Position Bias Estimation for Unbiased Learning to Rank in Personal Search

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

ABSTRACT

A well-known challenge in learning from click data is its inherent bias and most notably position bias. Traditional click models aim to extract the ⟨query, document⟩ relevance and the estimated bias is usually discarded after relevance is extracted. In contrast, the most recent work on unbiased learning-to-rank can effectively leverage the bias and thus focuses on estimating bias rather than relevance [20, 31]. Existing approaches use search result randomization over a small percentage of production traffic to estimate the position bias. This is not desired because result randomization can negatively impact users' search experience. In this paper, we compare different schemes for result randomization (i.e., *RandTopN* and *RandPair*) and show their negative effect in personal search. Then we study how to infer such bias from regular click data without relying on randomization. We propose a regression-based Expectation-Maximization (EM) algorithm that is based on a position bias click model and that can handle highly sparse clicks in personal search. We evaluate our EM algorithm and the extracted bias in the learning-to-rank setting. Our results show that it is promising to extract position bias from regular clicks without result randomization. The extracted bias can improve the learning-to-rank algorithms significantly. In addition, we compare the pointwise and pairwise learning-to-rank models. Our results show that pairwise models are more effective in leveraging the estimated bias.

CCS CONCEPTS

• **Information systems → Learning to rank**;

KEYWORDS

Position bias estimation; inverse propensity weighting; expectation-maximization

ACM Reference Format:
Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, Marc Najork. 2018. Position Bias Estimation for Unbiased Learning to Rank in Personal Search. In *WSDM 2018: WSDM 2018: The Eleventh ACM International Conference on Web Search and Data Mining , February 5–9, 2018, Marina Del Rey, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3159652.3159732

WSDM 2018, February 5–9, 2018, Marina Del Rey, CA, USA

ACM ISBN 978-1-4503-5581-0/18/02.
https://doi.org/10.1145/3159652.3159732

1 INTRODUCTION

Personal search is an important research topic in information retrieval that has been evolving from desktop search [13] to on-device search [21] and more recently to email search [5]. With the increasing popularity of cloud-based email and storage services, there is a large amount of personal data stored in the cloud. This large amount of data calls for effective search capabilities to improve its utility [2, 5, 6, 15]. The important difference of personal search from web search is the private content in personal search. Users can only see and search their own content. This presents an important challenge when applying learning-to-rank techniques, because collecting explicit relevance judgements becomes much harder and raters can only label their own documents with their own queries. Though possible, such an approach can be heavily biased by the selected raters and costly to maintain due to the fast-evolving nature of private content.

Click data in personal search provides implicit but abundant user feedback. It thus becomes a natural source to improve personal search quality. However, a well-known challenge in learning from click data is its inherent bias: position bias [19], presentation bias [32], and trust bias [26], etc. Among them, the position bias has a strong influence on users' clicks [20]. A prerequisite of fully leveraging the power of click data is to debias it. As a result, there has been a great deal of research on extracting reliable signals from click data.

One such effort is known as click modeling. Click models were initially proposed for web search with the goal of estimating ⟨query, document⟩ relevance by taking the click bias into account [9]. The position bias model [27] and the Cascade model [10] are the two classic click models. In the position bias model, users' clicks are determined by two factors: document positions, and the ⟨query, document⟩ relevance. The Cascade model assumes sequential user behavior: a user scans the documents one by one from the top, and the scan continues after a non-relevant document but stops after a relevant document. More sophisticated models (e.g., UBM [14], DBN [7], and CCM [17]) build upon these two classic models. In the learning-to-rank setting, the existing click models belong to the pointwise category because they estimate ⟨query, document⟩ relevance but not based on the correct *order* of documents as in pairwise and listwise approaches.

Different from click models, Joachims et al. [19] studied a few heuristics in the pairwise manner (e.g., *SkipAbove*) and found that these heuristics lead to more reliable relevance assessments. However, the data obtained by these heuristics are systematically biased and a ranking function learned from it tends to reverse the existing order [20].

Recently, Wang et al. [31] and Joachims et al. [20] introduced the unbiased learning-to-rank framework by treating the bias as a counterfactual effect. Inside the framework, the critical component is to quantify the click bias, rather than estimate the ⟨query, document⟩ relevance in click models. Due to this difference, randomizing results on a small fraction of search traffic is sufficient.

In this paper, we study different ways of position bias estimation for unbiased learning-to-rank. Result randomization intuitively degrades the users' search experience. We quantify the negative impacts of different degrees of randomization (*RandTopN* and *RandPair*) and compare their effectiveness for bias estimation. This naturally motivates us to study how to estimate the bias from regular clicks. The classical position bias model is a candidate method. A direct application of this model requires the same ⟨query, document⟩ pairs appearing at multiple positions multiple times [7], but this is not realistic in personal search. We propose a novel regression-based EM algorithm that does not need the query and document identifiers but works in a feature space. Such an approach is applicable to any ranking system where clicks and ranking features are collected. It also greatly reduces the pre-processing that is usually needed in standard click models. Furthermore, as position bias can potentially change as a ranking system evolves, our approach can be used to re-estimate position bias without any intervention like result randomization.

The contribution of this paper is summarized as follows:

- We study the problem of position bias estimation without result randomization and propose a regression-based EM [11] in personal search that can be applied to any ranking system without the need of intervention.
- We compare different degrees of result randomization based on effectiveness of bias estimation and negative impacts on an email and a file storage personal search service.
- We conduct extensive experiments and our results show that position bias estimated using EM can achieve surprisingly good results without using randomization, and that pairwise learning-to-rank algorithms can better leverage these bias estimates than pointwise ones.

The rest of this paper is organized as follows. In Section 2, we review previous related work. The unbiased learning-to-rank is set up in Section 3 and is followed by different methods of propensity estimation based on the position bias model 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

Click-through data has become an indispensable resource in both web search and personal search. There is an abundance of prior work on click modeling in web search that address click bias to extract useful signals from click data. One of the seminal papers by Joachims et al. [19] evaluates a few heuristics via a user study and shows that clicks are valuable as long as certain biases, especially the position bias, are accounted for. Many click models have been proposed to model the user click behaviors based on probabilistic graphical models [7–10, 14, 17, 27, 34]. These models can account for the click bias and provide more reliable relevance estimation between queries and documents.

Unbiased learning-to-rank takes a different approach to account for click bias. Wang et al. [31] study the selection bias at the query level when click-based training data is collected in learning-to-rank. Joachims et al. [20] present a counterfactual inference framework with a principled theoretical basis. Both approaches rely on Inverse Propensity Weighting (IPW) developed in the causal inference field [28], and both rely on result randomization to estimate the propensity. Our paper is in the same unbiased learning-to-rank framework; however, we estimate the propensity without randomization.

Our work is related to recent work on personal search [2, 3, 6, 22, 33] that emphasized relevance-based over time-based ranking [6]. User behaviors in personal search show different traits compared with web search [2]. For example, repeated visits to the same results are less common in email search. Thus, applying learning-to-rank techniques is important; however, the click data is highly sparse in personal search. We use a regression-based EM algorithm that extends the standard EM algorithm to handle the highly sparse data effectively.

Inverse propensity weighting is a commonly-used technique to address the sample bias. It has been adopted for unbiased evaluation and learning (e.g., [1, 12, 23, 24, 29, 30]). Most work in this area assumes that the propensity scores are present in the logs and study how to reduce the model variance while staying unbiased. The unbiased learning-to-rank framework is different in that the propensity is not explicitly logged but buried in the user behavior data implicitly. How to estimate this propensity in a less costly manner is critical in unbiased learning-to-rank and the focus of this paper.

3 UNBIASED LEARNING-TO-RANK

For simplicity, we assume that a clicked document is relevant. Note that such an assumption can be relaxed to accommodate noisy clicks in unbiased learning-to-rank as shown in [20]. However, the reverse is not true due to the position bias: relevant documents may not get clicked all the time because the users do not go down the list to examine it. Thus, click data obtained from search logs is biased due to the unobserved feedback. Learning from it without accounting for the bias leads to less effective ranking functions. In this section, we review and discuss different ways of unbiased learning-to-rank from click data.

3.1 Inverse Propensity Weighting

In the learning-to-rank setting, there are two existing propensity-based bias correction methods. In the following, we use a Bernoulli variable O to denote whether the relevance of a document is observed. The variable O can depend on various factors such as positions or even an entire rank list. Without loss of generality, for i-th document, we use a vague notation $P(O_i = 1)$ to denote the propensity and $o_i \in \{0, 1\}$ to denote a specific value. We use $r_i \in \{0, 1\}$ to denote whether a document is relevant and use π to denote the list $[1, \cdots, n]$ of ranks of n results.

Query-level propensity. A key observation in [31] is that queries without clicks are not useful in the pairwise or listwise learning-to-rank approaches and not included in the training data. Hence, due to the position bias, queries whose relevant documents are ranked lower can be under-represented in the training data. The propensity that a query is selected in the training data depends on the highest position of its relevant documents:

$$P(O_{i_q} = 1) \text{ where } i_q = \min\{i \in \pi : r_i = 1\}.$$

The bias is corrected by using a query-level Inverse Propensity Weighting (IPW)

$$w_q = \frac{1}{P(O_{i_q} = 1)}.$$

Such an approach fits perfectly well when there is a single click per query but can be less accurate for queries with multiple relevant documents.

Document-level propensity. Document-level propensity is studied in [20]. It starts with a performance metric that measures the rank of the relevant documents (denoted as *Rank*):

$$Rank = \sum_{i \in \pi} r_i \cdot i.$$

In biased data, IPW is used to correct the bias:

$$\overline{Rank} = \sum_{i \in \pi : o_i = 1} \frac{r_i \cdot i}{P(O_i = 1)} = \sum_{i \in \pi : o_i = 1, r_i = 1} \frac{i}{P(O_i = 1)}.$$

The *Rank* metric is proven unbiased, i.e., $\mathbb{E}\{\overline{Rank}\} = Rank$ [20]. The document-level IPW is

$$w_i = \frac{1}{P(O_i = 1)}.$$

3.2 Unbiased Learning Algorithms

The *Rank* metric can be optimized by a pairwise learning-to-rank method such as SVMRank. To show this, let s_i and s_j be the scores of a ranking function for document i and j. *Rank* can be expressed as a pairwise loss using the indicator function $\mathbb{I}$:

$$\begin{aligned}\sum_{i \in \pi : o_i = 1, r_i = 1} w_i \cdot i &= \sum_{i \in \pi : o_i = 1, r_i = 1} w_i \cdot (\sum_{j \in \pi} \mathbb{I}_{s_j > s_i} + 1) \\ &= \sum_{i \in \pi : o_i = 1, r_i = 1} \sum_{j \in \pi} w_i \cdot \mathbb{I}_{s_j > s_i} + const\end{aligned}$$

The loss above is upper-bounded by the common classification loss functions (e.g., the hinge loss or the log loss [18]) and can be optimized by pairwise learning-to-rank algorithms by constructing the following preference pairs: For each **observed** and **relevant** document i, we pair it with each of the other documents j in the given query and create a preference pair $i \rhd j$. Note that the weight for each pair is only determined by w_i, the IPW of document i.

Given a single-click training data, both query-level propensity and document-level propensity lead to the same pairwise learning algorithm. For multi-click training data, the query-level propensity can be adapted by replicating a multi-click query to multiple single-click queries with each corresponding to a click. It is easy to show that such a strategy of the query-level IPW leads to the same pairwise learning algorithm as the document-level IPW.

In addition, both *Rank* and $\overline{Rank}$ can be optimized using the λ-gradient in LambdaRank or LambdaMART [4]. Similar to *Rank*, we can define the DCG-like metric *Prec* and its IPW version $\overline{Prec}$ as follows:

$$\begin{aligned}Prec &= \sum_{i \in \pi} \frac{r_i}{i} = \sum_{i \in \pi : r_i = 1} \frac{1}{i} \\ \overline{Prec} &= \sum_{i \in \pi : o_i = 1} w_i \frac{r_i}{i} = \sum_{i \in \pi : o_i = 1, r_i = 1} w_i \frac{1}{i}\end{aligned}$$

$\overline{Prec}$ is also provenly unbiased, i.e., $\mathbb{E}\{\overline{Prec}\} = Prec$. We use *Prec* in our paper as the optimization metric and other types of DCG-like metrics can be used easily in the current setting.

In the unbiased learning-to-rank framework, the critical part is to estimate the propensity $P(O_i = 1)$. Note that a nice property of the current metrics is that we only need the propensity of **observed** and **relevant** documents, i.e., $o_i = 1$ and $r_i = 1$, and we do not need the propensity of non-relevant documents. Such a nice property holds for pairwise learning algorithms because they are closely related to the metrics (e.g., *Rank* and *Prec*) in which non-relevant documents have no contributions to the metric values. However, it does not hold for pointwise learning algorithms because a non-relevant document usually contributes to the pointwise loss functions. We will show some attempts in our experiments; however, how to extend the IPW method to pointwise approaches is an open area of research.

4 PROPENSITY ESTIMATION

In this section, we show how to estimate the propensity from the click data based on a position bias model. The position bias model is a simple yet effective generative click model. We first set up the model and then present different methods for parameter estimation, including a novel regression-based EM algorithm.

4.1 Position Bias Model

The position bias model assumes that the observed click Bernoulli variable C depends on two other hidden Bernoulli variables E and R where E represents the event whether a user examines a document at a certain position k and R represents the event whether a document d is relevant to a query q. Specifically,

$$P(C = 1|q, d, k) = P(E = 1|k) \cdot P(R = 1|q, d),$$

where $P(C = 1|q, d, k)$ is the probability of clicking document d that is shown at position k given query q, $P(E = 1|k)$ is the probability that position k is examined, and $P(R = 1|q, d)$ is the probability that document d is relevant to query q. The model assumes that the examination only depends on the position and the relevance only depends on the query and document. We use the following shorthands for derivation:

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

Though simple, this model has been shown to be as effective as more sophisticated click models [9]. In particular, when there is a single click per query, this model is equivalent to the User Browsing Model (UBM) [14] that achieves the state-of-the-art performance. On the other hand, more sophisticated click models can be potentially used as well for propensity estimation.

Theorem 4.1. *In the position bias model, the* **observed** *variable O is equivalent to the* **examination** *variable E: $O \Leftrightarrow E$.*

This has been proven in [20]. When $E = 1$, the relevance R is fully observed by C and thus $O = 1$. When $E = 0$, $C = 0$ always holds and the document may or may not be relevant. In another word, we have no knowledge of R and thus $O = 0$.

With Theorem 4.1, estimating propensity $P(O = 1)$ equals to estimating the examination probability $P(E = 1)$. Also, a clicked document i means it is examined and relevant. In other words, it is relevant ($r_i = 1$) and observed ($o_i = 1$) and thus we only need to estimate and apply the propensity scores on the clicked documents.

4.2 Result Randomization

To estimate θ_k, we need a way to estimate $\gamma_{q,d}$ or eliminate it from the equation. Result randomization is a way to integrate it out. We describe two methods here.

4.2.1 Randomize TopN. In the position bias model, the relevance component $\gamma_{q,d}$ is hidden. Result randomization can be leveraged to estimate θ_k without explicitly modeling the relevance component. For a randomized data set $\mathcal{R}$ where the top N documents are randomly shuffled before showing them to users, let $\mathcal{R}_k$ be the subset of the collected logs from position k. We have

$$\begin{aligned}\mathbb{E}(C|k) &= \int_{q,d\in\mathcal{R}_k} \mathbb{E}(C|q,d,k)P(q,d) \\ &= \int_{q,d\in\mathcal{R}_k} P(C=1|q,d,k)P(q,d) \\ &= \int_{q,d\in\mathcal{R}_k} \theta_k\gamma_{q,d}P(q,d) \\ &= \theta_k \cdot \int_{q,d\in\mathcal{R}_k} \gamma_{q,d}P(q,d) \\ &\propto \theta_k\end{aligned}$$

This is the case because $\int_{q,d\in\mathcal{R}_k} \gamma_{q,d}P(q,d) = \mathbb{E}\{\gamma_{Q,D}\}$ is constant across all the positions in the randomized data $\mathcal{R}$. Thus, the position bias θ_k is proportional to the number of clicks in $\mathcal{R}_k$. We use *RandTopN* to denote this method.

4.2.2 Randomize Pair. Shuffling a few top results randomly can lower the search quality a lot and thus create undesired user experience. A relatively smaller intervention is to randomize pairs. In [20], results at rank 1 and rank k are randomly swapped half the time. In this paper, we use a simpler variant that randomly swaps adjacent pairs at position $k-1$ and k. We vary k and collect search logs for each k separately. Similarly to *RandTopN*, we can get a relative ratio $\frac{\theta_k}{\theta_{k-1}}$ for adjacent pairs based on the number of clicks at position $k-1$ and k when the results of these two positions are randomly swapped. In the position bias model, we can obtain the relative ratio between any two positions by multiplying the ratios of adjacent positions.

$$\frac{\theta_k}{\theta_1} = \frac{\theta_2}{\theta_1}\cdot\frac{\theta_3}{\theta_2}\cdots\frac{\theta_k}{\theta_{k-1}}.$$

We name this scheme *RandPair*. It is different from *RandTopN* in that the expected relevance $\mathbb{E}\{\gamma_{Q,D}\}$ for a position pair is different from another position pair. But under the assumption in the position bias model, the relative ratios between different positions should be the same as *RandTopN*, which also applies to the one used in [20] similarly.

	Email (N=3)	File Storage (N=5)
RandTopN	-13.94%*	-31.04%*
RandPair(1, 2)	-6.80%*	-12.44%*
RandPair(2, 3)	-0.56%	+3.75%
RandPair(3, 4)	+0.20%	+1.09%
RandPair(4, 5)	+0.38%	+0.36%

Table 1: The negative effect due to result randomization measured by the relative change of MRR against the production. * means statistically different.

4.2.3 Empirical Comparison. The position bias model is a simplification of real-world user click behavior and can be less precise. We empirically compare *RandTopN* and *RandPair* here to see how well they align. We use a portion of traffic for randomization on two search services: one is email search and the other is file storage search, and run multiple randomization experiments on each service. Figure 1 plots the θ_k that we got from *RandTopN* and *RandPair* experiments. For *RandTopN*, we set $N = 3$ for email search and $N = 5$ for file storage search. From this figure, we can see that *RandPair* aligns very well with *RandTopN* and this confirms that the assumption of the position bias model is reasonable.

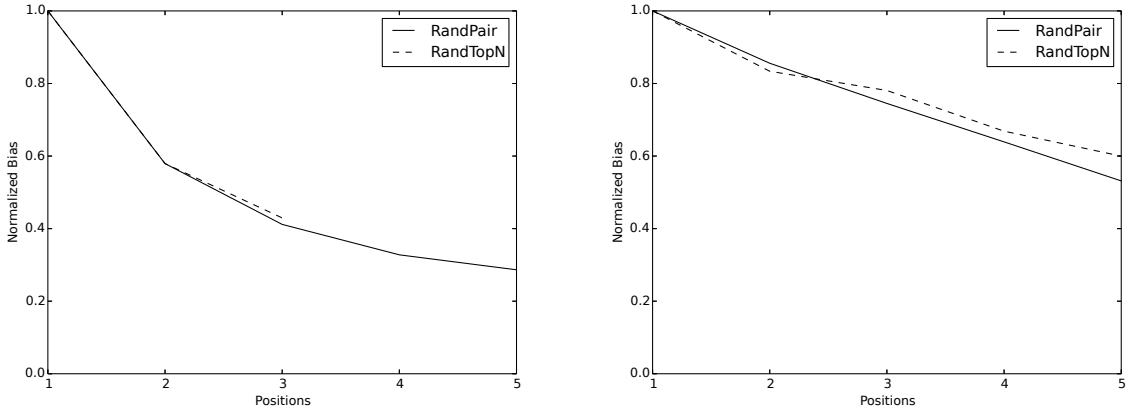
One of the drawbacks of result randomization is its negative effect on the search experience. Reusing the randomization experiments, we quantify the impact on Mean Reciprocal Rank (MRR) of the clicked positions and compare result-randomized traffic with production traffic. Table 1 summarizes the comparison using the relative change of MRR. From this table, we can see that the search experience is dramatically decreased due to randomization on both services. *RandTopN* is worse than *RandPair* since it perturbs results more aggressively. For *RandPair*, the impact generally becomes smaller as the positions become lower. In the table, the impact becomes neutral after position 2. The fluctuation at the lower positions is mainly due to the larger variance of user buckets where fewer clicks are collected at those positions. Compared with [20], our *RandPair* is less aggressive in perturbing results per search, but may need to be run longer to get sufficient clicks for bias estimation since pairs in our method are at less visible positions. An interesting comparison is on how much traffic is necessary for bias estimation in different methods, which we leave as future work.

4.3 Estimation from Regular Clicks

Can we estimate the position bias from regular production clicks without randomization and how accurate would that be? To the best of our knowledge, there is no existing study on such a comparison. In this section, we give details of the standard EM algorithm and present a regression-based EM for personal search.

Given a regular click log $\mathcal{L} = \{(c,q,d,k)\}$, the log likelihood of generating this data is

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



(a) Email. **(b) File Storage.**

Figure 1: Position bias estimated by result randomization and normalized by the top position.

The EM algorithm can find the parameters that maximize the log likelihood of the whole data.

4.3.1 Standard EM. The standard EM algorithm iterates over the Expectation and Maximization steps to update parameters $\{\theta_k\}$ and $\{\gamma_{q,d}\}$. At iteration $t+1$, the Expectation step estimates the distribution of hidden variable E and R given parameters from iteration t: $\{\theta_k^{(t)}\}$ and $\{\gamma_{q,d}^{(t)}\}$ and the observed data in $\mathcal{L}$.

$$
\begin{aligned}
P(E=1,R=1|C=1,q,d,k) &= 1\\
P(E=1,R=0|C=0,q,d,k) &= \frac{\theta_k^{(t)}(1-\gamma_{q,d}^{(t)})}{1-\theta_k^{(t)}\gamma_{q,d}^{(t)}}\\
P(E=0,R=1|C=0,q,d,k) &= \frac{(1-\theta_k^{(t)})\gamma_{q,d}^{(t)}}{1-\theta_k^{(t)}\gamma_{q,d}^{(t)}}\\
P(E=0,R=0|C=0,q,d,k) &= \frac{(1-\theta_k^{(t)})(1-\gamma_{q,d}^{(t)})}{1-\theta_k^{(t)}\gamma_{q,d}^{(t)}}
\end{aligned}
\tag{1}
$$

From this, we can compute the marginals $P(E=1|c,q,d,k)$ and $P(R=1|c,q,d,k)$ for every data point in $\mathcal{L}$ and this can be seen as complete data where the hidden variables are estimated.

The Maximization step updates the parameters using the quantities from the Expectation step:

$$
\begin{aligned}
\theta_k^{(t+1)} &= \frac{\sum_{c,q,d,k'} \mathbb{I}_{k'=k}\cdot(c+(1-c)P(E=1|c,q,d,k))}{\sum_{c,q,d,k'}\mathbb{I}_{k'=k}}\\
\gamma_{q,d}^{(t+1)} &= \frac{\sum_{c,q',d',k}\mathbb{I}_{q'=q,d'=d}\cdot(c+(1-c)P(R=1|c,q,d,k))}{\sum_{c,q',d',k,}\mathbb{I}_{q'=q,d'=d}}
\end{aligned}
\tag{2}
$$

4.3.2 Regression-based EM. The Maximization step in the standard EM usually works with (q,d) identifiers for $\gamma_{q,d}$. Using the exact identifiers is challenging in personal search for several reasons: (1) due to privacy concerns, these identifiers may not be accessible; (2) the click data is highly sparse and noisy since documents are private and clicks on a document may just come from a single user; (3) user corpora change quickly and a document may get very few clicks before it becomes irrelevant when a new document (e.g. email) is created. To overcome these difficulties, we propose a regression-based EM.

The regression-based EM only modifies the Maximization step in the standard EM. Instead of working with (q,d) identifiers, we assume 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. For any ranking system, there are usually available ranking features used in the system. The feature vector $\mathbf{x}$ in our EM can be the same as the ranking features. Thus, the proposed EM algorithm can be easily deployed to any of these systems where regular clicks and ranking features are collected. Also, as noted in [7], for the EM to work the standard EM requires that a document for a query appears in multiple positions. This is less of a concern in regression-based EM as long as similar feature vectors appear in different positions.

Specifically, for each $(c,q,d,k)\in\mathcal{L}$, the expectation step gives a probability $P(R=1|c,q,d,k)$. Intuitively, we can regress the feature vector $\mathbf{x}_{q,d}$ to the probability $P(R=1|c,q,d,k)$. In this paper, 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 the 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:

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

Algorithm 1: Regression-based EM

Input: $\mathcal{L} = \{(c, q, d, k)\}$, $\{\mathbf{x}_{q,d}\}$, $\{\theta_k\}$, $\{\gamma_{q,d}\}$.
Output: $\{\theta_k\}$ and $f(\mathbf{x})$.

```
Let F(x) = 0
repeat
    Estimate the hidden variable probability based on Eq 1.
    Let S = {}
    for all (c, q, d, k) ∈ L do
        Sample r ∈ {0, 1} from P(R = 1|c, q, d, k)
        S = S ∪ (x_{q,d}, r)
    end for
    F(x) = GBDT(F(x), S)
    Update {θ_k} based on Eq 2.
    Update {γ_{q,d} = f(x_{q,d})} using Eq 3.
until Convergence.
return {θ_k}, f(x)
```

where we use the sigmoid in the objective function

$$f(\mathbf{x}) = \frac{1}{1 + e^{-F(\mathbf{x})}} \tag{3}$$

$F(x)$ is the log odd of function $f(x)$ and can be learned by the standard logistic regression. However, the features in a ranking system are usually optimized to perform correct ordinal ranking, and may not have a linear correlation with the relevance. To account for non-linearity, we use the Gradient Boosted Decision Tree (GBDT) method [16] to learn the function $F(\mathbf{x})$.

We summarize the regression-based EM in Algorithm 1. For the GBDT, we use depth-3 trees and set the shrinkage to 0.2. A new iteration uses the GBDT trees from the previous iteration and refines them additively using the new data.

4.3.3 Embedded in Discriminative Methods. Another approach to estimate position effect is to use a discriminative method (e.g., logistic regression) by embedding positions as features. That is, we append the position of each document to the feature vector $\mathbf{x}$ and train a function to predict the click odds. We use the one-hot encoding of the positions and form a feature vector $\mathbf{k}$. Thus, for each $(c, q, d, k) \in \mathcal{L}$, we have a training instance $([\mathbf{k}, \mathbf{x}_{q,d}], c)$. A discriminative model GDBT can then be trained to optimize the likelihood of the click data. In order to separate the position effect from the ranking features, we set the split depth to be 1 in GBDT to disallow feature interaction. This is similar to the linear function used in logistic regression, but allows us to model non-linear transformation for individual features. We use $g(\mathbf{k}, \mathbf{x})$ to represent the learned function that predicts the click probability.

In the Embedded method, the positions are treated the same as the regular ranking features, so the click odds can be equally attributed to positions or to regular ranking features. When there is a high correlation between them, the attribution can be arbitrary. This also creates dependency between positions and regular click features. When positions are not present, the Embedded method cannot perform well. On the contrary, the EM algorithm is based on probabilistic graphical models and has a clear separation of a position-based bias component and a relevance component. Such a structure makes each component predictive independently. We will show this point in our experiments.

5 EVALUATIONS

In this section, we conduct a suite of experiments to compare different methods that estimate position-based propensity. We first describe our experimental setup (data sets, and metrics design), then we report our experimental results in the unbiased learning-to-rank setting.

5.1 Experimental Setup

We set up our evaluation using a standard supervised learning-to-rank framework [25]. In this section, we describe our evaluation data sets and the evaluation metrics used in our experiments.

5.1.1 Data Sets. The data sets we used in this paper are search logs from both an email and a file storage service. For each service, there is at most a single click for each query. This is because each service uses an overlay to show the results as users type and the overlay disappears when a click on the overlay happens. We discard all the queries that do not lead to clicks when we process the search logs.

For each service, we take a sample of its processed logs from a two-week period in April 2017. The first week of the data is used for training and the second is used for test. We have approximately 4M queries in each week. Each query has around 5 results. We use the existing ranking features from these services in our regression and learning-to-rank algorithms. We do not do any other pre-processing of the data.

5.1.2 Evaluation Metrics. For ranking effectiveness, the evaluation metric used in this paper is a variant of Mean Reciprocal Rank (MRR). Given a test data set with N queries, the standard MRR is defined as follows:

$$\text{MRR} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\text{rank}_i} \tag{4}$$

where rank_i is the rank of the clicked document of the i-th query in our data set.

Due to click bias, the standard MRR is not suitable for offline evaluation. We thus define a weighted MRR using the propensity. Let w_i be the inverse propensity estimated from the *RandPair* described in Section 4.2.2 for the clicked position of the i-th query, then the weighted MRR is defined as

$$\text{MRR} = \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} w_i \frac{1}{\text{rank}_i}, \tag{5}$$

which is the MRR definition that we will use in the remainder of this paper. It is the same as the $\overline{Prec}$ in Section 3 given that we only have a single click per query in our data sets. Note that we use the propensity estimated from the *RandPair* as the ground truth in the following comparisons and this offline metric is shown to be consistent with the online experiments in [31].

We also use the average log likelihood to measure how well a model fits the click data. This can be used to compare the EM and

	EM	Embedded
Email	**-0.124**	-0.130
File Storage	-0.121	**-0.116**

Table 2: The average LogLikelihood on the test data set. A lower absolute value means a better fit.

	EM	Embedded
Email	**+0.50%***	-4.44%*
File Storage	**+0.11%**	-3.10%*

Table 3: Comparison of relevant components using the relative difference of MRR against the pointwise GBDT baseline.

the Embedded methods. Specifically,

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

where $p = \theta_k f(\mathbf{x})$ for the EM method and $p = g(\mathbf{k}, \mathbf{x})$ for the Embedded method. Besides fitting wellness, we also compare the relevance components estimated by these two methods using the MRR metric. In the EM method, we use the relevance component $f(\mathbf{x})$ directly. The Embedded method does not have a clearly separated relevance component. However, since the GBDT model has split depth 1, there are no interactions between features. Hence, we can remove the position feature related subtrees and use the rest as the relevance components. This is equivalent to comparing the relevance of all the documents by placing them at the same canonical position (e.g., position 1).

5.2 Experimental Results

We report our experimental results on propensity estimation and the ranking effectiveness of different learning methods.

5.2.1 Propensity Estimation. Both EM and Embedded aim to maximize the likelihood. We show the LogLikelihood results in Table 2. On the email data, EM outperforms Embedded, but Embedded fits the file storage data better. The EM method imposes a factor model to separate the effect of position while the Embedded method is limited to a split depth equal to 1 to have a separation between position and relevance features. The LogLikelihood suggests that both EM and Embedded are comparable. However, we will show that the EM method is far better in the following aspects:

- **Estimated relevance.** Table 3 shows the evaluation of relevance components of both the EM and the Embedded methods. For the baseline, we train a pointwise GBDT model over the ranking feature vector $\mathbf{x}$ directly without considering position bias. We report the relative difference over the baseline using MRR. Surprisingly, the Embedded method performs significantly worse than the baseline, while the EM method outperforms the baseline on both data sets and the improvement is statistically significant on the email data set. This confirms that the Embedded method cannot effectively attribute the click odds to the ranking features but the EM method can.

Pointwise			Pairwise
QueryLevel	DocLevel	EM	EMCorrected
-0.07%	-0.06%	+0.50%*	+1.30%*

Table 4: Comparison of unbiased pointwise learning algorithms based on relative change of MRR against the pointwise GBDT baseline without correction.

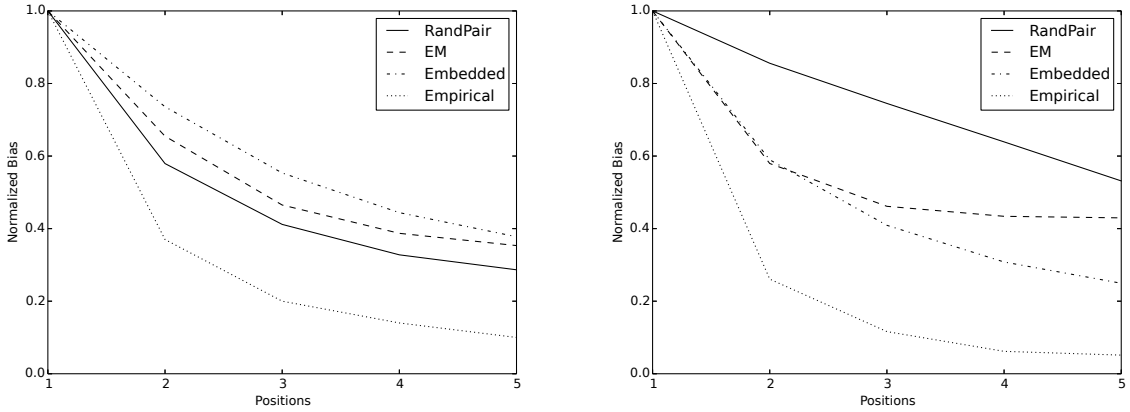
- **Estimated position bias.** Figure 2 shows the estimated position bias for both methods, together with the *RandPair* ground truth. In addition, we plot the relative empirical click rates at different positions normalized by the top position (denoted as Empirical). We can see that Empirical has more skew than *RandPair*, consistent with previous findings [7]. Both EM and Embedded differ from Empirical by taking relevance into consideration. On both data sets, EM gives much closer biases to *RandPair* than Embedded, showing that our proposed EM algorithm is effective in both relevance and position bias estimation.

5.2.2 Effects on Ranking Improvement. The position bias from the EM method is not the same as the *RandPair* method. Thus, the question is how effective the estimated propensity is to improve ranking. In this section, we leverage the estimated propensity from EM to learn a ranking function. For the baseline, we train a pairwise model on *Prec* using lambdaMART without any bias correction (denoted as NoCorrection). We use the bias estimated from *RandPair* and train a lambdaMART model to optimize IPW $\overline{Prec}$. This is the expected optimal model (denoted by RandPairCorrected). Similarly, we train a third lambdaMART ranking function to optimize the IPW $\overline{Prec}$ using EM estimated propensity scores (denoted as EMCorrected).

In Figure 3, we compare the three methods by reporting the relative MRR over the NoCorrection one. From this figure, we can see that RandPairCorrected can outperform NoCorrection significantly (+2.14% on email and +0.15% file storage). What's more important, EMCorrected achieves significant improvement over NoCorrection by +1.95% on the email data set. It performs neutrally (-0.07%) comparing with NoCorrection on the file storage data set, given that the upper bound for an improvement on file storage is very small. Overall, the EMCorrected method is shown effective in improving ranking quality without relying on randomization.

5.2.3 Unbiased Pointwise Learning. The EM method can be seen as a way to remove the bias and thus give a debiased relevance function. In this section, using the email data set, we compare it against a few other attempts for unbiased pointwise learning. As we discussed before, the IPW approach cannot be directly applied to the pointwise learning because the o_i for non-clicked documents are needed but are uncertain in click data (in contrast, $o_i = 1$ for all clicked documents). To bypass this, we use the following approximation:

- Though query-level propensity is supposed to apply to pairwise and listwise approaches, we borrow it for pointwise. In this way, all the non-clicked documents for a query will have the same IPW weights as its clicked document. We denote this by QueryLevel.



(a) Email. **(b) File Storage.**

Figure 2: Estimated position bias on the email and file storage services normalized by the top position.

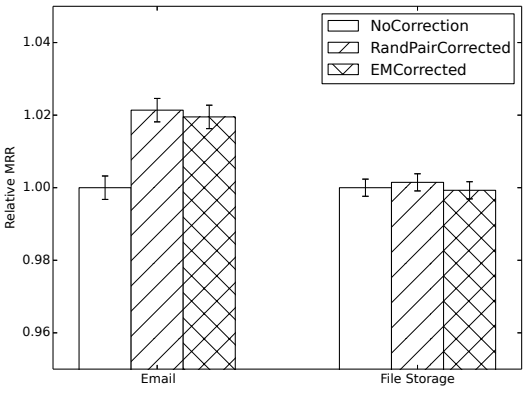


Figure 3: Ranking effectiveness comparison of pairwise approaches with different position bias correction methods.

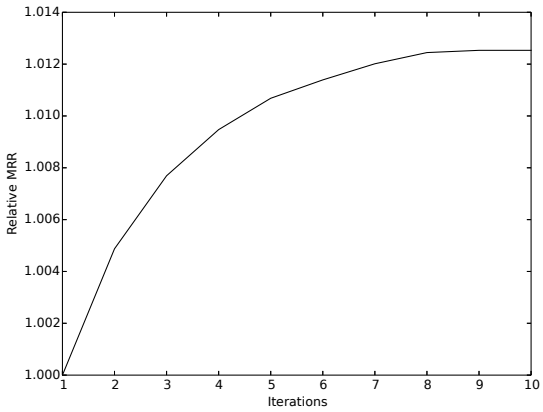


Figure 4: Convergence of EM along the iterations.

- To adapt document-level propensity, we assume $o_i = 1$ for both clicked and non-clicked documents for all the queries with clicks in our data sets. In this way, a non-clicked document has IPW weight $\frac{1}{\theta_k}$ where k is its position. We denote this by DocLevel.

In Table 4, we report the relative MRR against the pointwise model without any bias correction. As shown in the table, neither QueryLevel nor DocLevel can outperform the baseline. In contrast, the relevance function extracted from the EM algorithm is significantly better than the baseline. This shows that both QueryLevel and DocLevel have some unrealistic assumptions and are not suitable to debias the click for pointwise learning.

Furthermore, we compare the pairwise EMCorrected and the pointwise EM methods against the same baseline using the MRR metric in Table 4. Clearly, the pairwise model can leverage the estimated position bias better to improve the ranking accuracy in the learning-to-rank setting, showing the theoretical soundness of the unbiased pairwise learning-to-rank methods.

5.2.4 EM Convergence. In Figure 4, we plot the relative MRR of the relevance component against the first iteration along with the EM iterations on the email data. The results show that the EM algorithm can steadily improve the relevance function and converge quickly in about 10 iterations.

6 CONCLUSIONS

In this paper, we studied the problem of position bias estimation for unbiased learning-to-rank algorithms. We showed that the propensity estimation is equivalent to the position bias estimation in the classical position bias model. Then we compared a few estimation methods with and without result randomization. A novel regression-based EM algorithm was proposed to estimate position bias from

regular production clicks. Our methods are easily applicable to any ranking system. Experimental results on email search and file storage search click data showed that our proposed EM method works well in estimating position bias and can improve ranking effectiveness significantly.

Our work inspires a few interesting future research directions. (1) The results we showed are mainly based on pairwise approaches. It is still unclear how to design an unbiased pointwise algorithm. This is worth studying given the importance of estimating pointwise CTR in applications like online advertising. (2) The current unbiased learning-to-rank is designed for binary relevance. Extending it to graded relevance with motivating applications is another interesting direction. (3) The position bias model is a simple click model; it would be interesting to study more sophisticated click models in unbiased learning-to-rank tasks.

ACKNOWLEDGMENTS

We thank Olivier Chapelle for his discussions on tradeoff among different position bias estimation methods and Roberto Bayardo for his suggestions on improving the paper presentation.

REFERENCES

[1] Aman Agarwal, Soumya Basu, Tobias Schnabel, and Thorsten Joachims. 2017. 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)*. 687–696.

[2] Qingyao Ai, Susan T. Dumais, Nick Craswell, and Dan Liebling. 2017. Characterizing Email Search Using Large-scale Behavioral Logs and Surveys. In *Proc. of the 26th International Conference on World Wide Web (WWW)*. 1511–1520.

[3] Michael Bendersky, Xuanhui Wang, Donald Metzler, and Marc Najork. 2017. Learning from User Interactions in Personal Search via Attribute Parameterization. In *Proc. of the 10th ACM International Conference on Web Search and Data Mining (WSDM)*. 791–799.

[4] Christopher J.C. Burges. 2010. *From RankNet to LambdaRank to LambdaMART: An Overview*. Technical Report MSR-TR-2010-82. Microsoft Research. http://research.microsoft.com/apps/pubs/default.aspx?id=132652

[5] David Carmel, Guy Halawi, Liane Lewin-Eytan, Yoelle Maarek, and Ariel Raviv. 2015. Rank by Time or by Relevance?: Revisiting Email Search. In *Proc. of the 24th ACM International Conference on Information and Knowledge Management (CIKM)*. 283–292.

[6] David Carmel, Liane Lewin-Eytan, Alex Libov, Yoelle Maarek, and Ariel Raviv. 2017. Promoting Relevant Results in Time-Ranked Mail Search. In *Proc. of the 26th International Conference on World Wide Web (WWW)*. 1551–1559.

[7] Olivier Chapelle and Ya Zhang. 2009. A dynamic bayesian network click model for web search ranking. In *Proc. of the 18th International Conference on World Wide Web (WWW)*. 1–10.

[8] Weizhu Chen, Zhanglong Ji, Si Shen, and Qiang Yang. 2011. A Whole Page Click Model to Better Interpret Search Engine Click Data.. In *Proc. of the 25th AAAI Conference on Artificial Intelligence (AAAI)*.

[9] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2015. *Click Models for Web Search*. Morgan & Claypool.

[10] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An Experimental Comparison of Click Position-bias Models. In *Proc. of the 1st International Conference on Web Search and Data Mining (WSDM)*. 87–94.

[11] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)* 39 (1977), 1–38. Issue 1.

[12] Miroslav Dudík, John Langford, and Lihong Li. 2011. Doubly Robust Policy Evaluation and Learning. In *Proc. of the 28th International Conference on Machine Learning (ICML)*. 1097–1104.

[13] Susan Dumais, Edward Cutrell, JJ Cadiz, Gavin Jancke, Raman Sarin, and Daniel C. Robbins. 2003. Stuff I've Seen: A System for Personal Information Retrieval and Re-Use. In *Proc. of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 72–79.

[14] Georges E. Dupret and Benjamin Piwowarski. 2008. 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)*. 331–338.

[15] David Elsweiler, Morgan Harvey, and Martin Hacker. 2011. Understanding re-finding behavior in naturalistic email interaction logs. In *Proc. of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 35–44.

[16] Jerome H. Friedman. 2000. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics* 29 (2000), 1189–1232.

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

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

[19] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. 2005. 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)*. 154–161.

[20] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased Learning-to-Rank with Biased Feedback. In *Proc. of the 10th ACM International Conference on Web Search and Data Mining (WSDM)*. 781–789.

[21] Maryam Kamvar, Melanie Kellar, Rajan Patel, and Ya Xu. 2009. Computers and iPhones and Mobile Phones, Oh My!: A Logs-based Comparison of Search Users on Different Devices. In *Proc. of the 18th International Conference on World Wide Web (WWW)*. 801–810.

[22] Jin Young Kim, Nick Craswell, Susan Dumais, Filip Radlinski, and Fang Liu. 2017. Understanding and Modeling Success in Email Search. In *Proc. of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 265–274.

[23] Lihong Li, Shunbao Chen, Jim Kleban, and Ankur Gupta. 2015. Counterfactual Estimation and Optimization of Click Metrics in Search Engines: A Case Study. In *Proc. of the 24th International Conference on World Wide Web (WWW) Companion*. 929–934.

[24] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. 2011. 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)*. 297–306.

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

[26] Maeve O'Brien and Mark T Keane. 2006. Modeling result–list searching in the World Wide Web: The role of relevance topologies and trust bias. In *Proc. of the 28th Annual Conference of the Cognitive Science Society (CogSci)*. 1–881.

[27] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting Clicks: Estimating the Click-Through Rate for New Ads. In *Proc. of the 16th International Conference on World Wide Web (WWW)*. 521–530.

[28] Paul R Rosenbaum and Donald B Rubin. 1983. The Central Role of the Propensity Score in Observational Studies for Causal Effects. *Biometrika* 70, 1 (1983), 41–55.

[29] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations As Treatments: Debiasing Learning and Evaluation. In *Proc. of the 33rd International Conference on International Conference on Machine Learning (ICML)*. 1670–1679.

[30] Adith Swaminathan and Thorsten Joachims. 2015. Batch Learning from Logged Bandit Feedback Through Counterfactual Risk Minimization. *Journal of Machine Learning Research* 16 (2015), 1731–1755. Issue 1.

[31] Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. 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)*. 115–124.

[32] Yisong Yue, Rajan Patel, and Hein Roehrig. 2010. Beyond Position Bias: Examining Result Attractiveness As a Source of Presentation Bias in Clickthrough Data. In *Proc. of the 19th International Conference on World Wide Web (WWW)*. 1011–1018.

[33] Hamed Zamani, Michael Bendersky, Xuanhui Wang, and Mingyang Zhang. 2017. Situational Context for Ranking in Personal Search. In *Proc. of the 26th International Conference on World Wide Web (WWW)*. 1531–1540.

[34] Zeyuan Allen Zhu, Weizhu Chen, Tom Minka, Chenguang Zhu, and Zheng Chen. 2010. A novel click model and its applications to online advertising. In *Proc. of the 3rd ACM International Conference on Web Search and Data Mining (WSDM)*. 321–330.