics used in online experiments.

Data Sets

For offline evaluation, we use search logs collected from three search services: Gmail for general users (denoted as **Email**), Google Drive for general users (denoted as **File Storage**), and Gmail for expert users (denoted as **Email Expert**). The Email and Email Expert services have the same type of contents, but the users in the Email Expert dataset are more active, as measured by their search activity. The File Storage service has a different type of content compared with the other two. In all services, each query can result in at most a single click of a document because an overlay showing results is displayed as a query is typed and it disappears as a result is clicked. The clicked document is then shown on the screen.

In our data sets, we discard all the queries that did not result in any clicks. No further pre-processing is done. In all these services, ranking features are routinely computed and logged. We collected a sample from the search logs of each service in a two-week period in May 2018. Data from the first week are used for training, and those from the second week are used for testing. Basic statistics for the data sets are shown in Table 5.1. Each data set contains millions of queries, and between 5 and 6 documents per query. The logs also contain existing ranking features for each query-document pair, which we use in our regression-based EM algorithms and unbiased LTR algorithms.

Table 5.1: Statistics of the three data sets used in offline evaluation.

	Email	File Storage	Email Expert
#Queries per user	> 3	> 3	< 4
#Docs in total	59.69M	50.77M	13.80M
#Docs per query	5.99	5.00	6.00

Offline evaluation metrics

Our offline evaluation consists of two parts. The first part is to compare the PBM and TrustPBM models based on their fitness to the data. For this part, we use the standard log-likelihood on how the models predict clicks in our test data.

$$\text{LogLikelihood} = \frac{1}{|\mathcal{L}|} \sum_{\mathcal{L}} c \log(p) + (1 - c) \log(1 - p) \tag{5.2}$$

where $p = \theta_k \gamma_{q,d}$ for PBM and $p = \theta_k(\epsilon_k^+ \gamma_{q,d} + \epsilon_k^-(1 - \gamma_{q,d}))$ for TrustPBM. Log-likelihood is commonly used to evaluate click models [16]. It is a negative number and the larger the better.

For the second part, we compared different methods on the effectiveness of learned ranking functions for unbiased LTR. Following [55, 54], we use the weighted version of Mean Reciprocal Rank (MRR) as the evaluation metric for offline experiment. It is an unbiased version of MRR and more suitable than standard MRR for offline evaluation given click bias. Specifically, given a test data set with n queries, the standard MRR is defined as:

$$\text{MRR} = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\text{rank}_i} \tag{5.3}$$

where rank_i denotes the rank of the first clicked document of the i-th query in our data sets. Note that since each query can have at most one clicked document, we do not consider more complicated metrics like normalized discounted cumulative gain. We denote weighted MRR [55] by MRR$_w$, which is defined as:

$$\text{MRR}_w = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i \frac{1}{\text{rank}_i} \tag{5.4}$$

where w_i is the IPS correction weight estimated from our algorithm. Given different propensity estimation methods, the weights are used for both training and offline evaluation.

Online experiment metrics

For online experiments, we run A/B testing using ranking functions learned based on different IPS weights. This is the ultimate test of effectiveness of ranking functions. A better ranking function can attract more clicks from users and push the user clicks to higher positions. We thus use the click-through rate (CTR) that is defined as the ratio of queries with clicks and the total number of queries. We also use the standard MRR in Eq 5.3 for online evaluation to take into account the effect of positions. For any queries without clicks, their MRR values are set to 0 for online experiments.

5.4.2 Experimental Results

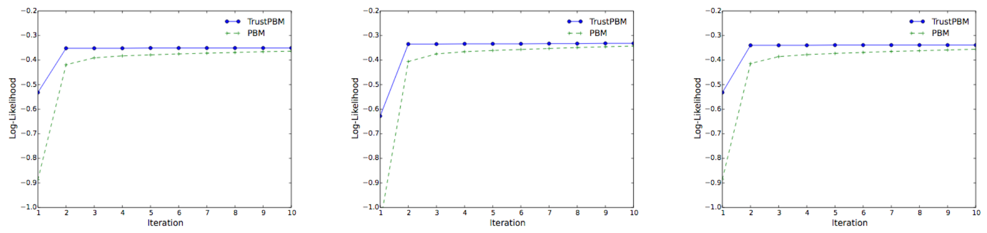
In this section, we report both offline and online experimental results. Our comparisons are conducted for both click modeling and unbiased LTR.

Table 5.2: Comparison of PBM and TrustPBM click models based on log-likelihood. The best number per row is in bold face and + means statistically significance based on the student t-test.

	PBM	TrustPBM	Improvement
Email	-0.364	**-0.351**	3.57%$^+$
File Storage	-0.343	**-0.332**	3.21%$^+$
Email Expert	-0.356	**-0.339**	4.78%$^+$

Does the extended model fit the data better?

Both PBM and TrustPBM are click models. Our first result is to compare them on how well they can fit our click data. We run our EM algorithms for both PBM and TrustPBM respectively and compare them in terms of log-likelihood in Eq 5.2. The results on our test data sets are shown in Table 5.2. From this table, we can see that the TrustPBM model has higher log-likelihood values than the PBM model on all the three data sets. We also report the relative improvement of the log-likelihood in this table. All these improvements are statistically significant based on the student t-test. This means that our proposed method of incorporating click noise in TrustPBM is reasonable since TrustPBM can indeed fit the data significantly better than PBM. In addition, we also study the convergence of the EM algorithms for PBM and TrustPBM in Figure 5.3, where we plot log-likelihood along with the EM iterations. It can be observed that TrustPBM not only converges to a higher log-likelihood, but also converges faster than PBM at early stages. This confirms that our proposed EM algorithm can estimate parameters for TrustPBM effectively. Overall, this result shows that TrustPBM is better that PBM.



(a) Email (b) File Storage (c) Email Expert

Figure 5.3: Convergence of EM based on log-likelihood.

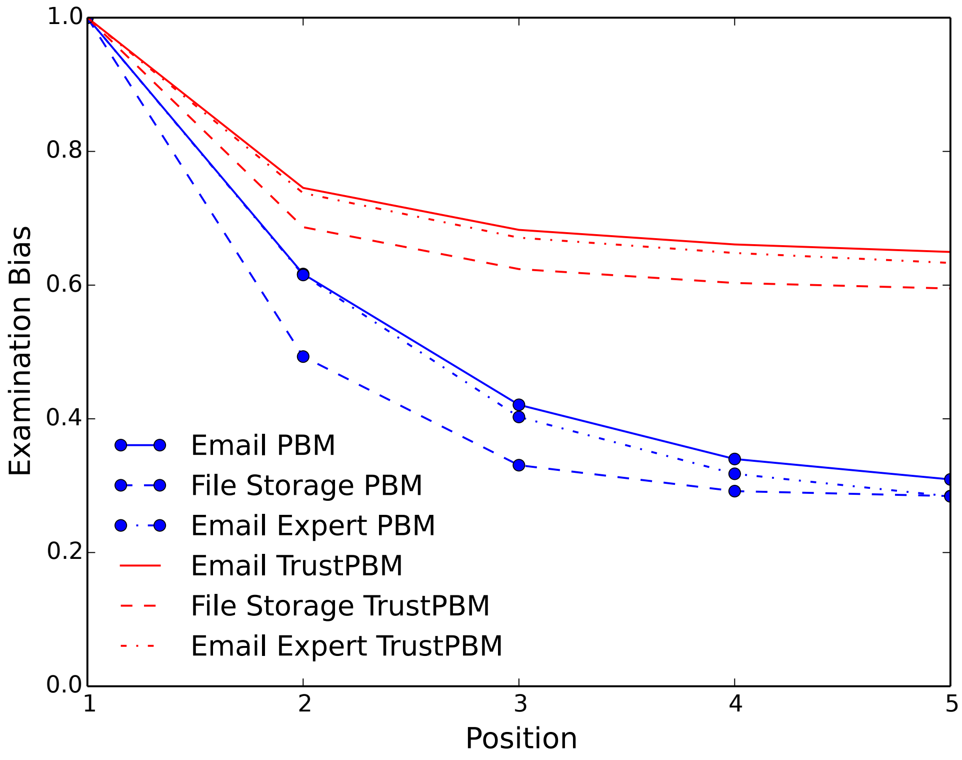


Figure 5.4: The normalized examination bias estimated from PBM and TrustPBM using EM.

What are the biases estimated from data?

We have examination bias for both PBM and TrustPBM. For TrustPBM, we also have the trust bias ϵ_k^+ and ϵ_k^-. We compare these estimated biases from data and show some interesting findings.

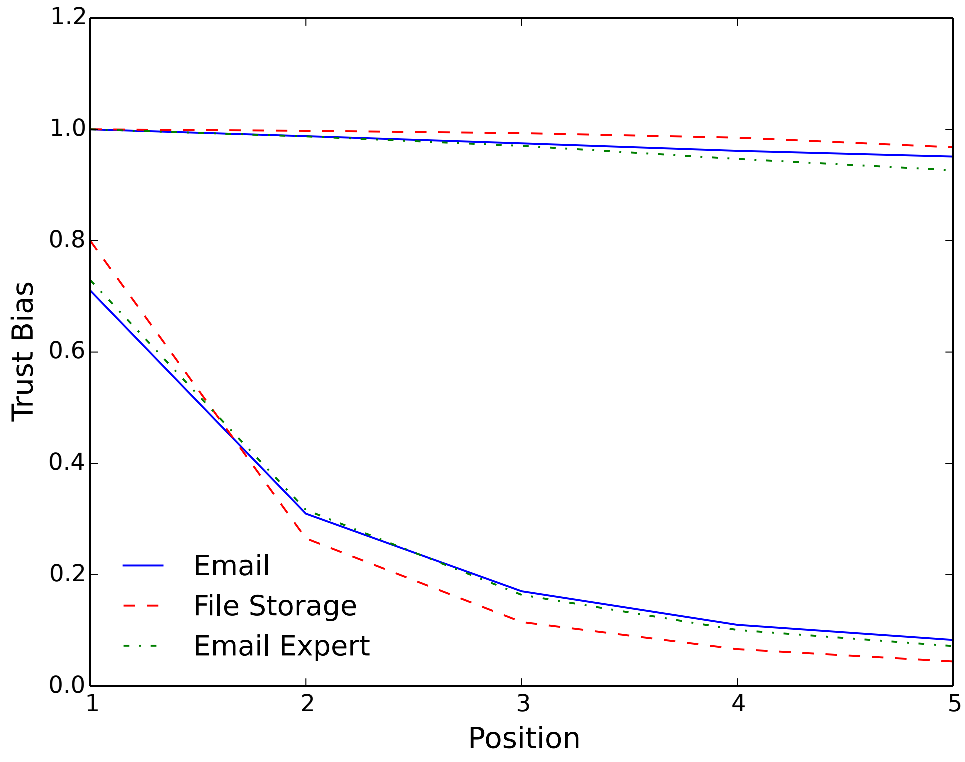


Figure 5.5: The normalized trust bias estimated for TrustPBM using EM. The upper lines are ϵ_k^+ and the lower lines are ϵ_k^-.

The examination biases for PBM and TrustPBM are plotted in Figure 5.4, where we normalize the bias values by the one at position 1. This figure shows that PBM and TrustPBM give quite different estimation of examination bias, especially for lower ranked documents. TrustPBM shows that there are higher probabilities that users examine lower ranked documents than PBM. This aligns better with the user study in [30] as the user study showed that users examine the surrounding documents before issuing clicks, especially the ones immediately below their clicked ones.

The trust bias estimated by TrustPBM is shown in Figure 5.5 where we normalize all of them by the ϵ_k^+ for $k = 1$. There are a few interesting observations.

(1) All ϵ_k^+ are higher than ϵ_k^-. This means that the probability of clicks given relevant documents is higher than the probability given non-relevant documents after examination. Thus clicks represent reasonable relevance judgments even though they are noisy. (2) ϵ_k^+ are very close to 1 over all positions k, and the difference over positions is very small, showing that relevant documents are not missed often once examined. (3) In contrast, ϵ_k^- varies a lot over positions k. It is much higher at the higher positions (lower values of k) than lower positions, meaning that users are more inclined to click non-relevant documents shown at high positions. This is congruent with the trust bias user study [30], which found that users may place more trust in highly-ranked results. (4) Across different data sets, the trust bias is different. For example, the File Storage data set has a steeper curve for ϵ_k^-, meaning that there is higher trust bias for this service. Overall, this result shows that TrustPBM is better that PBM qualitatively.

Is the Bayes-IPS correction different?

We now turn to the unbiased LTR evaluation. The inverse propensity weights estimated by different methods in the Email data set are shown in Figure 5.7. For result randomization, we use the FairPair approach in [55], that is, two adjacent documents are randomly swapped before presenting results to users. This method has been shown to be aligned well with other randomization methods for the PBM model. We use Randomization to denote the IPS weights estimated from this method. We use PBM to denote the IPS weights estimated by EM in PBM, and we TrustPBM to denote the Bayes-IPS weights estimated by EM in TrustPBM. Compared with TrustPBM, the EM-based PBM is much closer to Randomization. This result is expected since Randomization is also

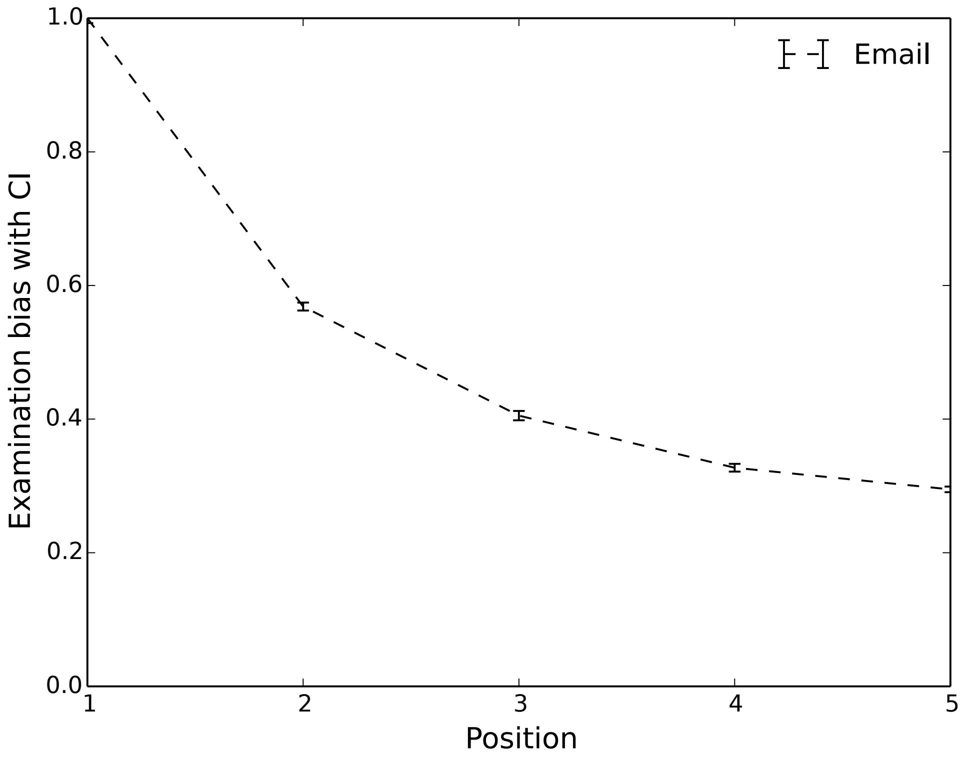


Figure 5.6: The examination bias and the confidence interval (CI) based on randomization data.

based on PBM. However, TrustPBM gives much smaller estimation of weights for lower positions. This is because TrustPBM uses a different model that disentangles examination and trust bias, leading to rather different estimation of inverse propensity weights.

The lower IPS weights at lower positions in TrustPBM is more desired. As commonly known, IPS-based approaches usually suffer from the problem of the unbounded variance since the variance for a smaller propensity (corresponding to the lower positions in unbiased LTR) can lead to a large difference in IPS weights. Several techniques such as capping methods are developed to address this problem [22, 51, 43, 18, 7]. For Randomization, we compute the variance of

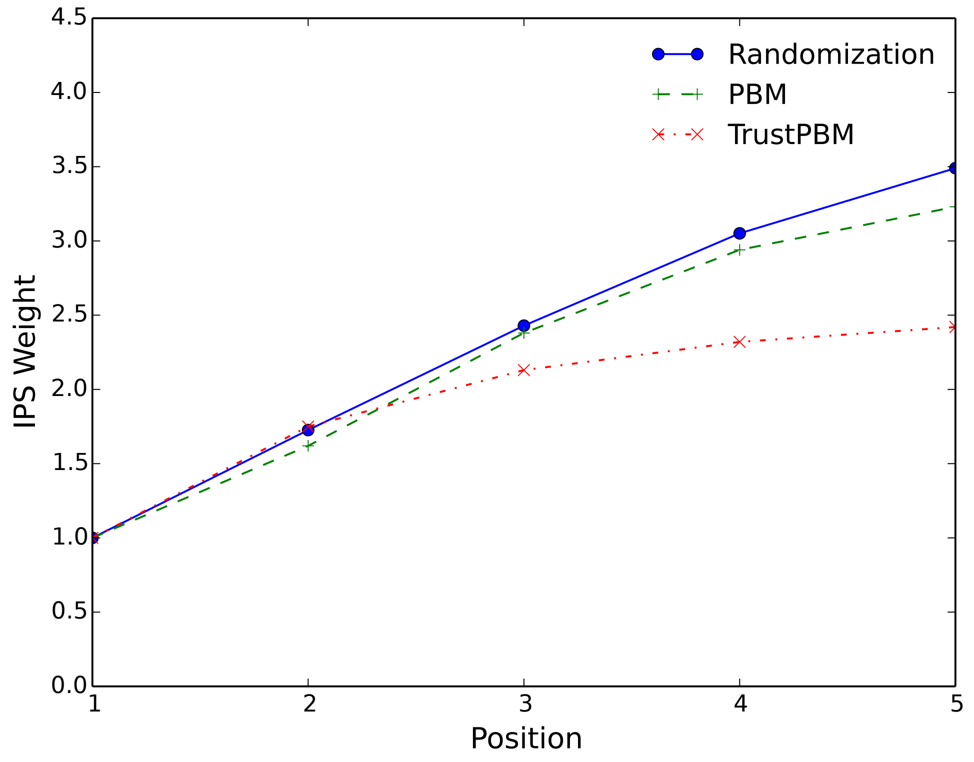


Figure 5.7: The IPS or Bayes-IPS weights based on Randomization (IPS), PBM (IPS) and TrustPBM (Bayes-IPS).

our estimated propensity θ_k as follows. We split our randomized data set into 10 folds and use them to estimate the confidence interval (CI). The results are show in Figure 5.6. From this figure, we see that there is a rather negligible variance based on CI. This means that we have sufficient randomized data for IPS weights estimation. As we will show later that TrustPBM leads to better online experiment results, the result here indicates that PBM is a less accurate model than TrustPBM for extracting relevance from user click behaviors.

In addition, Figure 5.8 shows the Bayes-IPS weights estimated for different data sets. The weights being different for different data sets means that our method can indeed take click noise into account that varies across data sets.

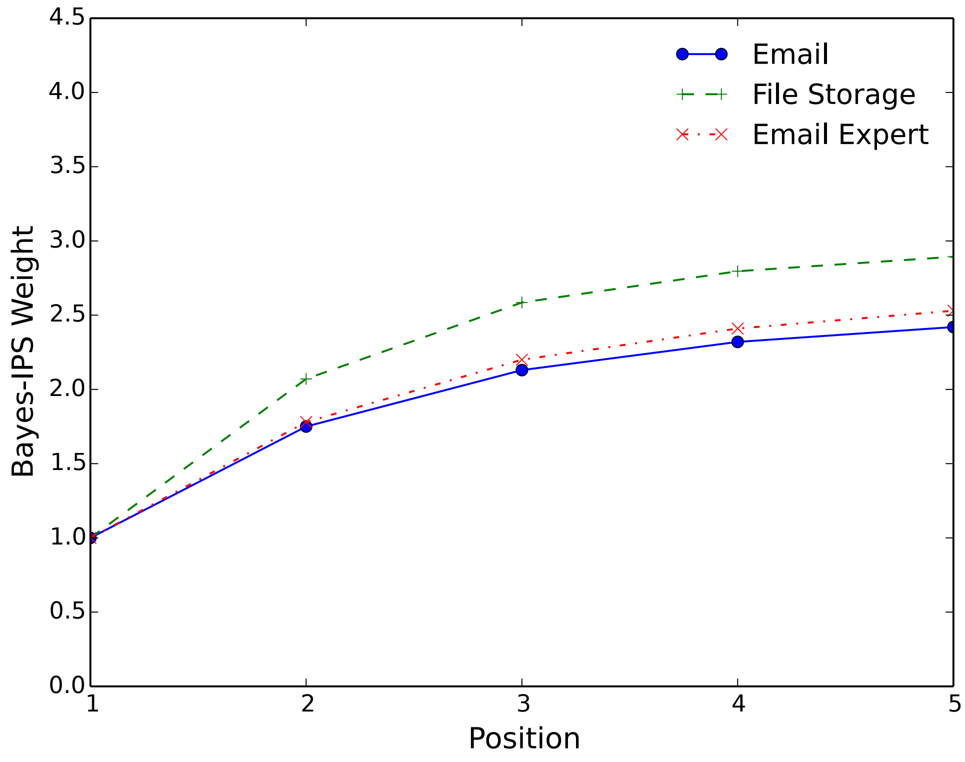


Figure 5.8: The Bayes-IPS weights estimated for different data sets.

Table 5.3: Cross offline evaluation to compare IPS weights for training and evaluation.

Training	Evaluation: Weighted MRR in Eq 5.4			
	NoIPS	Randomization	PBM	TrustPBM
NoIPS	**0.00%**	-0.50%	-0.36%	-0.20%
Randomization	-1.11%	**0.00%**	0.00%	-0.11%
PBM	-0.89%	-0.02%	**0.00%**	-0.06%
TrustPBM	-0.48%	-0.11%	-0.05%	**0.00%**

Can the learning algorithms leverage different IPS weights?

We have shown that different methods give different IPS weights. These IPS weights can be used for both evaluation and learning for unbiased LTR. In this sense, we do not have ground-truth IPS weights to compare different methods.

Table 5.4: Online experiment results based on LambdaMART models learned using different weights. Relative improvement over Randomization is reported, together with the confidence interval (CI). The best number per column is bold and + means the improvement is significant based on student t-test.

	MRR	MRR CI	CTR	CTR CI
Randomization	0.00%	-	0.00%	-
PBM	0.25%	±0.39%	0.26%	±0.40%
TrustPBM	**0.33%**$^+$	±0.29%	**0.33%**$^+$	±0.28%

We rely on the online experiments to do the comparison. Before doing so, we need to make sure that our learning algorithms can effectively leverage the IPS weights during training.

We use an industrial implementation of LambdaMART as our learning algorithm. Table 5.3 shows the cross comparison using different IPS weights for both training and evaluation. This comparison serves as a sanity check as to whether our training methods can take into account different settings of weights. All the values in Table 5.3 are relative numbers comparing to the diagonal lines per column. The diagonal values are expected to be the largest numbers per column since both training and evaluation use the same weights. The results in this table confirm that. For example, using the Bayes-IPS weights in TrustPBM, all models trained with different weights show negative relative evaluation numbers. Thus our learning algorithms are able to differentiate different weights to optimize their intended metrics.

Results of online experiments

We compare Randomization, PBM and TrustPBM using the live traffic of the Email service, conducting an online experiment for each method on a portion of the search traffic. These experiments were run for several weeks in August

2018 and our comparisons are based on approximately 100M user queries per experiment. For clarity, we normalize the online evaluation metrics using the Randomization method as the baseline.

We compute online metrics MRR and CTR; the results are shown in Table 5.4. In this table, we can see that TrustPBM improves over the Randomization method significantly, with MRR and CTR both increased by +0.33%, and these results are statistically significant. In contrast, PBM achieves neutral results compared with the Randomization baseline. Moreover, TrustPBM is more robust than PBM because it has a much smaller confidence interval (CI). This is probably due to the fact that having higher weights on lower positions can lead to higher variance when learning a ranking function for the PBM model. By addressing click noise, TrustPBM is more robust.

5.5 Conclusion

In this chapter, we proposed to handle click noise for unbiased LTR. We have introduced an enhanced position bias model called TrustPBM which explicitly models position-dependent click noise, conceptualized as a result of the trust users have in the quality of the search ranking. The parameters of the TrustPBM are estimated from offline click logs via regression-based Expectation Maximization, and the novel Bayes-IPS correction provides a simple weighting scheme for unbiased, de-noised LTR using the estimated parameters. Experiments on three widely used commercial search services revealed new insights about the asymmetry in positive and negative trust bias, and super performance metrics with the Bayes-IPS correction.

CHAPTER 6

CONCLUSION AND FUTURE WORK

We have addressed the problem of learning-to-rank (LTR) directly from logged implicit user feedback such as click logs such that the learned ranking systems are optimized for unbiased relevance-based performance metrics. As we have seen, this is challenging due to various forms of position bias inherent in the logged data. We address this challenge through a systematic framework comprising of user behavior modeling, effective and efficient bias estimation techniques, and novel ranker training procedures.

Specifically, we proposed a counterfactual learning-to-rank framework that is general enough to cover a broad class of additive IR metrics as well as non-linear deep network models. Based on the generalized framework, we developed the SVM PropDCG and Deep PropDCG methods that optimize DCG via the Convex-Concave Procedure (CCP) and stochastic gradient descent respectively. We then presented the idea of Intervention Harvesting, allowing the use of multiple historic loggers for generating interventional data under mild assumptions. We showed how this idea can be used to control for relevance, providing the first method for propensity estimation in the PBM that does not require intrusive interventions, a relevance model, or repeat queries. Finally, we introduced an enhanced position bias model called TrustPBM which explicitly models position-dependent click noise, conceptualized as a result of the trust users have in the quality of the search ranking. The parameters of the TrustPBM are estimated from offline click logs via regression-based Expectation Maximization, and the novel Bayes-IPS correction provides a simple weighting scheme for unbiased, de-noised LTR using the estimated parameters. Experiments on three widely used commercial search services revealed new insights about the asymmetry in positive and negative trust bias.

There are many directions for future work. It is open for which other classes of ranking metrics it is possible to develop efficient and effective methods using the generalized counterfactual framework. The general counterfactual learning approach may also provide unbiased learning objectives for other settings beyond ranking, like full-page optimization and browsing-based retrieval tasks. It is also an open question whether non-differentiable (e.g. tree-based) ranking models can be trained in the counterfactual framework as well.

Our work also opens an interesting space of research questions for how other intervention data can be harvested and what other estimation problems can be addressed in this way. In particular, in our follow-up work in [20], we show how Intervention Harvesting can be adapted to a more expressive Contextual Position-Based Model (CPBM) in which the examination propensity can vary based on query, user and other context observable information in addition to position. Another direction for future work is finding alternate methods for TrustPBM estimation, such as via intervention harvesting adapted for trust bias, and also designing alternate schemes for unbiased LTR that weaken the prior assumption in our Bayes-IPS correction.

BIBLIOGRAPHY

[1] Aman Agarwal, Soumya Basu, Tobias Schnabel, and Thorsten Joachims. Effective evaluation using logged bandit feedback from multiple loggers. In *Proc. of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD, pages 687–696, 2017.

[2] Aman Agarwal, Kenta Takatsu, Ivan Zaitsev, and Thorsten Joachims. A general framework for counterfactual learning-to-rank. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 5–14, New York, NY, USA, 2019. Association for Computing Machinery.

[3] Aman Agarwal, Xuanhui Wang, Cheng Li, Michael Bendersky, and Marc Najork. Addressing trust bias for unbiased learning-to-rank. In *The World Wide Web Conference*, WWW '19, page 4–14, New York, NY, USA, 2019. Association for Computing Machinery.

[4] Aman Agarwal, Ivan Zaitsev, and Thorsten Joachims. Counterfactual learning-to-rank for additive metrics and deep models. In *ICML Workshop on Machine Learning for Causal Inference, Counterfactual Prediction, and Autonomous Action*, CausalML, 2018.

[5] Aman Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Li, Marc Najork, and Thorsten Joachims. Estimating position bias without intrusive interventions. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, page 474–482, New York, NY, USA, 2019. Association for Computing Machinery.

[6] Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W Bruce Croft. Unbiased learning to rank with unbiased propensity estimation. In *Proc. of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR, pages 385–394, 2018.

[7] Léon Bottou, Jonas Peters, Joaquin Qui nonero Candela, Denis X. Charles, D. Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research*, 14:3207–3260, 2013.

[8] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent.

In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 89–96, New York, NY, USA, 2005. ACM.

[9] Chris J.C. Burges. From ranknet to lambdarank to lambdamart: An overview. Technical Report MSR-TR-2010-82, Microsoft Research, June 2010.

[10] Christopher J Burges, Robert Ragno, and Quoc V Le. Learning to rank with nonsmooth cost functions. In *Advances in neural information processing systems*, pages 193–200, 2007.

[11] Ben Carterette and Praveen Chandar. Offline comparative evaluation with incremental, minimally-invasive online feedback. In *Proc. of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR, pages 705–714, 2018.

[12] Praveen Chandar and Ben Carterette. Estimating clickthrough bias in the cascade model. In *Proc. of the 27th ACM International Conference on Information and Knowledge Management*, CIKM, pages 1587–1590, 2018.

[13] Olivier Chapelle, Thorsten Joachims, Filip Radlinski, and Yisong Yue. Large-scale validation and analysis of interleaved search evaluation. *ACM Transactions on Information Systems (TOIS)*, 30(1):6:1–6:41, 2012.

[14] Olivier Chapelle and Mingrui Wu. Gradient descent optimization of smoothed information retrieval metrics. *Inf. Retr.*, 13(3):216–235, June 2010.

[15] Olivier Chapelle and Ya Zhang. A dynamic Bayesian network click model for web search ranking. In *Proceedings of the 18th International Conference on World Wide Web*, WWW, pages 1–10, 2009.

[16] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. *Click Models for Web Search*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2015.

[17] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. An experimental comparison of click position-bias models. In *Proc. of the 1st International Conference on Web Search and Data Mining*, WSDM, pages 87–94, 2008.

[18] Miroslav Dudík, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. In *Proc. of the 28th International Conference on Machine Learning*, ICML, pages 1097–1104, 2011.

[19] Georges E. Dupret and Benjamin Piwowarski. A user browsing model to predict search engine click data from past observations. In *Proc. of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, pages 331–338, 2008.

[20] Zhichong Fang, Aman Agarwal, and Thorsten Joachims. Intervention harvesting for context-dependent examination-bias estimation. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR'19*, 2019.

[21] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[22] Alexandre Gilotte, Clément Calauzènes, Thomas Nedelec, Alexandre Abraham, and Simon Dollé. Offline a/b testing for recommender systems. In *Proc. of the 11th ACM International Conference on Web Search and Data Mining*, WSDM, pages 198–206, 2018.

[23] Fan Guo, Chao Liu, Anitha Kannan, Tom Minka, Michael Taylor, Yi-Min Wang, and Christos Faloutsos. Click chain model in web search. In *Proc. of the 18th International Conference on World Wide Web*, WWW, pages 11–20, 2009.

[24] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[25] Ziniu Hu, Yang Wang, Qu Peng, and Hang Li. A novel algorithm for unbiased learning to rank, 2018.

[26] G. Imbens and D. Rubin. *Causal Inference for Statistics, Social, and Biomedical Sciences*. Cambridge University Press, 2015.

[27] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[28] T. Joachims. Optimizing search engines using clickthrough data. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142, 2002.

[29] T. Joachims, T. Finley, and Chun-Nam Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

[30] Thorsten Joachims, Laura A. Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proc. of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, pages 154–161, 2005.

[31] Thorsten Joachims, Laura A. Granka, Bing Pan, Helene Hembrooke, Filip Radlinski, and Geri Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems (TOIS)*, 25(2), April 2007.

[32] Thorsten Joachims, Adith Swaminathan, and Maarten de Rijke. Deep learning with logged bandit feedback. In *6th International Conference on Learning Representations*, ICLR, 2018.

[33] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. Unbiased learning-to-rank with biased feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 781–789, New York, NY, USA, 2017. ACM.

[34] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3146–3154. Curran Associates, Inc., 2017.

[35] John Langford, Alexander Strehl, and Jennifer Wortman. Exploration scavenging. In *Proc. of the 25th International Conference on Machine Learning*, ICML, pages 528–535, 2008.

[36] Lihong Li, Shunbao Chen, Jim Kleban, and Ankur Gupta. Counterfactual estimation and optimization of click metrics for search engines, 2014.

[37] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proc. of the 4th International Conference on Web Search and Web Data Mining*, WSDM, pages 297–306, 2011.

[38] Thomas Lipp and Stephen Boyd. Variations and extension of the convex–concave procedure. *Optimization and Engineering*, 17(2):263–287, 2016.

[39] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[40] Alistair Moffat and Justin Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Transactions on Information Systems (TOIS)*, 27(1):2:1–2:27, 2008.

[41] Maeve O'Brien and Mark T. Keane. Modeling user behavior using a search-engine. In *Proc. of the 12th International Conference on Intelligent User Interfaces*, IUI, pages 357–360, 2007.

[42] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. Deeprank: A new deep architecture for relevance ranking in information retrieval. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 257–266. ACM, 2017.

[43] M. J. D. Powell and J. Swann. Weighted uniform sampling — a monte carlo technique for reducing variance. *IMA Journal of Applied Mathematics*, 2(3):228–236, 1966.

[44] Tao Qin and Tie-Yan Liu. Introducing letor 4.0 datasets. *CoRR*, abs/1306.2597, 2013.

[45] Matthew Richardson, Ewa Dominowska, and Robert Ragno. Predicting clicks: Estimating the click-through rate for new ads. In *Proc. of the 16th International Conference on World Wide Web*, WWW, pages 521–530, 2007.

[46] L. Rigutini, T. Papini, M. Maggini, and F. Scarselli. Sortnet: Learning to rank by a neural preference function. *IEEE Transactions on Neural Networks*, 22(9):1368–1380, Sept 2011.

[47] Paul R. Rosenbaum and Donald B. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

[48] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. Recommendations as treatments: Debiasing learning and evaluation. In *Proceedings of the 33nd International Conference on Machine Learning*, ICML, pages 1670–1679, 2016.

[49] B. Schoelkopf and A. J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, 2002.

[50] A. Swaminathan and T. Joachims. Batch learning from logged bandit feed-

back through counterfactual risk minimization. *Journal of Machine Learning Research (JMLR)*, 16:1731–1755, Sep 2015.

[51] A. Swaminathan and T. Joachims. The self-normalized estimator for counterfactual learning. In *Neural Information Processing Systems (NIPS)*, 2015.

[52] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. Softrank: Optimizing non-smooth rank metrics. *WSDM '08*, 2008.

[53] V. Vapnik. *Statistical Learning Theory*. Wiley, Chichester, GB, 1998.

[54] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. Learning to rank with selection bias in personal search. In *Proc. of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, pages 115–124, 2016.

[55] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. Position bias estimation for unbiased learning to rank in personal search. In *Proc. of the 11th ACM International Conference on Web Search and Data Mining*, WSDM, pages 610–618, 2018.

[56] Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. The lambdaloss framework for ranking metric optimization. In *Proc. of the 27th ACM International Conference on Information and Knowledge Management*, CIKM, pages 1313–1322, 2018.

[57] Mingrui Wu, Yi Chang, Zhaohui Zheng, and Hongyuan Zha. Smoothing dcg for learning to rank: A novel approach using smoothed hinge functions. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1923–1926, New York, NY, USA, 2009. ACM.

[58] Yisong Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 271–278, 2007.

[59] Yisong Yue, Rajan Patel, and Hein Roehrig. Beyond position bias: examining result attractiveness as a source of presentation bias in clickthrough data. In *International Conference on World Wide Web (WWW)*, pages 1011–1018. ACM, 2010.