# Learning to Rank with Selection Bias in Personal Search

Xuanhui Wang, Michael Bendersky, Donald Metzler, Marc Najork
Google Inc.
Mountain View, CA 94043
{xuanhui, bemike, metzler, najork}@google.com

## ABSTRACT

Click-through data has proven to be a critical resource for improving search ranking quality. Though a large amount of click data can be easily collected by search engines, various biases make it difficult to fully leverage this type of data. In the past, many click models have been proposed and successfully used to estimate the relevance for individual query-document pairs in the context of web search. These click models typically require a large quantity of clicks for each individual pair and this makes them difficult to apply in systems where click data is highly sparse due to personalized corpora and information needs, e.g., personal search. In this paper, we study the problem of how to leverage sparse click data in personal search and introduce a novel *selection bias* problem and address it in the learning-to-rank framework. This paper proposes a few bias estimation methods, including a novel query-dependent one that captures queries with similar results and can successfully deal with sparse data. We empirically demonstrate that learning-to-rank that accounts for query-dependent selection bias yields significant improvements in search effectiveness through online experiments with one of the world's largest personal search engines.

## Keywords

Personal Search; Selection Bias; Learning-to-Rank

## 1. INTRODUCTION

In the past several years, click-through data has become an indispensable resource for online information retrieval services. It provides a natural, abundant and continuously renewable source of user feedback. However, despite its tremendous value, click-through data is inherently biased and very noisy. Previous research shows that in order to reliably leverage click-through data one has to account for multiple sources of bias including: position bias [22], presentation bias [33], and trust bias [28]. Therefore, directly using click-through data may result in noisy and biased training data, which will negatively impact the downstream applications [21]. As a result, there has been a great deal of research on extracting reliable signals from click-through data [10].

Previous work has typically focused on click modeling to estimate relevance for individual query-document pairs. For instance, Craswell et al. [11] proposed the Cascade model, in which the conditional probability of a click on a document $\mathbf{x}$ given position $i$ is predicated on the marginal probability of the document $\mathbf{x}$ being relevant to the query and the marginal probabilities of documents at positions $1, \ldots, i-1$ being non-relevant to the query. In order to estimate these marginals, the click models often assume access to large quantities of click data for each document $\mathbf{x}$ given the query [8, 11, 15]. Such models have generally proven to be successful in the context of web search, where this assumption holds. However, it is less clear how they can be applied in search scenarios where click data is highly sparse.

One such scenario that is the focus of this paper is *personal search*. Personal search is an important and well studied information retrieval task with applications such as email search [6], desktop search [13], and, most recently, on-device search [23]. One important difference between personal and web search is that in the personal search scenario each user has access only to their own *private* document corpus (e.g., emails, files, or mobile application data). Therefore, the vast majority of the existing click models that learn click probabilities from large quantities of clicks for individual query-document pairs are not applicable in the personal search scenario.

Another important challenge in the context of personal search is the collection of explicit relevance judgments. Collection of TREC-like document relevance judgments by third party raters (that are commonly used in other information retrieval tasks) such as LETOR data set [29] are difficult to obtain due to privacy restrictions. In addition, since each user will have their own unique set of information needs and documents that evolve over time (e.g., new emails arrive every day), explicit relevance judgments may be prohibitively costly to maintain. Therefore, development of ranking models in general, and specifically learning-to-rank models [26] that utilize click-through data as a *noisy and biased* source of relevance data becomes essential for personal search.

In this paper, we study the the problem of learning-to-rank from click data in personal search. Different from the majority of past click modeling work whose focus is on estimating the relevance for individual query-document pairs, we propose a novel *selection bias* problem in the context of the learning-to-rank from click data. The basic idea of the



*SIGIR '16 July 17-21, 2016, Pisa, Italy*



ACM ISBN 978-1-4503-4069-4/16/07.

DOI: http://dx.doi.org/10.1145/2911451.2911537

selection bias problem is that queries are under-sampled to different extents and thus biased when click data is collected to learn ranking functions. We propose several methods to estimate this bias. We begin with a *global* bias model and refine it to a *segmented* bias model. We show that such a segmented bias model gives rise to a general framework that defines a *query-dependent* bias, where every query (and its associated result set) can be associated with a potentially different bias model. This general query-dependent framework is especially powerful in the personal search scenario as it allows accurate bias estimation without explicit access to a large number of clicks for any given query-document pair. To the best of our knowledge, this is the first study that both proposes a theoretical framework for eliminating selection bias in personal search and provides an extensive empirical evaluation using large-scale live experiments.

Our primary contributions of this paper can be summarized as follows:

- We propose the problem of selection bias and address it when applying learning-to-rank to click data.
- We propose several novel bias prediction methods, including a query-dependent model that does not need a large quantity of click data for any given query-document pair.
- We propose a novel, unbiased and theoretically sound offline evaluation methodology for our problem.
- We verify the effectiveness of the proposed methods in the context of personal search through rigorous offline experiments and large-scale online experiments.

The remainder of the paper is organized as follows. In Section 2, we review previous related work. Our problem is formally defined in Section 3. Different methods of quantifying selection bias are described in Section 4. We present our extensive experimental study and our evaluation methodology in Section 5. Finally, we conclude and discuss future work in Section 6.

## 2. RELATED WORK

There is an abundance of prior work on interpreting clicks as implicit feedback from users. One of the seminal papers in the field by Joachims et al. [22] evaluates the reliability of click-through signals via a user study. The overall conclusion of the study is that clicks are indeed useful for implicit feedback interpretation as long as certain biases are accounted for, including "trust bias" (commonly referred to as "position bias" in later work) that leads to more clicks on higher-ranked results, and "quality bias" in which the click behavior is influenced by the overall quality of the ranked list.

Joachims et al. [22] also proposed five simple strategies to eliminate these biases, including the Click > Skip Above strategy, which gave rise to the well-known Cascade model [11]. Later work also introduced a variety of click modeling techniques including, among many others, a dynamic Bayesian network click model [8], click chain model [17], session utility model [14], whole page click model [9], multiple browsing model [15], and a general click model [36]. A recent survey by Chuklin et al. [10] provides a good overview of the latest advances in the field.

While both click models in prior work and selection bias estimation presented in this paper focus on deriving useful implicit feedback from click-through data, there are several important differences. First, while the majority of the past click modeling work focuses on estimating the degree of relevance between a query and a document, the main goal of this paper is to study the selection bias problem in which click data used in learning-to-rank may drift away from the true underlying distribution. Second, the existing click models assume that a sufficient amount of click data is available for each query-document pair in each position to reliably train the model parameters. While this assumption holds in the web search setting, it is not feasible in domains like desktop search, email search or enterprise search, where each user might have access to a different set of documents, and it is impossible to leverage the "wisdom of the crowds" to aggregate clicks across users. Third, the click models are generally evaluated using the perplexity between the estimated and observed clicks [9, 15]. In contrast, we directly evaluate the ranking effectiveness of our methods through offline evaluation and online live experiments.

Click data is extensively used in sponsored search where the main goal is to predict the click-through rate for ads (e.g., [30, 36]). Noticeably, our selection bias estimation methods are related to the position bias model in [30]. The main difference is that we use the estimated bias to address the selection bias problem, while the bias was used as the expected ads impressions when computing the click-through rate of ads using data from multiple positions. Furthermore, we also propose more advanced query-dependent bias models that are more tractable even in scenarios where training data can be scarce due to small sample sizes, low search volume or personal document collections.

Our bias estimation models rely on the randomized experimental data. Order randomization removes the position biases that are inherent to click data, and therefore one can view the proposed models as propensity score [31] estimates. Furthermore, randomized data is the basis for our proposed unbiased offline evaluator. Similar evaluation methodologies were proposed by Li et al. in prior work [24, 25]. The difference is that we also use the randomized data for selection bias estimation to improve ranking functions, which is not the case in the past work.

Order randomization also eliminates, to a certain degree, the selection bias inherent in many information retrieval applications that employ pooling of top retrieved results [27]. Previous work [5] proposed methods to avoid the selection bias in TREC-style evaluation settings. However, such approaches do not easily extend to the online evaluation case, which is addressed in this work.

The methods proposed in this paper can also be viewed as a novel extension of sample selection bias correction methods, which are well-studied in the context of regression and classification [20, 34, 32], in the online learning-to-rank setting. In contrast to previously proposed learning-to-rank models that make explicit assumptions about user behavior [19] or use heuristic-based method for document selection [1], we learn the selection bias directly from experimental data.

## 3. PROBLEM FORMULATION

In this section, we introduce the selection bias problem for learning-to-rank in personal search scenarios. We begin by briefly reviewing the general setting of learning-to-rank.

## 3.1 Learning-to-Rank

Let $Q = (q, \{\mathbf{x}_1, ..., \mathbf{x}_n\})$ denote a query string $q$ and its set of result documents. We write $\mathbf{x} \in Q$ to indicate that $\mathbf{x}$ is in the result set of $Q$. Let $P(Q)$ denote the probability of observing query $Q$, based on the underlying distribution of queries in the universe $\mathcal{Q}$ of all possible queries that users can issue together with all possible result combinations. The goal of learning-to-rank is to find a scoring function $f(\mathbf{x})$ that can minimize the loss function defined as:

$$\mathrm{L}(f) = \int_{Q \in \mathcal{Q}} l(Q, f)\, \mathrm{d}P(Q) \tag{1}$$

where $l(Q, f)$ is the incurred loss of scoring function $f$ applied to query $Q$. Let $\mathbf{x}_i \succ_Q \mathbf{x}_j$ denote all pairs $\mathbf{x}_i, \mathbf{x}_j$ of result documents in $Q$ for which $\mathbf{x}_i$ is more relevant than $\mathbf{x}_j$. An example of a pair-wise loss function used in [35] is defined as:

$$l(Q, f) = \sum_{\mathbf{x}_i \succ_Q \mathbf{x}_j} \max(0, f(\mathbf{x}_j) - f(\mathbf{x}_i))^2 \tag{2}$$

The intuition behind this loss function is to penalize the out-of-order pairs when ranked by $f$.

In practice, the distribution of queries in $\mathcal{Q}$ is is unknown and the empirical loss defined over a uniformly random sample $\mathcal{U} = \{Q \in \mathcal{Q} : Q \sim P(Q)\}$ is used as the objective function for learning.

$$\mathrm{L}_{\mathcal{U}}(f) = \frac{1}{|\mathcal{U}|} \sum_{Q \in \mathcal{U}} l(Q, f) \tag{3}$$

Most learning-to-rank algorithms differ in how the loss function $l(Q, f)$ is defined [26]. Generally, the state-of-the-art loss functions are pair-wise or list-wise. Practically, pair-wise loss functions tend to be more efficient for training and have been widely adopted by large search engines [4, 35]. Thus, in the rest of the paper, we make the assumption that a pair-wise loss function (e.g., Eq 2) is being used. However, it is important to point out that the methods described in this paper are general enough to be applied to list-wise loss functions as well.

## 3.2 Selection Bias Problem

The data set $\mathcal{U}$ in Eq 3 is the training data used to learn the scoring function $f(\mathbf{x})$. There are two commonly used approaches to obtain relevance estimates for $\mathcal{U}$. One way is to sample a set of queries and ask human raters to *explicitly* judge the relevance of the retrieved documents. The other way is to collect *implicit* relevance judgments such as click-through data. The click-through data approach has attracted the attention of the research community [21], as it is much cheaper to obtain than human-judged data, especially for major search engines. However, as we mentioned before, click data is biased and very noisy. For example, because of *position bias*, simple click counts can not be used directly to estimate relevance. A great deal of previous work (see Section 2) focuses on overcoming such bias to infer actual (or unbiased) relevance. Our focus is on the more general *selection bias* problem that arises when using click-through data to train learning-to-rank models.

**Observation 1** *When using click-through data for learning-to-rank, queries without clicks provide no useful information when optimizing pair-wise loss functions.*

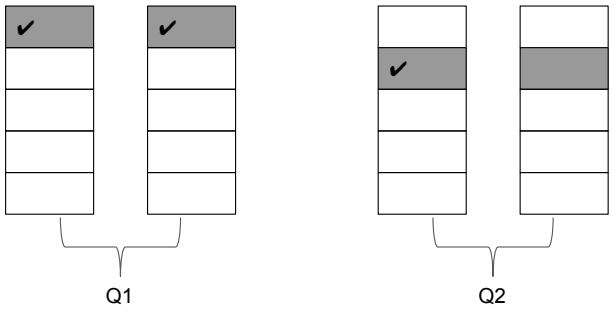


**Figure 1: An illustration of selection bias in click data. The shaded documents are the relevant ones. A check mark means the document is clicked.**

For example, consider Eq 2. When there are no clicks for query $Q$, the set $\mathbf{x}_i \succ_Q \mathbf{x}_j$ is empty since there is no way to derive preferences between any pairs of documents. Such an observation can be generalized to list-wise loss functions as well. In the following, we focus on the collection of queries with clicks and use $\mathcal{S}$ to denote this collection.

**Observation 2** *The collection of queries $\mathcal{S}$ is biased. Formally, let $\hat{P}(Q)$ denote the probability mass of query $Q$ in $\mathcal{S}$, then $\hat{P}(Q) \neq P(Q)$.*

We use an example to better explain this observation. In Figure 1, we have two queries $Q_1$ and $Q_2$ that both have equal probability of being issued by users, i.e., $P(Q_1) = P(Q_2)$ as they have equal probability in $\mathcal{U}$. The relevant document for $Q_1$ is at position 1 and is clicked every time the query is issued. On the contrary, the relevant document for $Q_2$ is at position 2 and is clicked half of the time when the query is issued. Thus, $\hat{P}(Q_2) = \frac{1}{2}\hat{P}(Q_1)$ in $\mathcal{S}$, which helps illustrate how selection bias may arise in click data. The problem illustrated in this example is rooted at the commonly known *position bias* and confirmed by eye tracking studies [22, 30] as well, which found that the users are less likely to see, and hence click on, lower-ranked documents.

## 3.3 Inverse Propensity Weighting

Selection bias is a widely known problem in many other scientific communities, such as the health care field in which the problem arises in clinical trial studies [31]. Many methods such as propensity matching, inverse propensity weighting, and doubly robust estimation have been applied in online settings [7] with the goal of comparing the effect of a treatment *vs.* its control (e.g., showing *vs.* not showing the ad). Some methods, including propensity matching, are specifically designed for comparison: given any individual in the treatment group, match it with another individual in the control group in the sense of their propensity scores being equal; the effect is obtained as the difference between the average of the treatment group and the matched individuals from the control group. It is not immediately clear how to adapt these methods to our use-case.

On the other hand, the inverse propensity weighting approach can easily be adopted to help overcome selection bias for learning-to-rank. With inverse propensity weighting, $\hat{P}(Q)$ is known as the *propensity score* of $Q$. Let $w_Q = P(Q)/\hat{P}(Q)$, i.e. the ratio between the probability of $Q$ appearing in $\mathcal{U}$ and the probability that $Q$ actually appears in

$\mathcal{S}$. Then the empirical loss function becomes:

$$\mathrm{L}_{\mathcal{S}}(f) = \frac{1}{|\mathcal{S}|} \sum_{Q \in \mathcal{S}} \frac{P(Q)}{\hat{P}(Q)} l(Q, f) = \frac{1}{|\mathcal{S}|} \sum_{Q \in \mathcal{S}} w_Q \cdot l(Q, f)$$

To the best of our knowledge, our work is the first to generally study selection bias to improve the effectiveness of learning-to-rank models. The problem of selection bias is especially important in the scenario of personal search where the personalized nature of information needs strongly biases the available training data.

To apply selection bias in practice, the primary challenge becomes estimating the inverse propensity weights $w_Q$. An open question is also whether such a weighting approach will have a significant impact on the effectiveness of learning-to-rank models. We address this challenge and answer this question in the following sections.

# 4. PROPOSED METHODS

Before we describe different methods of inverse propensity weighting estimation, we briefly describe our application scenario and the data set used to quantify the bias.

**Application Scenario.** Our application is a search engine for one of the world's largest commercial email and cloud file storage services. Given a query, the search engine provides instant results (i.e., the results refresh as the user types). These instant results provide an efficient way for the user to examine the results. There are up to $n$ instant results *with no* pagination. The results are retrieved from a personal corpus (e.g. emails or cloud storage files) and therefore are generally unique to the user. Once the user detects a relevant result and clicks on it, the clicked document is immediately opened in the browser.

Our click data is obtained exclusively from the instant results. Therefore, for each issued query, there will be either no click or exactly one click. In the rest of the paper, we study the selection bias problem in this setting. While the methods presented here can easily be extended to the web search setting, it is beyond the scope of the current study.

**Result Randomization.** In order to quantify the position bias, which will be used for inverse propensity weighting estimation, we employ result randomization and collect user click data on the randomized result sets. Specifically, given a ranked result list of $n$ documents returned for some query, instead of showing the original list, we permute the results uniformly at random and present the shuffled list to a small fraction of end users. We denote the collected randomized data by $\mathcal{R}$. As a special case, when $n = 2$, the randomization reduces to the previously proposed FairPairs algorithm [11, 33].

In the rest of this section, we present different methods to quantify the selection bias using the collected randomized data.

## 4.1 Global Bias Model

The global bias model can be viewed as the standard position bias model [11, 30]. It assumes that the bias is a function of the position within the ranked list itself. Formally, let $c^Q_{\mathbf{x}i}$ denote the probability of receiving a click when a document $\mathbf{x}$ is shown at position $i$ for query $Q$, $r^Q_{\mathbf{x}}$ be the probability of relevance of $\mathbf{x}$ to $Q$, and $b_i$ be the bias at position $i$ (meaning how likely a user is to examine the document at this position), then it follows that:

$$c^Q_{\mathbf{x}i} = r^Q_{\mathbf{x}} \cdot b_i$$

Our goal is to estimate $b_i$ for $1 \le i \le n$. Given query $Q$, $P(\mathbf{x}|Q, i)$ denotes the probability of showing result $\mathbf{x} \in Q$ at position $i$. In the randomized data, the probability of showing a given result $\mathbf{x} \in Q$ is the same for all positions, i.e., $P(\mathbf{x}|Q, i_1) = P(\mathbf{x}|Q, i_2)$ for all $1 \le i_1, i_2 \le n$. Thus, given $Q$, for all $1 \le i_1, i_2 \le n$:

$$\int_{\mathbf{x} \in Q} r^Q_{\mathbf{x}} \, \mathrm{d}P(\mathbf{x}|Q, i_1) = \int_{\mathbf{x} \in Q} r^Q_{\mathbf{x}} \, \mathrm{d}P(\mathbf{x}|Q, i_2)$$

Hence,

$$b_i = \frac{\sum_{Q \in \mathcal{R}} \int_{\mathbf{x} \in Q} c_{\mathbf{x}i} \, \mathrm{d}P(\mathbf{x}|Q, i)}{\sum_{Q \in \mathcal{R}} \int_{\mathbf{x} \in Q} r_{\mathbf{x}} \, \mathrm{d}P(\mathbf{x}|Q, i)} \propto \sum_{Q \in \mathcal{R}} \int_{\mathbf{x} \in Q} c_{\mathbf{x}i} \, \mathrm{d}P(\mathbf{x}|Q, i),$$

which is the total number of clicks at position $i$ in the randomized data $\mathcal{R}$. In our application, every query in $\mathcal{S}$ has a single click. Let $i$ be the clicked position for $Q$. Then $b_i$ is proportional to the ratio between the probability of $Q$ appearing in $\mathcal{S}$ and the probability of $Q$ in the uniformly sampled collection $\mathcal{U}$, i.e., $\hat{P}(Q)/P(Q)$, and thus

$$w_Q = \frac{P(Q)}{\hat{P}(Q)} \propto \frac{1}{b_i}$$

We now show some empirical data for the global bias model. We ran our randomization experiments for two document corpora (user emails and cloud storage files) and collected their data. We set $n = 4$, normalize all $b_i$ so that $\sum_i b_i = 1$, and plot the normalized $b_i$ values in Figure 2. As the results show, there are clearly position biases in both the email corpus and the cloud file storage corpus. For example, $b_1$ in the email corpus is about 0.40 but $b_4$ is about 0.15, confirming the strong bias of clicking top positions. More interestingly, Figure 2 also shows that the emails and the cloud storage files have very different bias values: the bias for files is much more flat than that of the emails.

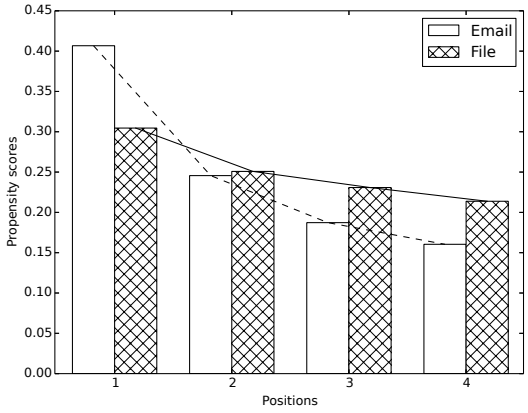


**Figure 2: The position bias propensity scores for user emails and cloud storage files.**

## 4.2 Segmented Bias Model

The global bias model quantifies the position bias solely based on the clicked position, which is a rather coarse esti-

mate. Most of past work does not go beyond this. However, motivated by the bias difference shown in Figure 2, it is possible that even within a single document corpus (e.g., email), different segments of queries have different position biases. We thus propose a more fine-grained model called the *segmented bias model*.

The basic idea of the segmented bias model is to partition queries into a few segments and then apply the global position bias model separately within each segment. Thus, we have a specific bias model for each segment. There may be multiple application-specific ways of segmenting queries such as using a query classifier. In our application, we focus on the email corpus from now on, and rely on the categories or labels assigned to each email. There are several such labels available in our corpus such as *Promotional* or *Social*. Each email can be associated with multiple labels[1]. There are multiple emails in the result list for each query and our goal is to select a single label and treat it as the segment for the query.

Inspired by the inverse document frequency (IDF) metric, we compute the inverse query frequency (IQF) for each label. For a label $t$,

$$IQF(t) \propto \frac{1}{|\{Q : t \in Q\}|}$$

where $t \in Q$ means that label $t$ is attached to some email retrieved for query $Q$ and $|\{Q : t \in Q\}|$ is the total number of queries that have label $t$ attached in the randomized data. Then the label $t(Q)$ of a query $Q$ is:

$$t(Q) = \arg\max_{t \in Q} \{IQF(t)\}$$

A by-product is that this creates segments with more balanced size. Given all the queries labelled by $t$, we can then estimate the position bias $b_i^t$ for this segment. Thus, for query $Q$, its inverse propensity weighting becomes:

$$w_Q \propto \frac{1}{b_i^{t(Q)}}$$

where $i$ is the clicked position of $Q$. Though simple, we find that this method is quite effective in our experiments.

## 4.3 Generalized Bias Model

The segmented model goes a step further to model the bias in a more fine-grained manner. A natural question is how to generalize this even further. For example, is it possible to have a query-dependent bias model? In other words, each query can potentially have different position biases. Due to the large number of unique queries, such a formulation seems intractable. However, as we show in this section, result randomization makes it possible to formulate a generalized *query-dependent* bias model.

Specifically, to estimate the position bias for query $Q$, suppose that we can present a randomly shuffled list of documents every time the query is issued and that user clicks are independent. A similar approach to the global bias model can be applied on the randomized data specifically for query $Q$. However, this is not practically feasible for the following reasons. First, to be able to accurately estimate the position bias for the query $Q$, hundreds or thousands of data points are needed. This means most queries will be filtered out because of data sparseness. Second, in the private search scenario, as discussed here, documents are unique to an individual user. Even for the same query string, the retrieved documents will differ across users. In order to collect sufficient data for a query, we need to show different randomized results for the same user from time to time. This will not only annoy the users, but also miss the purpose of data randomization since the data independence assumption will be violated. Thus, the challenge is how to tackle the following prediction problem.

**Definition 1 (Position Bias Prediction)** *Given a query* $Q = (q, \{\mathbf{x}_1, ..., \mathbf{x}_n\})$*, the problem of* Position Bias Prediction *is to estimate the click probability at each position* $i$ $(1 \le i \le n)$ *if we show the set of documents in a uniformly random order, specifically for* $Q$*.*

Recall that we have a set of queries in the randomized data. To solve the problem above, we propose a learning-based approach using multi-class logistic regression. We have $n$ positions and thus $n$ classes in the regression. For each query, we seek to estimate the probability of the query belonging to each class. Thus, we construct the training data from the randomized data and describe our approach as follows:

- **Labels**: For each query instance in the randomized data, we have the clicked position $i$. The label of this instance is class $i$. We use binary logistic regression as our algorithm and this query becomes a positive training example for class $i$ and a negative example for all other classes.
- **Features**: For each query $Q$, we can construct a feature vector $\mathbf{v}(Q)$. In our setting, a feature can be query-dependent or user-dependent. For a feature which depends on documents, the feature should only depend on the "set" of retrieved documents, without dependency on the actual order in the randomized data. For example, $t(Q)$ in the segmented bias model is such a feature that only depends on the "set" of the emails.
- **Training**: We train $n$ logistic regression models, each for a single position, based on the feature vectors and positive/negative training examples defined above. The logistic regression model for position $i$ is parameterized by a vector $\beta_i$. For a feature vector $\mathbf{v}(Q)$,

$$b_i^Q = \frac{1}{1 + \exp(\beta_i \cdot \mathbf{v}(Q))} \tag{4}$$

  The parameter $\beta_i$ can be obtained by maximizing the likelihood on the training data.
- **Prediction**: Given a query with its features, we can apply these $n$ models and obtain $n$ prediction values based on Eq 4. A value corresponding to position $i$ is the click probability when the results are shown uniformly randomly and thus the position bias.

We call the above the *generalized bias model*. Indeed, both global and segmented models are special cases in this generalized bias model.

**Proposition 1** *When the feature vector for each query has a single constant element* 1*, the generalized bias model is reduced to the global bias model.*

[1]The labeling algorithm itself is out of the scope of the paper, but the reader may refer to Grbovic et al. [16] or Bekkerman [2] for some prior work on the subject.

*Proof Sketch.* Let $c_i$ denote the total number of clicks on position $i$ and $C = \sum_i c_i$. When the feature vector for each query has a single constant element 1, $b_i^Q$ only depends on $i$ and the log likelihood of $b_i$ in Eq 4 is

$$c_i \log(b_i) + (C - c_i) \log(1 - b_i)$$

which is maximized when $b_i = c_i/C$. □

**Proposition 2** *The segmented bias model becomes a special case of the generalized bias model when we construct the feature vectors as follows. We create a binary feature for each segment, and a query has a value* 1 *in the feature corresponding to its segment and* 0 *elsewhere.*

*Proof Sketch.* The feature vector defined for the segmented model can partition the queries in the same way as the segmented model. The log likelihood of the whole data set can be separated into a few individual components, with each corresponding to a segment. Each component will be maximized similarly to Proposition 1 and thus the yielded biases are the same as the segmented bias model. □

As noted before, the generalized bias model is flexible to take any types of features such as query-specific or user-specific features. In our experiments we use a simple yet effective set of query length and query segment features described in Table 1. In Section 5, we will report some empirical results on the effectiveness of the generalized bias model based on these features.

# 5. EXPERIMENTS

In this section, we conduct experiments to compare different position bias prediction methods. All the methods that we compare are summarized in Table 2. In this table, NoCorrection means learning a scoring function without taking selection bias into account. It serves as the baseline to compare the other methods against. In the following, we first describe the experimental design and then present the experimental results.

## 5.1 Experimental Design

### 5.1.1 Data Sets

We use two data sets in this paper: regular click data and randomized click data.

- **Regular Data.** This is data collected from our click logs that is used for learning a scoring function. This data is made up of a random sample of email search logs from 2015-12-01 to 2015-12-07, resulting in 4 million queries with clicks. The training and test sets used in our offline evaluations are comprised of a 50/50 split of this data.

- **Randomized Data.** This is a randomized data set that is used to estimate bias in our proposed methods. To obtain randomized data, we randomly permuted the top search results returned for a a small fraction of email search queries (from 2015-11-18 to 2015-11-23), resulting in a total of 208K queries in total. To estimate position bias, we only retain the queries with exactly 4 results, which yields a data set with 148K queries.

| Feature Type | Description |
|---|---|
| Query length | Binary indicator based on the bucketized number of query characters: $[0, 10)$, $[10, 20)$, $[20, 30)$, $[30, \infty)$. |
| Segment | Binary indicator based on category segment $t(Q)$, as described in Section 4.2. |

**Table 1: Generalized bias model features.**

| Name | Method Description |
|---|---|
| NoCorrection | No bias correction is applied. This serves as our baseline. |
| Global | The bias is estimated for each position globally. |
| Segmented | The bias is estimated for each position per segment. |
| Generalized | The bias is estimated for each position per query using logistic regression. |

**Table 2: List of position bias prediction methods.**

### 5.1.2 Learning-to-Rank Algorithm

Our learning-to-rank algorithm is an adaptive one, in which we build a new model on top of the existing score. This is different from the standard approach, which learns a scoring function using the entire set of features. Instead, in the adaptive approach, we aim to train the adjustment $\delta(\mathbf{x})$ over the base score $s(\mathbf{x})$. The final scoring function becomes:

$$f(\mathbf{x}) = s(\mathbf{x}) + \delta(\mathbf{x})$$

We use the following ranking features to learn the adjustment $\delta(\mathbf{x})$:

- Email categories. This is the same set of categories used in the segmented bias model. An email can belong to multiple categories. For each category, we have a binary feature with 1 indicating that the email belongs to this category, and 0 otherwise.

- User interactions. We have a set of user interaction features logged for each email. For example, an interaction feature can be whether a user opened the email in the past or not.

This yields tens of ranking features. Although this may seem like a small number of features, the base score $s(\mathbf{x})$ is already highly optimized and includes hundreds of different features. The email category and user interaction features considered here add some additional information that is somewhat orthogonal to those used to compute the base score. For the NoCorrection baseline (see Table 2), we train $\delta(\mathbf{x})$ without applying any bias correction (i.e. $w_Q = 1$), and apply the respective selection bias weights during training for the Global, Segmented and Generalized models.

The additive nature in our adaptive model naturally fits the Multiple Additive Regression Trees (MART) learning algorithm [18]. In every iteration, MART trains a new tree to be added to the existing list of trees. In our setting, we start with our base score $s(\mathbf{x})$ and then train additive trees over it.

## 5.2 Experimental Results

In this section, we evaluate the position bias prediction models in a couple of different settings. Among them, the

| | Uniform | Global | Segmented | Generalized |
|---|---|---|---|---|
| Mean | 4.0 | 3.7360 | 3.7337 | **3.7336** |
| 95% CI | / | ±0.0202 | ±0.0201 | ±0.0197 |

**Table 3: Perplexity on the randomized data with 95% Confidence Interval.**

online experiments serve as the ultimate ground truth, but are expensive because they need to run against live traffic. We thus explore cheap offline evaluation methodologies and discuss their strengths and weaknesses.

### 5.2.1 Perplexity on Randomized Data

The position bias prediction problem can be treated as a standard prediction problem and thus can be evaluated using techniques like cross-validation. We split the randomized data into 10 folds and use the leave-one-out strategy to evaluate the different prediction methods. For each query, a prediction method gives a distribution over all the positions. We can thus use *perplexity* as the evaluation metric, which is defined as:

$$perplexity = 2^{-\frac{1}{N}\sum_{o=1}^{N}\log_2 p_o} \tag{5}$$

where $N$ is the total number of observations in the test data and $p_o$ is the probability of observation $o$ as predicted by the model to be evaluated. Perplexity measures how well a distribution predicts samples and is often used to evaluate or compare language models [3]. It is also used in recent work to compare click models [15]. In our case, each sample corresponds to a click at a position in the test data, and $p_o$ is the predicted bias probability for that sample. A lower perplexity score means the model is better at predicting the observations.

In Table 3, we show the perplexity score and the 95% confidence intervals based on the cross-validation. In this table, Uniform is a non-informative baseline and corresponds to the uniform prediction that gives a 25% for each of the 4 positions. From this table, we can see that all the position bias methods outperform Uniform significantly. For example, the 95% confidence interval of Global is $[3.7158, 3.7562]$. The Uniform perplexity 4.0 is outside of this range and thus the difference is significant. Comparing across all the bias prediction methods, we can see that Generalized achieves the best score and Segmented is very close to Generalized. However, the difference among all the 3 methods showed in the table are not significant.

Perplexity is an intrinsic measure of a given method's prediction accuracy. However, from the practical perspective we are much more interested in extrinsic evaluations of how models trained using each approach perform in terms of retrieval effectiveness. In the following, we examine different options for directly evaluating search quality.

### 5.2.2 Offline Evaluation on Regular Data

The obvious way of evaluating the ranking quality offline is to apply our learnt scoring function to a held-out test data set. However, as we will show, such a method is not well-grounded. Let us consider the evaluation metric Mean Reciprocal Rank (MRR), which is defined as follows:

$$\text{MRR} = \frac{1}{|\mathcal{S}|}\sum_{Q\in\mathcal{S}}\frac{1}{\text{rank}_Q} \tag{6}$$

| | Weighted MRR in Eq 7 | | |
|---|---|---|---|
| | Global $w_Q$ | Segmented $w_Q$ | Generalized $w_Q$ |
| NoCorrection | 1.0055 | 1.0055 | 1.0000 |
| Global | 1.0154 | - | - |
| Segmented | - | 1.0156 | - |
| Generalized | - | - | 1.0101 |
| Improvement | 0.9822% | 0.9968% | 1.0099% |

**Table 4: Offline evaluation on the regular data based on the weighted MRR. The number in this table is normalized by the smallest MRR.**

where $\text{rank}_Q$ is the rank of the first clicked document of query $Q$. In the context of selection bias, we need to incorporate the bias correction $w_Q$ into the metric as well and thus get the weighted MRR:

$$\text{MRR} = \frac{1}{\sum_{Q\in\mathcal{S}} w_Q}\sum_{Q\in\mathcal{S}} w_Q\frac{1}{\text{rank}_Q} \tag{7}$$

Different position bias prediction models give rise to different $w_Q$ values in Eq 7 and thus the MRR metric is not comparable across different $w_Q$ weights. This means that even for the same data set and the same scoring function, we may get different MRR values. To illustrate this, Table 4 reports the weighted MRR on the test data set of the regular data. We normalize the raw MRR values by the smallest value in the table. The row denoted by NoCorrection corresponds to the evaluation results on the same data set with the same scoring function. Clearly, the table shows different values when different $w_Q$ are used in the MRR metric. Though we can compute the relative improvement of different position bias correction models over the NoCorrection model, using the MRR with the corresponding $w_Q$, such improvement numbers are still not grounded as a reliable comparison metric. By changing the weight $w_Q$, one can artificially manipulate the improvement. For example, a high weight $w_Q$ can be given to queries where the new model has a higher MRR.

### 5.2.3 Unbiased Offline Evaluator

In this section, we address the challenge of offline evaluation by proposing a novel *unbiased* offline evaluator based on the randomized data. Such a method directly evaluates the ranking quality and thus is more practically useful than perplexity. Furthermore, it is theoretically sound, unbiased, and can overcome the comparability issues that arise when using the regular data for offline evaluations.

Unbiased offline evaluators have been studied extensively in the setting of contextual-bandit problems [12, 25]. We adapt this strategy to our problem setting. Our proposed algorithm is detailed in Algorithm 1. This algorithm goes through every query in the randomized data set $\mathcal{R}$ and selects a matched subset $\mathcal{R}_s$ based on the provided scoring function $f(\mathbf{x})$. The matching condition is that the ranking recorded in our log data $\mathcal{R}$ is the same as that ranked by $f(\mathbf{x})$ for the top $k$ documents. The metric value is then computed on the selected subset $\mathcal{R}_s$.

**Theorem 1** *Given uniformly randomized data $\mathcal{R}$, Algorithm 1 gives an unbiased estimate of any metric $M$ for any scoring function $f(\mathbf{x})$.*

*Proof*: This is a simplified version of Theorem 1 in [25] in that we have a static scoring function. We only prove the

**Algorithm 1** Offline Evaluator

Input: scoring function $f$; randomized data $\mathcal{R}$; evaluation metric $M$ on top $k$: $M_k$.
Output: evaluation value of $f$.

```
Set matched data collection R_s := ∅
for Q = (q, ⟨x_1, ..., x_n⟩) in R do
    Let ⟨x_{j_1}, ..., x_{j_n}⟩ be ⟨x_1, ..., x_n⟩ re-ranked by f
    if ⟨j_1, ..., j_k⟩ = ⟨1, ..., k⟩ then
        R_s := R_s ∪ Q
    end if
end for
return M_k(R_s)
```

case when $k = n$. Our goal is to show that $\mathcal{R}_s$ is an unbiased sample of events if we use $f(\mathbf{x})$ as the scoring function. Since all the rankings in $\mathcal{R}_s$ are the same as ranked by $f(\mathbf{x})$, we only need to prove that the marginal probability on the query string itself, $P(q)$, in $\mathcal{R}_s$ is unbiased.

$$P(q) = \sum_{\{Q:Q\in\mathcal{R}_s \text{ and } q\in Q\}} P(Q)$$

where $q \in Q$ means $q$ is the query string of $Q$. This is the case because $P(q)$ in $\mathcal{R}$ is unbiased since we collect all the data and the probability of entering the if condition in Algorithm 1 is $\frac{1}{n!}$ for all the queries. □

One caveat of the above algorithm is that we only use queries with exactly $n$ results. For queries with $j < n$ results, we can weigh them by $\frac{j!}{n!}$ when estimating the metric.

The full procedure for evaluating a position bias prediction method is as follows:

- Split the randomized data into training and test (e.g., via a 50/50 split).
- Train the bias prediction model using the randomized training data.
- Apply the learnt position bias model to the regular training data to obtain a scoring function.
- Evaluate the scoring function on the randomized test data based on Algorithm 1.

Using NoCorrection as the baseline, we report the relative improvement of different bias prediction models in Table 5. In this table, we show the MRR in Eq 6 evaluated on the top $k$ results along with the size of $\mathcal{R}_s$. The results show that all position bias prediction methods outperform the NoCorrection baseline. For the position bias prediction methods, both Segmented and Generalized are better than Global, and Segmented is slightly better than Generalized, demonstrating the potential utility of advanced position bias models. However, the differences are not statistically significant, due to the high variance incurred by the small evaluation data set. For $k$, the expected size of $\mathcal{R}_s$ is about $\frac{(n-k)!}{n!}$ of the size of $\mathcal{R}$. A larger data set could be used to increase the statistical power of our proposed methods.

### 5.2.4 Online Experiments on Live Traffic

Online experiments are the ultimate litmus tests to evaluate different scoring functions. Our online experiments are in the form of A/B testing: we allocate a fraction of live traffic for each experiment. One half of the fraction is used as *control* (i.e., the NoCorrection model, as described in Section 5.1.2) and the other half is used as *treatment* (i.e., the one of the bias prediction methods). We create an online experiment for each of the methods listed in Table 2. We ran all four online experiments for a period of one week, collecting millions of clicks per experiment. Based on the click data, we compare the treatment and the control by computing the relative improvement in terms of our evaluation metric MRR (mean reciprocal rank of a click) in Eq 6. We also compute the relative improvement between two treatment methods using NoCorrection as the calibrator.

| $k$ | $\lvert\mathcal{R}_s\rvert$ | Global | Segmented | Generalized |
|---|---|---|---|---|
| 1 | 19.8K | 0.94% | **1.01%** | 0.97% |
| 2 | 6.7K | 1.08% | **1.28%** | 1.20% |
| 3 | 3.3K | 1.58% | 1.67% | **1.68%** |
| 4 | 3.3K | 1.37% | **1.44%** | 1.41% |

**Table 5: Comparison of different position bias prediction methods using the unbiased offline evaluator. We report the relative improvement over the NoCorrection baseline.**

| MRR | | | |
|---|---|---|---|
| Baseline | Global | Segmented | Generalized |
| NoCorrection | 0.67%*** | 0.88%*** | 0.79%*** |
| Global | - | 0.21%* | 0.12% |
| **CTR** | | | |
| Baseline | Global | Segmented | Generalized |
| NoCorrection | 0.46%*** | 0.71%*** | 0.62%*** |
| Global | - | 0.25%** | 0.15% |

**Table 6: Comparison of different bias prediction methods using online experiments. We report the relative improvement over the NoCorrection baseline. Notation *, ** and *** means the difference is significant at level 0.1, 0.05 and 0.01 respectively.**

Table 6 summarizes the results of our online experiments. In this table, we report the relative improvement in MRR compared with the NoCorrection and Global baselines. For the NoCorrection baseline, we can see that all our position bias methods yield statistically significant improvements at the 0.01 level. This confirms that selection bias in click data is significant and overcoming it can lead to significant quality improvement. From Global baseline, we can see that more fine-grained position bias models are capable of further improving our metric. For example, Segmented outperforms Global significantly at 0.1 level.

For the online experiments, we can also report the click-through rate (CTR). The CTR metric reflects how attractive the result section is as a whole. We also report the relative improvement in terms of CTR in Table 6. Our observations for the CTR metric parallel those for the MRR metric – the Global baseline significantly increases CTR, and Segmented and Generalized methods provide further improvements. Furthermore, Segmented model achieves significant improvement over Global at 0.05 level.

The results of the online experiments do not indicate a significant difference between Segmented and Generalized models, even though both outperform the Global baseline. As we showed before, Segmented is a special case of General-

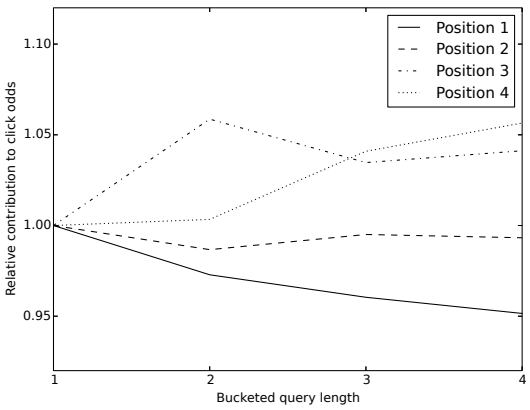


**Figure 3: The importance of query length varied with positions.**

ized when only the segmented features are used, suggesting the usefulness of the category segment feature, and the importance of feature engineering for further improvement of the generalized bias model.

### 5.2.5 Regression Models Analysis

The generalized bias model not only provides a flexible way for position bias prediction, but also enables us to understand the impact of different features in terms of their usefulness in modeling position bias. In this section, we analyze the features to distill additional insights.

We have two types of features in the regression models: segment features and query length features. The question is which group of features is more predictive. To answer this question, we compare 3 generalized models with different sets of features and observe the following changes in perplexity (as defined in Eq 5): segment and query length features (3.7336), segment features only (3.7337), and query length features only (3.7358). Clearly we can see that segment features are more useful than query length features in reducing the perplexity.

Furthermore, we can observe the impact of different lengths of query features. Our query lengths are bucketed as in Table 1 with larger bucket IDs corresponding to longer queries. For each position $i$ and each query length bucket $j$, we have $\beta_{ij}$ corresponding to the coefficient in the logistic regression model. Here, $e^{\beta_{ij}}$ represents the contribution of query length $j$ to the odd of click on position $i$: $b_i/(1-b_i)$. We plot the relative contribution $e^{\beta_{ij}}/e^{\beta_{i1}}$ in Figure 3. For position $i = 1$, $e^{\beta_{ij}}/e^{\beta_{i1}}$ becomes smaller when $j$ becomes larger. In other words, the odds of a click at position 1 *decrease* when the query is longer. In contrast, for position $i = 4$, $e^{\beta_{ij}}/e^{\beta_{i1}}$ becomes larger when $j$ becomes larger, which means the odds of a click at position 4 *increase* as the query becomes longer. This makes sense intuitively, since when queries are longer, users have more refined needs and the position bias becomes flatter. This means that the users are more willing to examine the lower-ranked documents.

## 6. CONCLUSIONS

In this paper, we studied the problem of learning-to-rank with selection bias for personal search. We discussed the infeasibility of using existing click models in personal search and proposed a novel approach to overcome the inherent selection bias for this application. We proposed a few methods to estimate the selection bias and addressed it using inverse propensity weighting. In addition, we study offline and online evaluation methodologies and also propose a novel unbiased offline evaluator. Through extensive offline and online experiments, we show that the proposed methods for modeling selection bias can significantly improve the quality of learning-to-rank models that use click data for training.

There are a few interesting lines of future work. (1) We evaluate our methods in the context of personal search, but it would be interesting to see how applicable they are to web search. (2) Our experiments use queries with a single click. It would be interesting to extend the framework to the search scenarios that allow multiple clicks per query. (3) Given a different application, such as cloud storage files, what are the effective features in bias estimation? This could inspire lots of interesting feature engineering work in the research community. (4) The expensive part of our method is the dependency on randomized data. How to collect randomized data in a cheaper, less-intrusive manner is also worth studying. Furthermore, how to adapt our offline evaluator to improve its data utilization is also an interesting research problem.

## 7. REFERENCES

[1] J. A. Aslam, E. Kanoulas, V. Pavlu, S. Savev, and E. Yilmaz. Document selection methodologies for efficient and effective learning-to-rank. In *32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 468–475, 2009.

[2] R. Bekkerman. Automatic categorization of email into folders: Benchmark experiments on Enron and SRI corpora. Technical report, University of Massachusetts Amherst, 2004.

[3] P. F. Brown, V. J. D. Pietra, R. L. Mercer, S. A. D. Pietra, and J. C. Lai. An estimate of an upper bound for the entropy of English. *Computational Linguistics*, 18(1):31–40, 1992.

[4] C. J. Burges. From RankNet to LambdaRank to LambdaMART: An overview. Technical Report MSR-TR-2010-82, Microsoft Research, 2010.

[5] S. Büttcher, C. L. A. Clarke, P. C. K. Yeung, and I. Soboroff. Reliable information retrieval evaluation with incomplete and biased judgements. In *30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 63–70, 2007.

[6] D. Carmel, G. Halawi, L. Lewin-Eytan, Y. Maarek, and A. Raviv. Rank by time or by relevance?: Revisiting email search. In *24th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 283–292, 2015.

[7] D. Chan, R. Ge, O. Gershony, T. Hesterberg, and D. Lambert. Evaluating online ad campaigns in a pipeline: Causal models at scale. In *16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 7–16, 2010.

[8] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *18th*

*International Conference on World Wide Web (WWW)*, pages 1–10, 2009.

[9] W. Chen, Z. Ji, S. Shen, and Q. Yang. A whole page click model to better interpret search engine click data. In *25th AAAI Conference on Artificial Intelligence (AAAI)*, 2011.

[10] A. Chuklin, I. Markov, and M. de Rijke. *Click Models for Web Search*. Morgan & Claypool, 2015.

[11] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *1st International Conference on Web Search and Data Mining (WSDM)*, pages 87–94, 2008.

[12] M. Dudík, J. Langford, and L. Li. Doubly robust policy evaluation and learning. In *28th International Conference on Machine Learning (ICML)*, pages 1097–1104, 2011.

[13] S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff I've seen: A system for personal information retrieval and re-use. In *26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 72–79, 2003.

[14] G. Dupret and C. Liao. A model to estimate intrinsic document relevance from the clickthrough logs of a web search engine. In *3rd ACM International Conference on Web Search and Data Mining (WSDM)*, pages 181–190, 2010.

[15] G. E. Dupret and B. Piwowarski. A user browsing model to predict search engine click data from past observations. In *31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 331–338, 2008.

[16] M. Grbovic, G. Halawi, Z. Karnin, and Y. Maarek. How many folders do you really need?: Classifying email into a handful of categories. In *23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM)*, pages 869–878, 2014.

[17] F. Guo, C. Liu, A. Kannan, T. Minka, M. Taylor, Y.-M. Wang, and C. Faloutsos. Click chain model in web search. In *18th International Conference on World Wide Web (WWW)*, pages 11–20, 2009.

[18] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[19] K. Hofmann, S. Whiteson, and M. de Rijke. Balancing exploration and exploitation in learning to rank online. In *Advances in Information Retrieval – 33rd European Conference on IR Research (ECIR)*, pages 251–263, 2011.

[20] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola. Correcting sample selection bias by unlabeled data. In *20th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 601–608, 2006.

[21] T. Joachims. Optimizing search engines using clickthrough data. In *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142, 2002.

[22] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 154–161, 2005.

[23] M. Kamvar, M. Kellar, R. Patel, and Y. Xu. Computers and iPhones and mobile phones, oh my!: A logs-based comparison of search users on different devices. In *18th International Conference on World Wide Web (WWW)*, pages 801–810, 2009.

[24] L. Li, S. Chen, J. Kleban, and A. Gupta. Counterfactual estimation and optimization of click metrics in search engines: A case study. In *24th International Conference on World Wide Web (WWW) Companion*, pages 929–934, 2015.

[25] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *4th International Conference on Web Search and Web Data Mining (WSDM)*, pages 297–306, 2011.

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

[27] T. Minka and S. Robertson. Selection bias in the LETOR datasets. In *SIGIR Workshop on Learning to Rank for Information Retrieval (LR4IR)*, pages 48–51, 2008.

[28] M. O'Brien and M. T. Keane. Modeling result–list searching in the world wide web: The role of relevance topologies and trust bias. In *28th Annual Conference of the Cognitive Science Society (CogSci)*, pages 1–881, 2006.

[29] T. Qin, T.-Y. Liu, J. Xu, and H. Li. Letor: A benchmark collection for research on learning to rank for information retrieval. *Inf. Retr.*, 13(4):346–374, 2010.

[30] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: Estimating the click-through rate for new ads. In *16th International Conference on World Wide Web (WWW)*, pages 521–530, 2007.

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

[32] A. Swaminathan and T. Joachims. Counterfactual risk minimization: Learning from logged bandit feedback. In *32nd International Conference on Machine Learning (ICML)*, pages 814–823, 2015.

[33] Y. Yue, R. Patel, and H. Roehrig. Beyond position bias: Examining result attractiveness as a source of presentation bias in clickthrough data. In *19th International Conference on World Wide Web (WWW)*, pages 1011–1018, 2010.

[34] B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *21st International Conference on Machine Learning (ICML)*, page 114, 2004.

[35] Z. Zheng, K. Chen, G. Sun, and H. Zha. A regression framework for learning ranking functions using relative relevance judgments. In *30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 287–294, 2007.

[36] Z. A. Zhu, W. Chen, T. Minka, C. Zhu, and Z. Chen. A novel click model and its applications to online advertising. In *3rd International Conference on Web Search and Data Mining (WSDM)*, pages 321–330, 2010.