Pessimistic Off-Policy Optimization for Learning to Rank

Paper #793

Abstract. Off-policy learning is a framework for optimizing policies without deploying them, using data collected by another policy. In recommender systems, this is especially challenging due to the imbalance in logged data: some items are recommended and thus logged more frequently than others. This is further perpetuated when recommending a list of items, as the action space is combinatorial. To address this challenge, we study pessimistic off-policy optimization for learning to rank. The key idea is to compute lower confidence bounds on parameters of click models and then return the list with the highest pessimistic estimate of its value. This approach is computationally efficient, and we analyze it. We study its Bayesian and frequentist variants and overcome the limitation of unknown prior by incorporating empirical Bayes. To show the empirical effectiveness of our approach, we compare it to off-policy optimizers that use inverse propensity scores or neglect uncertainty. Our approach outperforms all baselines, is robust, and is also general.

1 Introduction

Off-policy optimization is used to learn better policies in systems, where deploying sub-optimal solutions is costly, for example, recommender systems [10]. Despite the obvious benefits, off-policy optimization is often impeded by the *feedback loop*, where the earlier versions influence the training data in future iterations [13]. This type of bias in data is one of the main issues with off-policy optimization.

Several unbiased learning strategies exist to learn from biased data. Amongst the most popular approaches is *inverse propensity scoring (IPS)*, which re-weights observations with importance weights [11] to estimate a policy value. This so-called off-policy evaluation is often used in off-policy optimization, finding the policy with the highest estimated value [13]. While IPS is commonly used in practice [2], it has variance issues that compound at scale, which may prevent a successful deployment [6]. An important scenario where IPS has a high variance is recommending a ranked list of items. In this case, the action space is combinatorial, as the number of ranked lists, which represent actions, is exponential in the length of the lists.

Therefore, in real-world ranking problems (e.g., news, web search, and e-commerce), model-based methods often outperform IPS methods [14]. The model-based methods rely on an explicit model of the reward conditioned on a context-action pair, e.g., the probability of a user clicking on a given recommendation [8]. A prevalent approach to fitting model parameters, *maximum likelihood estimation (MLE)*, is impacted by non-uniform data collection. Consider choosing between two restaurants where the first has an average rating of 5.0 with five reviews, and the second has a rating of 4.8 with a thousand reviews. Optimizers using MLE would choose the first restaurant, as they consider only the average rating, while the second choice is safer.

In our work, we account for the uncertainty caused by unevenly explored action space by applying pessimism to reward models of action-context pairs for learning to rank. The challenge is to design lower confidence bounds that hold jointly for all lists as the number of unique lists grows exponentially with the list length. A naïve application of existing pessimistic methods to each unique list is sample inefficient. Also, user behavior signals are often biased for higher-ranked items and can only be collected on items that users actually saw. The main contributions of our paper are:

- We propose *lower confidence bounds (LCBs)* on parameters of model-based approaches in learning to rank and derive error bounds for acting on them in off-policy optimization.
- We study both Bayesian and frequentist approaches to estimating LCBs, including an empirical estimation of the prior, as it is often unknown in practice.
- We conduct extensive experiments that show the superiority of the proposed methods compared to IPS and MLE policies on four real-world learning to rank datasets with a large action space.

2 Related Work

Off-Policy Optimization: One popular approach to learning from bandit feedback is to employ the empirical risk minimization principle with IPS-based estimators [19, 2, 32]. An alternative to using IPS in learning from bandit feedback is the model-based approach. These approaches learn a reward regression model for specific context-action pairs, which is then used to derive an optimal policy. However, due to model misspecification, model-based methods are typically biased but have more favorable variance properties than IPS models [13]. Variance issues of IPS-based estimators are further perpetuated in the learning to rank problems as the action space grows at a combinatorial pace.

Counterfactual Learning to Rank: Training of learning to rank models is often done by leveraging feedback from user behavior as an alternative data source [16]. However, implicit feedback, such as user clicks, is noisy and affected by various kinds of biases [17]. Many studies have explored how to extract unbiased relevance signals from biased click signals. One approach is to model examination probability by using click models [3, 4]. While IPS estimators based on various click models have been studied in the past [25], the key assumption was that the value of a list is linear in the contributions of individual items in the list. IPS estimators have unbiased properties, and increased variance can be mitigated by various ways [20, 32, 33], for example, capping the probability ratio to a fixed value [12], but they fail to model a non-linear structure of a list.

Model-based methods can capture that non-linearity, but they suffer from biased estimates due to unexplored action space. While previous works for counterfactual learning to rank were focused mostly on evaluation [25, 34, 19], they use linear estimators for the objective function, such as the item-position model [4] and pseudoinverse estimator. More recently, a doubly robust method under the cascade model has been proposed that induces a much weaker assumption [22]. Although it is possible to use these methods for optimization, they still suffer from overly optimistic estimations - a phenomenon known as "the Optimiser's curse" [30]. Our proposed method works with both linear and non-linear click models while alleviating the Optimiser's curse.

Pessimistic Off-Policy Optimization: While off-policy methods learn from data that was collected under a different policy, on-policy methods learn from the data they collected. In online learning, the policy needs to balance the immediate reward of action with the informational value for future actions [27]. Here, the common approach is to be optimistic about the potential reward of action and methods using an *upper confidence bound (UCB)* proved to be successful [24].

In an offline setting, as the methods cannot learn directly from the actions, we need to be pessimistic (as we have only one shot). Pessimistic LCBs on a reward model were applied using Bayesian uncertainty estimates and achieved a robust increase in the performance [13]. Principled Bayesian methods can be used to obtain closed-form expressions, but they require to know prior in advance, and they are often restricted to specific model classes [13, 3, 24].

We are the first to apply pessimism to model the reward function in learning to rank. While pessimism is popular in offline reinforcement learning [37], regarding the recommender systems domain, it was applied only in a single-recommendation scenario and did not consider structured actions [13]. We extend this work from pointwise to list-wise pessimism and compare multiple approaches for constructing pessimistic estimates.

3 Setting

We start with introducing our setting. Specifically, we formally define a ranked list, how a user interacts with it, and how the data for off-policy optimization are collected.

We consider the following general model of a user interacting with a ranked list of items. Let $\mathcal{E}$ be a *ground set of items*, such as all web pages or movies that can be recommended. Let $\Pi_K(\mathcal{E})$ be the set of all lists of length K over items $\mathcal{E}$. A user is recommended a ranked list of items. We denote a *ranked list* with K items by $A = (a_1, \ldots, a_K) \in \Pi_K(\mathcal{E})$, where $a_k \in \mathcal{E}$ is the item at position k. The user clicks on items in the list and we observe click indicators on all positions $Y = (Y_1, \ldots, Y_K)$, where $Y_k \in \{0, 1\}$ is the *click indicator* on position k. The list is chosen as a function of *context* $X \in \mathcal{X}$, where X can be a user profile or a search query coming from a set of contexts $\mathcal{X}$.

A ranking *policy* $\pi(\cdot \mid X)$ is a conditional probability distribution over lists given context X. It interacts with users for n rounds indexed by $t \in [n]$. In round t, π observes context X_t and then selects a list $A_t \sim \pi(\cdot \mid X_t)$, where $A_t = (a_{t,1}, \ldots, a_{t,K}) \in \Pi_K(\mathcal{E})$. After that, it observes clicks $Y_t = (Y_{t,1}, \ldots, Y_{t,K})$ on all recommended items in the list. All interactions are recorded in a *logged dataset* $\mathcal{D} = \{(X_t, A_t, Y_t)\}_{t=1}^n$. The policy that collects $\mathcal{D}$ is called the *logging policy* and we denote it by π_0.

Our goal is to find a policy that recommends the *optimal list* in every context. The optimal list in context X is defined as

$$A_{*,X} = \operatorname*{arg\,max}_{A \in \Pi_K(\mathcal{E})} V(A, X)\,, \tag{1}$$

where $V(A, X)$ is the value of list A in context X. This can be the expected number of clicks or the probability of observing a click. The definition of V depends on the chosen user interaction model and we present several choices in Section 5.

Algorithm 1 Conservative off-policy optimization.

Inputs: Logged dataset $\mathcal{D}$
for $X \in \mathcal{X}$ **do**
  $\hat{A}_X \leftarrow \arg\max_{A \in \Pi_K(\mathcal{E})} L(A, X)$
end for
Output: $\hat{A} = (\hat{A}_X)_{X \in \mathcal{X}}$

4 Pessimistic Optimization

Suppose that we want to find list $A_{*,X}$ in (1) but $V(A, X)$ is unknown. Then the most straightforward approach is to estimate it and choose the best list according to the estimate. As an example, let $\hat{V}(A, X)$ be a *maximum likelihood estimate (MLE)* of $V(A, X)$. Then the best empirically-estimated list in context X would be

$$\hat{A}_X = \arg\max_{A \in \Pi_K(\mathcal{E})} \hat{V}(A, X)\,. \tag{2}$$

This approach is problematic when $\hat{V}$ is a poor estimate of V. Specifically, we may choose a list with a high estimated value $\hat{V}(\hat{A}_X, X)$ but low actual value $V(\hat{A}_X, X)$ when $\hat{V}(\hat{A}_X, X)$ is a highly-uncertain estimate of $V(\hat{A}_X, X)$.

To account for uncertainty, prior works in bandits and reinforcement learning designed pessimistic *lower confidence bounds (LCB)* and acted on them [15]. We adopt the same design principle in our proposed algorithm, which we present in Algorithm 1. At a high level, the algorithm first computes an LCB for each action-context pair (A, X), denoted by $L(X, A)$. The lower confidence bound satisfies $L(A, X) \leq V(A, X)$ with a high probability. Then it takes an action $\hat{A}_X$ with the highest lower confidence bound $L(\cdot, X)$ in each context X. In Sections 5 and 6, we show how to design LCBs for entire lists of items efficiently. These LCBs, and our subsequent analysis in Section 7, are our main technical contributions.

Lower confidence bounds are beneficial when $\hat{V}$ does not approximate V uniformly well. Specifically, suppose that $\hat{V}$ approximates V better around optimal solutions $A_{*,X}$. This is common in practice, as deployed logging policies π_0 are already optimized to select high-value items. Then low-value items can only be chosen if the LCBs of high-value items are low. This cannot happen because the high-value items are logged frequently; and thus their estimated mean values are high and their confidence intervals are tight.

As a concrete example, consider two lists of recommended items. The first list contains items with an estimated click-through rate (CTR) of 1, but all of them were recommended only once. The other list contains items with an estimated CTR of 0.5, but those items are popular and were recommended a thousand times. Off-policy optimization with the MLE estimator would choose the first list, whose estimated value is high but the actual value may be low. Off-policy optimization with LCBs would choose the other list, since its estimated value is reasoanbly high but more certain.

5 Structured Pessimism

In this section, we construct lower confidence bounds for lists. The main challenge is how to establish useful LCBs for all lists jointly, since there can be exponentially many lists. To do that, we rely on user-interaction models with ranked lists, the so-called click models [4]. The models allow us to construct LCBs for the whole list by decomposing it into LCBs of items in it.

To illustrate the generality of our approach, we study three popular click models. To simplify notation, we assume that the context X is fixed in this section and drop it from all terms. In each click model, the relevance of item $a \in \mathcal{E}$ is described by its attraction probability $\theta_a \in [0,1]$. This is the probability that the item is clicked given being examined. In each model, we show that when we have LCBs for each θ_a, we have LCBs for all lists A, trivially by the union bound.

5.1 Cascade Model

The *cascade model (CM)* [28, 5] assumes that a user scans items in a list from top to bottom until they find a relevant item [4]. Under this assumption, item a_k at position k is examined if and only if item a_{k-1} at the previous position is examined but not clicked. The item at the first position is always examined. It follows that at most one item is clicked in the CM. Therefore, a natural choice for the value of list A is the probability of a click defined as

$$V_{\text{CM}}(A) = 1 - \prod_{k=1}^{K}(1 - \theta_{a_k})\,, \tag{3}$$

where $\theta_a \in [0,1]$ denotes the attraction probability of item $a \in \mathcal{E}$. To stress that the above value is for a specific model, the CM in this case, we write V_{CM}. The optimal list A_* contains K items with the highest attraction probabilities [23].

To establish LCBs for all lists, we need LCBs for all model parameters. In the CM, the value of a list depends only on the attraction probabilities of its items. Let $L(a)$ be the LCB on the attraction probability of item a, where $\theta_a \geq L(a)$ holds with probability at least $1-\delta$. Then for all lists A jointly, the LCB

$$L_{\text{CM}}(A) = 1 - \prod_{k=1}^{K}(1 - L(a_k)) \leq 1 - \prod_{k=1}^{K}(1 - \theta_{a_k})$$

holds with probability at least $1 - \delta\,|\mathcal{E}|$, by the union bound over all items. The above inequality holds because we have a lower bound on each term in the product.

5.2 Dependent-Click Model

The *dependent-click model (DCM)* [7] extends the CM to multiple clicks. This model assumes that after a user clicks on an item, they may continue examining items at lower positions in the list. Specifically, at position $k \in [K]$, the probability that the user continues to explore after a click is denoted by $\lambda_k \in [0,1]$.

A natural choice for the value of list A in the DCM is the probability of a satisfactory click, a click upon which the user leaves satisfied. This can be formally written as

$$V_{\text{DCM}}(A) = 1 - \prod_{k=1}^{K}(1 - (1-\lambda_k)\theta_{a_k})\,, \tag{4}$$

where $\theta_a \in [0,1]$ denotes the attraction probability of item $a \in \mathcal{E}$, identically to Section 5.1. The optimal list A_* contains K items with the highest attraction probabilities, where the k-th most attractive item is placed at the k-th most satisfactory position [21].

Let $L(a)$ be defined as in Section 5.1. Then for all lists A jointly, the LCB

$$\begin{aligned} L_{\text{DCM}}(A) &= 1 - \prod_{k=1}^{K}(1 - (1-\lambda_k)L(a_k)) \\ &\leq 1 - \prod_{k=1}^{K}(1 - (1-\lambda_k)\theta_{a_k}) \end{aligned}$$

holds with probability at least $1 - \delta\,|\mathcal{E}|$, by the union bound over all items. We assume that the position parameters λ_k are known, although we could also estimate them and plug in their LCBs.

5.3 Position-Based Model

The *position-based model (PBM)* [5] assumes that the click probability depends only on the item and its position, and allows multiple clicks. This is modeled through the examination probability $p_k \in [0,1]$ of position $k \in [K]$. Specifically, the item is clicked only if its position is examined and the item is attractive.

A natural choice for the value of list A in the PBM is the expected number of clicks

$$V_{\text{PBM}}(A) = \sum_{k=1}^{K} \theta_{a_k} p_k\,, \tag{5}$$

where $\theta_a \in [0,1]$ denotes the attraction probability of item $a \in \mathcal{E}$, identically to Section 5.1. The optimal list A_* contains K items with the highest attraction probabilities, where the k-th most attractive item is placed at the position with the k-th highest p_k.

Let $L(a)$ be defined as in Section 5.1. Then for all lists A jointly, the LCB

$$L_{\text{PBM}}(A) = \sum_{k=1}^{K} p_k L(a_k) \leq \sum_{k=1}^{K} p_k \theta_{a_k}$$

holds with probability at least $1 - \delta\,|\mathcal{E}|$, by the union bound over all items. Similarly to the DCM (Section 5.2), we assume that the position examination probabilities p_k are known.

6 Lower Confidence Bounds on Attraction Probabilities

In this section, we describe how to construct LCBs for attraction probabilities θ_a of individual items in Section 5. Note that these are means of Bernoulli random variables, which we use in our derivations. We consider two kinds of LCBs: Bayesian and frequentist. The Bayesian bounds assume that the attraction probabilities are drawn i.i.d. from a prior distribution, which is used in the construction of the bounds. The frequentist bounds make no assumption on the distribution of the attraction probabilities. The Bayesian bounds are more practical when the prior is available, while the frequentist bounds are more robust due to fewer modeling assumptions. Our bounds are derived independently for each context X. To simplify notation, we drop it in the derivations in this section.

6.1 Bayesian Lower Confidence Bounds

Let $\theta_a \in [0,1]$ be the mean of a Bernoulli random variable representing the attraction probability of item a. Let $Y_1, \dots, Y_n$ be n i.i.d. observations of θ_a. Let the number of positive observations be $n_{a,+}$ and the number of negative observations be $n_{a,-}$. We show how we estimate $n_{a,+}$ and $n_{a,-}$ for each click model in Appendix A. In the Bayesian setting, we make an additional assumption that $\theta_a \sim \mathrm{Beta}(\alpha, \beta)$. By definition, $\theta_a \mid Y_1, \dots, Y_n \sim \mathrm{Beta}(\alpha + n_{a,+}, \beta + n_{a,-})$. Consequently, a $1-\delta$ lower confidence bound on θ_a is $L(a) =$

$$\max\left\{\ell \in [0,1] : \int_{z=0}^{\ell} \mathrm{Beta}(z; \alpha + n_{a,+}, \beta + n_{a,-})\,\mathrm{d}z \leq \frac{\delta}{2}\right\} \tag{6}$$

According to (6), $L(a)$ is the largest value such that the attraction probability $\theta_a \leq L(a)$ is at most $\frac{\delta}{2}$ [1]. Note that in (6), $L(a)$ is found as a quantile of the probability density.

6.2 *Frequentist Lower Confidence Bounds*

If the prior of θ_a is unknown or poorly estimated, Bayesian estimates could be biased. In this case, Hoeffding's inequality would be preferred, as it provides a confidence interval for any random variable on $[0, 1]$. Specifically, let θ_a be any value in $[0, 1]$, and all other quantities be defined as in the Bayesian estimator. Then a $1 - \delta$ lower confidence bound on θ_a is

$$L(a) = \frac{n_{a,+}}{n_{a,+} + n_{a,-}} - \sqrt{\log(1/\delta)/(2n_a)}\,, \tag{7}$$

where $\frac{n_{a,+}}{n_{a,+}+n_{a,-}}$ is the MLE, $n_a = n_{a,+} + n_{a,-}$, and event $\theta_a \leq L(a)$ occurs with probability at most δ [36].

6.3 *Prior Estimation*

One shortcoming of Bayesian methods is that the prior is often unknown. To address this issue, we show how to estimate it using empirical Bayes [26], which can be implemented for attraction probabilities as follows. Let $|\mathcal{E}|$ be the number of attraction probabilities of different items. For any $a \in \mathcal{E}$, let $\theta_a \sim \mathrm{Beta}(\alpha, \beta)$ be the mean of a Bernoulli random variable, which is drawn i.i.d. from the unknown prior $\mathrm{Beta}(\alpha, \beta)$. Let $N_{a,+}$ and $N_{a,-}$ be the random variables that denote the number of positive and negative observations, respectively, of θ_a. Let $n_{a,+}$ and $n_{a,-}$ be the actual observed values, and $n_a = n_{a,+} + n_{a,-}$. Then the likelihood of the observations for any fixed α and β is

$$\begin{aligned}
\mathcal{L}(\alpha, \beta) &= \prod_{a \in \mathcal{E}} \mathbb{P}\left(N_{a,+} = n_{a,+},\, N_{a,+} = n_{a,+} \,|\, \alpha, \beta\right) \\
&= \prod_{a \in \mathcal{E}} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \int_{z_a=0}^{1} z_a^{\alpha+n_{a,+}-1}(1-z_a)^{\beta+n_{a,-}-1}\,\mathrm{d}z_a \\
&= \prod_{a \in \mathcal{E}} \frac{\Gamma(\alpha+\beta)\Gamma(\alpha+n_{a,+})\Gamma(\beta+n_{a,-})}{\Gamma(\alpha)\Gamma(\beta)\Gamma(\alpha+\beta+n_i)}\,.
\end{aligned} \tag{8}$$

The last equality follows from the fact that

$$\int_{z_a=0}^{1} \frac{\Gamma(\alpha+\beta+n_i) z_a^{\alpha+n_{a,+}-1}(1-z_a)^{\beta+n_{a,-}-1}}{\Gamma(\alpha+n_{a,+})\Gamma(\beta+n_{a,-})}\,\mathrm{d}z_a = 1\,.$$

The empirical Bayes [26] is a statistical procedure that chooses (α, β) that maximize $\mathcal{L}(\alpha, \beta)$. To find the maximizer, we search on a grid. For instance, let $\mathcal{G} = [m]$ for some integer $m > 0$. Then we search over all $(\alpha, \beta) \in \mathcal{G}^2$. In this case, grid search is feasible since the parameter space has only 2 dimensions.

7 Analysis

The analysis is structured as follows. First, we derive confidence intervals for items and lists. Then we show how the error of acting with respect to LCBs is bounded. Finally, we discuss how different choices of π_0 affect the error in Theorem 3. All proofs are presented in Appendix B. We conduct a frequentist analysis, based on the confidence intervals in Section 6.2. A similar analysis could be done for the Bayesian setting (Section 6.1). Note the analysis is exact for the CM and DCM. For PBM, it is approximate, and we say how we approximate $n_{a,X}$ in Appendix A.

For any item $a \in \mathcal{E}$ and context $X \in \mathcal{X}$, let $\theta_{a,X} \in [0, 1]$ be the true unknown attraction probability of item a in context X, and $\hat{\theta}_{a,X} = n_{a,X,+}/n_{a,X}$ be its empirical estimate in (7), where $n_{a,X,+}$ is the number of clicks on item a in context X and $n_{a,X}$ is the number of times user observed item a in context X . Then, we get the following concentration bound on the attraction probabilities of items.

Lemma 1 (Concentration for item estimates)**.** *Let*

$$c(a, X) = \sqrt{\log(1/\delta)/(2n_{a,X})}\,.$$

Then for any item $a \in \mathcal{E}$ and context $X \in \mathcal{X}$, $\left|\hat{\theta}_{a,X} - \theta_{a,X}\right| \leq c(a, X)$ holds with probability at least $1 - \delta$.

For any list $A \in \Pi_K(\mathcal{E})$ and context $X \in \mathcal{X}$, let $V(A, X)$ be its value in context X using the true unknown attraction probabilities $\theta_{a,X}$ and $\hat{V}(A, X)$ be its estimated value using $\hat{\theta}_{a,X}$, for any click model introduced in Section 5. Then we get the following concentration bound on list values.

Lemma 2 (Concentration for list estimates)**.** *Let*

$$c(A, X) = \sum_{a \in A} \sqrt{\log(|\mathcal{E}|\,|\mathcal{X}|\,/\delta)/(2n_{a,X})}\,.$$

Then for any list $A \in \Pi_K(\mathcal{E})$ and context $X \in \mathcal{X}$, and any click model in Section 5, $\left|\hat{V}(A, X) - V(A, X)\right| \leq c(A, X)$ holds with probability at least $1 - \delta$, jointly over all A and X.

Now we show how the error due to acting pessimistically does not depend on the uncertainty of the chosen list but on the confidence interval width of optimal list $c(A_{*,X}, X)$. This is desirable, if the logging policy already performs well (Section 4).

Theorem 3 (Error of acting pessimistically)**.** *Let $A_{*,X}$ and $\hat{A}_X$ be defined as in* (1) *and Algorithm 1, respectively. Let $L(A, X) = \hat{V}(A, X) - c(A, X)$ be a high-probability lower bound on the value of list A in context X, where all quantities are defined as in Lemma 2. Then for any context X, the error of acting with respect to a lower confidence bound satisfies*

$$\begin{aligned}
V(A_{*,X}, X) - V(\hat{A}_X, X) &\leq 2c(A_{*,X}, X) \\
&\leq 2 \sum_{a \in A_{*,X}} \sqrt{\frac{\log(|\mathcal{E}|\,|\mathcal{X}|\,/\delta)}{2n_{a,X}}}
\end{aligned}$$

with probability at least $1 - \delta$, jointly over all X.

Theorem 3 shows that our error depends on the number of observations of items in the optimal list $a \in A_{*,X}$. To illustrate how it depends on the logging policy π_0, fix context $X \in \mathcal{X}$ and let n_X be the number of logged lists in context X. We discuss two cases.

Suppose that π_0 is uniform, and thus each item $a \in \mathcal{E}$ is placed at the first position with probability $1/\,|\mathcal{E}|$. Moreover, suppose that the first position is examined with a high probability. This occurs with probability 1 in the CM (Section 5.1) and DCM (Section 5.2), and with a high probability in the PBM (Section 5.3) when p_1 is high. Then $n_{a,X} = \tilde{\Omega}(n_X/\,|\mathcal{E}|)$ as $n_X \to \infty$ and thus the error bound in Theorem 3 is $\tilde{O}(K\sqrt{|\mathcal{E}|\,/n_X})$, where $\tilde{O}$ and $\tilde{\Omega}$ are asymptotic notation up to logarithmic factors. The bound is independent of the number of lists $|\Pi_K(\mathcal{E})|$, which is exponentially large.

Now suppose that we have a near-optimal logging policy. One way of formalizing this is as placing each item $a \in A_{*,X}$ at the first position with probability $1/K$. Then, under the same assumptions as in the earlier discussion, $n_{a,X} = \tilde{\Omega}(n_X/K)$ as $n_X \to \infty$ and the bound in Theorem 3 is $\tilde{O}(K\sqrt{K/n_X})$. Note that this bound improves by a factor of $|\mathcal{E}|/K$ upon the earlier discussed bound.

8 Experiments

We conduct extensive experiments where we compare learned policies by our method (Algorithm 1) to four baselines: MLE, IPS [29], structural item-position IPS [25], and pseudoinverse estimator [34]. We refer to our method as LCB because it optimizes lower confidence bounds.

8.1 Experimental Setup

We use the *Yandex* dataset [38] for the first four experiments. We treat each query as a different context X, perform the computations separately, and then average the results. Due to a huge position bias in the dataset, where most of the clicks occur at the first positions, we only keep the top 4 items in each list and discard the rest, as done in other works [25]. We observe improvements without this preprocessing step, but they are less pronounced.

All compared off-policy optimization methods are evaluated as follows. We first fit a click model from Section 5 to a given dataset. Because of that, we can efficiently compute the optimal list $A_{*,X}$ in each context X under that model. Then, for a given query, we randomly select a list of items from the original dataset and generate clicks based on the fitted click model. We repeat this n times and get a logged dataset $\mathcal{D} = \{(X_t, A_t, Y_t)\}_{t=1}^{n}$, where X_t and A_t are taken from the original dataset, and Y_t is generated by the fitted click model. After that, we apply off-policy methods to $\mathcal{D}$ to find the most valuable lists $\hat{A}_X$. We evaluate these lists against the true optimal lists $A_{*,X}$ using *error* $\mathbb{E}_X\left[V(A_{*,X}, X) - V(\hat{A}_X, X)\right]$. We estimate the logging policy π_0 from $\mathcal{D}$. We repeat each experiment 500 times, and report the mean and standard error of the results (shaded areas around the lines).

We experiment with both Bayesian and frequentist lower confidence bounds in Section 6. They hold with probability $1-\delta$, where $\delta \in [0.05, 1]$ is a tunable parameter representing the width of the confidence interval. The estimation of our model parameters is detailed in Appendix A. As Bayesian methods depend on the prior, we also evaluate Empirical Bayes for learning the prior (Section 6.3) with grid $\mathcal{G} = \{2^{i-1}\}_{i=1}^{10}$.

8.2 Baselines

One of our baselines is the best list under the same click model with MLE-estimated parameters. We also use relevant baselines from prior works [12, 25, 34].

IPS: We implement an estimator using propensity weights, where the whole list is a unique action. We compute the propensity weights separately for each query. We also implement *tunable clipping parameter* M [12]. IPS optimizer then selects $\hat{A}_X$ that maximizes

$$\hat{V}_{\text{IPS}}(A, X) = \sum_{t \in \mathcal{T}_X} \min\left\{M, \frac{\mathbb{1}\{A_t = A\}}{p_{A,X}}\right\} Y_t\,, \qquad (9)$$

where we estimate propensities $p_{A,X} = \frac{\sum_{t \in \mathcal{T}_X} \mathbb{1}\{A_t = A\}}{|\mathcal{T}_X|}$ as the frequency of recommending list A, Y_t is the number of clicks in list A_t, and $\mathcal{T}_X$ is the set of all indices $t \in [n]$ such that $X_t = X$. Maximization of $\hat{V}_{\text{IPS}}(A, X)$ is a linear program, where we search over all $A \in \Pi_K(\mathcal{E})$ [31]. When showing the results in our experiments, as $|\mathcal{D}| = 1000$, we map clipping parameter M to δ using this table:

δ	.05	.1	.15	.2	.25	.35	.45	.5
M	1	5	10	50	100	300	500	600
δ	.55	.65	.75	.8	.85	.9	.95	1
M	700	900	1100	1200	1300	1400	1500	∞

Item-Position IPS: Similarly to the IPS estimator, we implement a structured IPS estimator using linearity of the item-position model [25], where the expected value of a list is the sum of attraction probabilities of its item-position entries. The list value is

$$\hat{V}_{\text{IP-IPS}}(A, X) = \sum_{t \in \mathcal{T}_X} \sum_{k=1}^{K} \min\left\{M, \frac{\mathbb{1}\{a_{t,k} = a\}}{p_{a,k,X}}\right\} Y_{t,k}\,, \qquad (10)$$

where $p_{a,k,X} = \frac{\sum_{t \in \mathcal{T}_X} \mathbb{1}\{a_{t,k} = a\}}{|\mathcal{T}_X|}$ and $\mathcal{T}_X$ is the set of all $t \in [n]$ such that $X_t = X$. To maximize $\hat{V}_{\text{IP-IPS}}(A, X)$, we select an item with the highest attraction probability for each position $k \in [K]$.

Pseudo-Inverse Estimator (PI): As the last baseline, we implement the pseudo-inverse estimator [34] designed for off-policy evaluation that also assumes that the value of a list is the sum of individual items in it. The context-specific weight vector ϕ_X can then be learned in a closed form as

$$\hat{\phi}_X = \left(\mathbb{E}_{\pi_0}\left[\mathbf{1}_A \mathbf{1}_A^T \mid X\right]\right)^{\dagger} \mathbb{E}_{\pi_0}\left[Y \mathbf{1}_A \mid X\right], \qquad (11)$$

where $\mathbf{1}_A \in \{0,1\}^{K|\mathcal{E}|}$ is a *list indicator vector* whose components indicate which item is at which position. We denote by $M^{\dagger}$ the Moore-Penrose pseudoinverse of a matrix M and by Y the sum of clicks on the list A. Note that $\hat{\phi}_X$ uses conditional expectation over $A \sim \pi_0(\cdot \mid X)$ and $Y \sim p(\cdot \mid X, A)$. As mentioned by Swaminathan et al. [34], the trained regression model can be used for off-policy optimization. We adopt this procedure, which greedily adds the most attractive items to the list from the highest position to the lowest.

8.3 Yandex Results

The experiments are organized as follows. First, we compare LCBs to all baselines on frequent queries, while we vary the confidence interval width represented by parameter δ. Second, we compare LCBs to MLE while automatically learning parameter δ from data. Finally, we study the robustness of model-based LCB estimators to model misspecification. We refer readers to Appendix C for experiments on less frequent queries where we show LCBs work well even with less data, and the experiment for choosing the right size of hyperparameter space to estimate prior using empirical Bayes from Section 6.3.[2]

Top 10 Queries: We start with evaluating all estimators on 10 most frequent queries in the Yandex dataset. Results in Figure 1 show improvements when using LCBs for all models. Specifically, for almost any δ in all models, the error is lower than using MLE. When optimizing non-linear list values, such as those in CM and DCM, we outperform all the baselines that assume linearity. In PBM, where

[2] Code from the technical appendix will be available on GitHub following the publication.

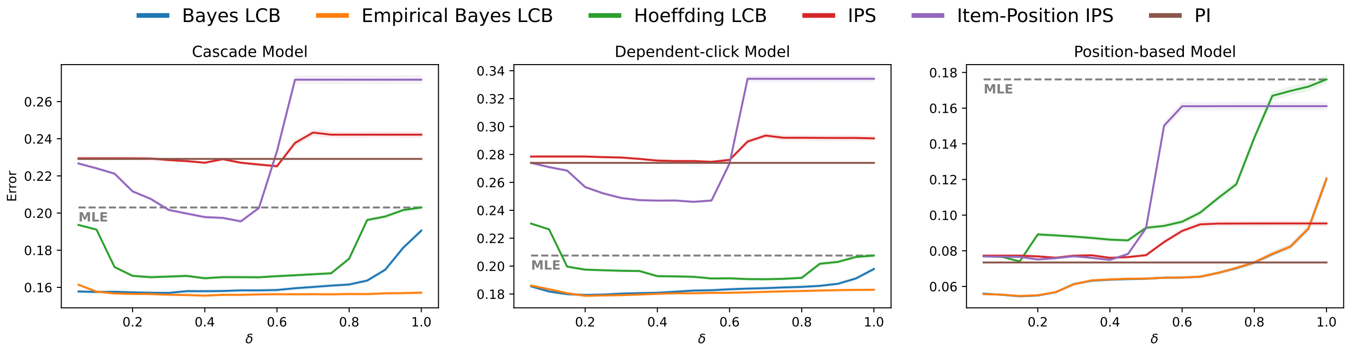
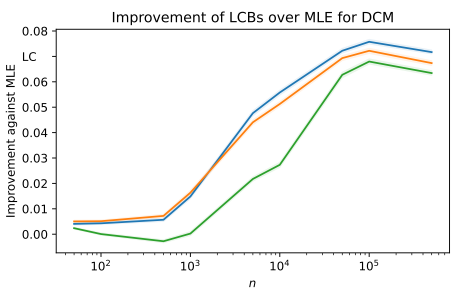


Figure 1: Comparison of our methods to baselines on three click models and top 10 queries. We vary the δ parameter that represents the confidence interval width. MLE is the grey dashed line.



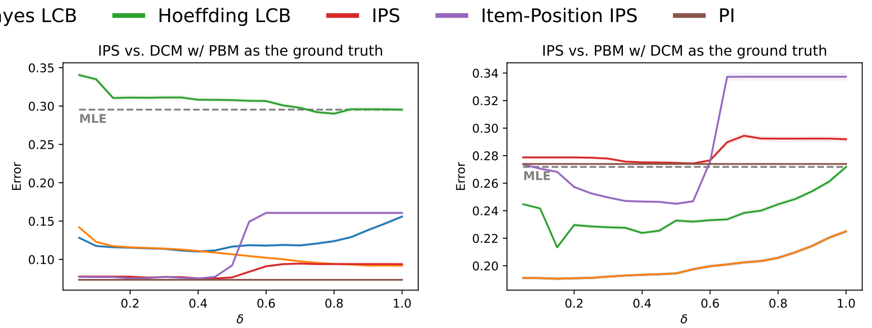


Figure 2: Comparison of our methods to MLE when increasing the sample size n.

Figure 3: Robustness evaluation of our methods and baselines.

the list value is linear, the baselines can perform similarly to the LCB estimators. We observe that the empirical estimation of the prior improves upon an uninformative $\mathrm{Beta}(1, 1)$ prior.

More Realistic Comparison to MLE: MLE is common in practice and does not have a hyper-parameter δ to tune, unlike our method. To show that our approach can beat the MLE in a realistic setting, we estimate δ on past data and then apply it to future data. We apply the evaluation protocol from the *Top 10 Queries* experiment on the first 5 days of data with fixed sample size $n = 1000$ for each query. We choose δ that minimizes the Bayesian LCB error. We apply the evaluation protocol from the *Top 10 Queries* experiment on the last 23 days of data with the above chosen $\delta = 0.2$. We report the difference between MLE and Bayesian LCB errors. This is repeated 500 times while varying logged sample size $n \in [50, 500\,000]$. Figure 2 shows that the largest improvements are at the sample size 50 000. This implies that LCBs have a *sweet spot* where they work the best. We observe smaller improvements for smaller sample sizes because the uncertainty is too high to leverage. On the other hand, when the sample size is large, the uncertainty is low everywhere and does not have to be modeled.

Robustness to Model Misspecification: We examine how the estimators behave if the underlying model does not hold. In the *Top 10 Queries* experiment, we observe that the baselines with linearity assumptions do not perform well in non-linear models, such as CM or DCM, but they perform well when the value of a list is the sum of clicks, such as PBM. We fit PBM and use it to generate the logged dataset. We then use DCM to learn the attraction probabilities for MLE and LCB methods. We also examine the opposite scenario, using DCM as a ground truth model and estimating attraction probabilities with PBM. This does not impact IPS and PI baselines. Our results are reported in Figure 3. In the left plot, we use a non-linear model to fit the linear reward. As a result, MLE and LCB methods estimate misspecified parameters. Other baselines that assume linearity perform better in this setup. However, Bayesian LCBs still achieve 50% lower error compared to MLE. In the right plot, the reward is non-linear, and all methods (except IPS) assume linearity in item-level rewards. Here, MLE performs on par with other baselines, and LCBs consistently outperform all other baselines. This shows us that LCBs are robust to model misspecification and can be used to improve model-based estimates further, even though we may not know the correct model class.

8.4 *Results on Other Datasets*

We validated the results on other popular datasets, namely Yahoo! Webscope[3], Istella[4], and MSLR-WEB[5]. These datasets do not contain clicks, only human-labeled query-document relevance scores; with $\mathrm{score}(a) \in \{0, 1, 2, 3, 4\}$ for item a ranked from 0 (not relevant) to 4 (highly relevant). We follow the standard procedure to generate the clicks by mapping relevance scores to attraction probabilities based on the *navigational* user model [Table 2, 9].

$\mathrm{score}(a)$	0	1	2	3	4
θ_a	0.05	0.1	0.2	0.4	0.8

For PBM, we define position examination probabilities based on an eye-tracking experiment [18] and for CDM, we define λ parameters as $\lambda_k = 1 - \exp(-k + 0.5)/0.5$ for positions $k \in [K]$. We randomly select 5000 queries. Then for each query, to form our logged

[3] http://webscope.sandbox.yahoo.com/
[4] https://istella.ai/data/
[5] https://www.microsoft.com/en-us/research/project/mslr/

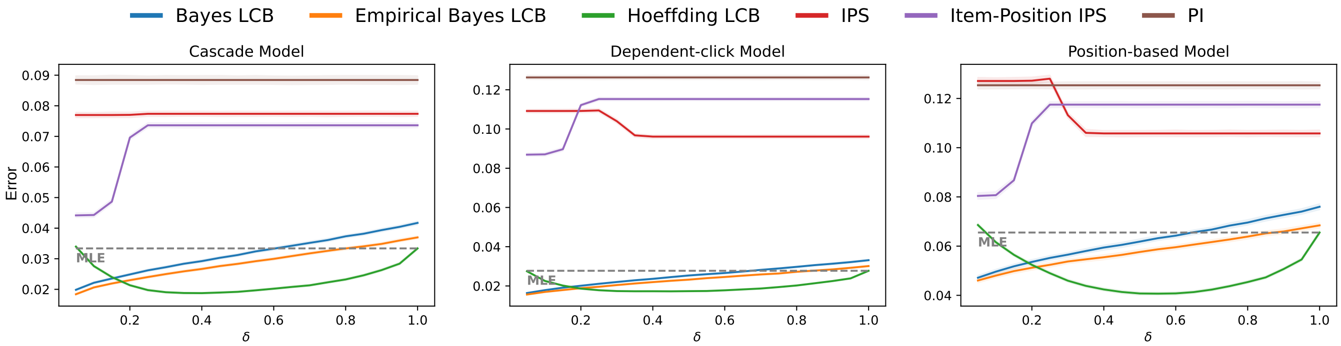


Figure 4: Comparison of our methods to baselines on the Yahoo! Webscope dataset.

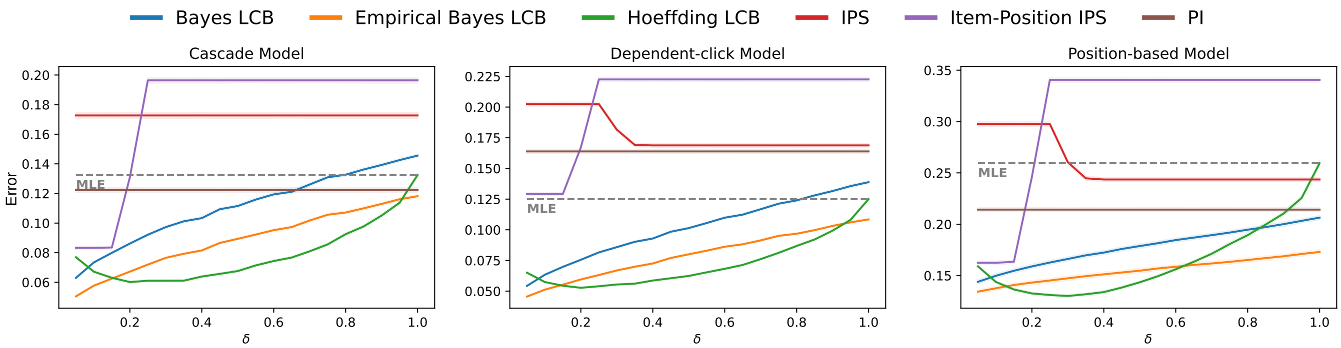


Figure 5: Comparison of our methods to baselines on the MSLR-WEB30k dataset.

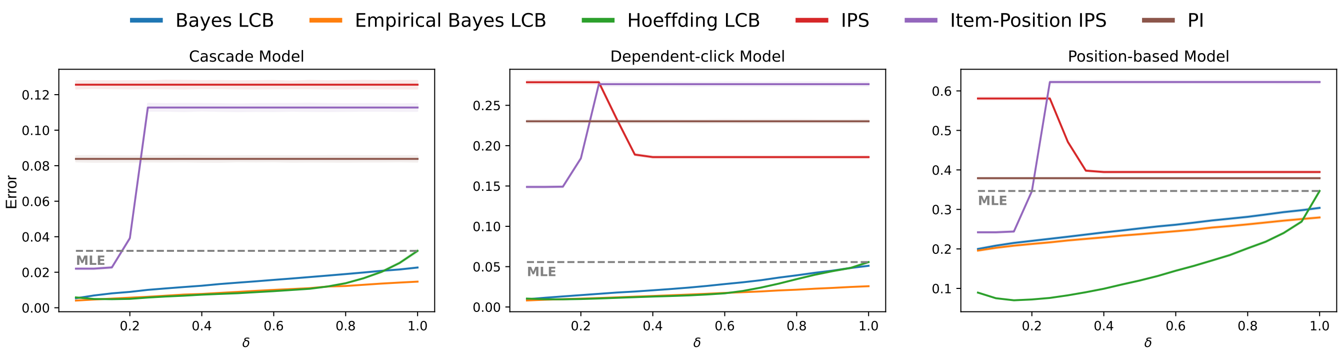


Figure 6: Comparison of our methods to baselines on the Istella dataset.

dataset, we sample 100 lists of length $K = 4$ from the set of labeled documents for given query $\mathcal{E}_q$ from Dirichlet distribution with its parameter set at $\alpha = (\theta_a)_{a \in \mathcal{E}_q}$. On these datasets, we use the same evaluation protocol as outlined in Section 8.1. Figures 4 to 6 support our prior findings. LCBs outperform MLE and other baselines for most δ values and provide major improvements. We observed similar results for other sample sizes and list lengths.

9 Conclusions

We study for the first time pessimistic off-policy optimization for learning to rank. We design lower confidence bounds (LCBs) for the value of ranked lists. Specifically, through LCBs on individual items, we get LCBs on exponentially large action spaces. We prove that the loss due to choosing the best list under our model is polynomial in the number of items in a list as opposed to polynomial in the number of lists, which is exponential. We also apply LCBs to non-linear objectives, such as CM and DCM in Equations (3) and (4), based on their linearization. This is the first paper in off-policy learning where this approximation was applied and analyzed. Our approach outperforms model-based approaches using maximum likelihood estimates (MLE) and optimizers using IPS. Furthermore, we show LCBs are robust to model misspecification and perform better with almost any confidence interval width. We show how to estimate prior with empirical Bayes when prior is not known in advance. Finally, LCBs proved to have a positive impact on almost any size of logged data.

One of the natural future directions is to apply listwise pessimism to any model class. This can be generally achieved with an ensemble of models, each trained on a different bootstrapped dataset, although this method presents computational challenges, and further research is needed so the optimization is tractable. We do not use theory-suggested confidence intervals in the experiments because they are too conservative. To address this limitation, studying the calibration of confidence interval width from logged data is needed. The main focus of our work is on a large action space, the space of all lists. We wanted to make this contribution clear and thus focus on tabular contexts. However, our algorithms can be extended to a large context space or items with features.

References

[1] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[2] L. Bottou, J. Peters, J. Quiñonero-Candela, D. X. Charles, D. M. Chickering, E. Portugaly, D. Ray, P. Simard, and E. Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research*, 14(11), 2013.

[3] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 621–630, 2009.

[4] A. Chuklin, I. Markov, and M. d. Rijke. Click models for web search. *Synthesis lectures on information concepts, retrieval, and services*, 7 (3):1–115, 2015.

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

[6] A. Gilotte, C. Calauzènes, T. Nedelec, A. Abraham, and S. Dollé. Offline a/b testing for recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 198–206, 2018.

[7] F. Guo, C. Liu, and Y. M. Wang. Efficient multiple-click models in web search. In *Proceedings of the 2nd ACM International Conference on Web Search and Data Mining*, pages 124–131, 2009.

[8] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, et al. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the eighth international workshop on data mining for online advertising*, pages 1–9, 2014.

[9] K. Hofmann, S. Whiteson, and M. D. Rijke. Fidelity, soundness, and efficiency of interleaved comparison methods. *ACM Transactions on Information Systems (TOIS)*, 31(4):1–43, 2013.

[10] J. Hong, B. Kveton, M. Zaheer, Y. Chow, and A. Ahmed. Non-stationary off-policy optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 2494–2502. PMLR, 2021.

[11] D. G. Horvitz and D. J. Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association*, 47(260):663–685, 1952.

[12] E. L. Ionides. Truncated importance sampling. *Journal of Computational and Graphical Statistics*, 17(2):295–311, 2008.

[13] O. Jeunen and B. Goethals. Pessimistic reward models for off-policy learning in recommendation. In *Fifteenth ACM Conference on Recommender Systems*, pages 63–74, 2021.

[14] O. Jeunen, D. Rohde, F. Vasile, and M. Bompaire. Joint policy-value learning for recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1223–1233, 2020.

[15] Y. Jin, Z. Yang, and Z. Wang. Is pessimism provably efficient for offline rl? In *International Conference on Machine Learning*, pages 5084–5096. PMLR, 2021.

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

[17] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems (TOIS)*, 25(2):7–es, 2007.

[18] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Acm Sigir Forum*, volume 51, pages 4–11. Acm New York, NY, USA, 2017.

[19] T. Joachims, A. Swaminathan, and T. Schnabel. Unbiased learning-to-rank with biased feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 781–789, 2017.

[20] T. Joachims, A. Swaminathan, and M. De Rijke. Deep learning with logged bandit feedback. In *International Conference on Learning Representations*, 2018.

[21] S. Katariya, B. Kveton, C. Szepesvari, and Z. Wen. Dcm bandits: Learning to rank with multiple clicks. In *International Conference on Machine Learning*, pages 1215–1224. PMLR, 2016.

[22] H. Kiyohara, Y. Saito, T. Matsuhiro, Y. Narita, N. Shimizu, and Y. Yamamoto. Doubly robust off-policy evaluation for ranking policies under the cascade behavior model. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 487–497, 2022.

[23] B. Kveton, C. Szepesvari, Z. Wen, and A. Ashkan. Cascading bandits: Learning to rank in the cascade model. In *International conference on machine learning*, pages 767–776. PMLR, 2015.

[24] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

[25] S. Li, Y. Abbasi-Yadkori, B. Kveton, S. Muthukrishnan, V. Vinay, and Z. Wen. Offline evaluation of ranking policies with click models. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1685–1694, 2018.

[26] J. S. Maritz and T. Lwin. *Empirical Bayes Methods*. Chapman and Hall/CRC, 2018.

[27] J. McInerney, B. Lacker, S. Hansen, K. Higley, H. Bouchard, A. Gruson, and R. Mehrotra. Explore, exploit, and explain: personalizing explainable recommendations with bandits. In *Proceedings of the 12th ACM conference on recommender systems*, pages 31–39, 2018.

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

[29] J. M. Robins, A. Rotnitzky, and L. P. Zhao. Estimation of regression coefficients when some regressors are not always observed. *Journal of the American statistical Association*, 89(427):846–866, 1994.

[30] J. E. Smith and R. L. Winkler. The optimizer's curse: Skepticism and postdecision surprise in decision analysis. *Management Science*, 52(3): 311–322, 2006.

[31] A. Strehl, J. Langford, L. Li, and S. M. Kakade. Learning from logged implicit exploration data. *Advances in neural information processing systems*, 23, 2010.

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

[33] A. Swaminathan and T. Joachims. The self-normalized estimator for counterfactual learning. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[34] A. Swaminathan, A. Krishnamurthy, A. Agarwal, M. Dudik, J. Langford, D. Jose, and I. Zitouni. Off-policy evaluation for slate recommendation. *Advances in Neural Information Processing Systems*, 30, 2017.

[35] G. Takács and D. Tikk. Alternating least squares for personalized ranking. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 83–90, 2012.

[36] R. Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[37] T. Xie, C.-A. Cheng, N. Jiang, P. Mineiro, and A. Agarwal. Bellman-consistent pessimism for offline reinforcement learning. *Advances in neural information processing systems*, 34, 2021.

[38] Yandex. Yandex personalized web search challenge. https://www.kaggle.com/c/yandex-personalized-web-search-challenge, 2013.

A Learning Click Model Parameters

To simplify notation, we only consider a small finite number of contexts, such as day of the week or user characteristics. For each such context, we estimate the attraction probabilities separately. In this section, we show the calculation of model parameters with respective applications of LCB to the three click models mentioned in Section 5. We assume that the context is fixed, and computations are done over each context separately; therefore, we define $\mathcal{T}_X$ to be the set of all indices of $[n]$ such that $X_t = X,\ \forall t \in \mathcal{T}_X$.

Cascade Model: The cascade model has only one type of parameters, attraction probabilities θ_a and these can be estimated from clicks, as the number of clicks over the number of examinations [23]. According to the cascade model assumptions, items are examined from the top until clicked on some item and the user does not examine any further. Therefore to model $\theta_{a,X}$, we collect the number of positive impressions $n_{a,X,+} = \sum_{t \in \mathcal{T}_X} \sum_{k=1}^{K} \mathbb{1}\{a_{t,k} = a \wedge Y_{t,k} = 1\}$ (user examined and clicked) and the number of negative impressions $n_{a,X,-} = \sum_{t \in \mathcal{T}_X} \sum_{k=1}^{K} \mathbb{1}\{a_{t,k} = a\} \mathbb{1}\left\{\sum_{j=1}^{k-1} Y_{t,j} = 0\right\}$ (examined, but did not click) for item a in context X and calculate either Bayesian, frequentist LCBs, and prior according to (6), (7), and Section 6.3.

Dependent-Click Model: When fitting the dependent-click model, we process the logged data according to the following assumption; examining items from the top until the final observed click at the lowest position and disregarding all items below. Observed impressions on each item a in context X are sampled from its unknown Bernoulli distribution $\mathrm{Ber}(\theta_{a,X})$. To model $\theta_{a,X}$, we collect the number of positive impressions $n_{a,X,+} = \sum_{t \in \mathcal{T}_X} \sum_{k=1}^{K} \mathbb{1}\{a_{t,k} = a\} Y_{t,k}$ (clicks) and the number of negative impressions $n_{a,X,-} = \sum_{t \in \mathcal{T}_X} \sum_{k=1}^{K} \mathbb{1}\{a_{t,k} = a\} (1 - Y_{t,k})$ (examined, but not clicked) for item a and calculate either Bayesian, frequentist LCBs, and prior according to (6), (7), and Section 6.3. To model the probability $\lambda_{k,X}$, we collect positive observations that the click is the last $n_{k,X,+} = \sum_{t \in \mathcal{T}_X} \mathbb{1}\left\{\sum_{j=k}^{K} Y_{t,j} = 1\right\} Y_{t,k}$ and negative observations that user continues exploring as $n_{k,X,-} = \sum_{t \in \mathcal{T}_X} \mathbb{1}\left\{\sum_{j=k}^{K} Y_{t,j} > 1\right\} Y_{t,k}$.

Position-Based Model: To learn the parameters of the position-based model, we use an EM algorithm. We solve $\min_{\theta,p} \sum_{t=1}^{n} \sum_{k=1}^{K} (\theta_{a_{t,k},X_t} p_k - Y_{t,k})^2$ by alternating least squares algorithm [35] to obtain an estimate of p. Then the propensity-weighted number of positive impressions is $n_{a,X,+} = \sum_{t \in \mathcal{T}_X} \sum_{k=1}^{K} \mathbb{1}\{a_{t,k} = a \wedge Y_{t,k} = 1\} / p_k$ and the number of negative impressions $n_{a,X,-} = \sum_{t \in \mathcal{T}_X} \sum_{k=1}^{K} \mathbb{1}\{a_{t,k} = a \wedge Y_{t,k} = 0\} / p_k$.

B Proofs of Pessimistic Optimization

Proof of Lemma 2:
PBM: By Lemma 1, for any item $a \in \mathcal{E}$ and context $X \in \mathcal{X}$, we have that

$$|\theta_{a,X} - \hat{\theta}_{a,X}| \leq \sqrt{\log(1/\delta)/(2n_{a,X})}$$

holds with probability at least $1 - \delta$. Therefore, by the union bound, we have that

$$|\theta_{a,X} - \hat{\theta}_{a,X}| \leq \sqrt{\log(|\mathcal{E}|\,|\mathcal{X}|\,/\delta)/(2n_{a,X})} \tag{12}$$

holds with probability at least $1 - \delta$, jointly over all items a and contexts X.

Now we prove our main claim. For any context X and list $A = (a_k)_{k \in [K]}$, we have from the definition of the PBM that

$$V_{\text{PBM}}(A, X) - \hat{V}_{\text{PBM}}(A, X) = \sum_{k=1}^{K} p_{k,X} (\theta_{a_k,X} - \hat{\theta}_{a_k,X})\,.$$

Since $p_{k,X} \in [0, 1]$, we have

$$\left| V_{\text{PBM}}(A, X) - \hat{V}_{\text{PBM}}(A, X) \right| \leq \sum_{k=1}^{K} |\theta_{a_k,X} - \hat{\theta}_{a_k,X}| = \sum_{a \in A} |\theta_{a,X} - \hat{\theta}_{a,X}| \leq \sum_{a \in A} \sqrt{\log(|\mathcal{E}|\,|\mathcal{X}|\,/\delta)/(2n_{a,X})}\,. \tag{13}$$

The last inequality is under the assumption that (12) holds, which holds with probability at least $1 - \delta$.

CM: To prove the bound for CM and DCM, we first show how the difference of two products with K variables is bounded by the difference of their sums.

Lemma 4. *Let $(a_k)_{k=1}^{K} \in [0, 1]^K$ and $(b_k)_{k=1}^{K} \in [0, 1]^K$. Then*

$$\left| \prod_{k=1}^{K} a_k - \prod_{k=1}^{K} b_k \right| \leq \sum_{k=1}^{K} |a_k - b_k|\,.$$

Proof. We start with

$$\prod_{k=1}^{K} a_k - \prod_{k=1}^{K} b_k = \prod_{k=1}^{K} a_k - a_1 \prod_{k=2}^{K} b_k + a_1 \prod_{k=2}^{K} b_k - \prod_{k=1}^{K} b_k = a_1 \left(\prod_{k=2}^{K} a_k - \prod_{k=2}^{K} b_k \right) + (a_1 - b_1) \prod_{k=2}^{K} b_k$$
$$\stackrel{(a)}{=} \sum_{k=1}^{K} \left(\prod_{i=1}^{k-1} a_i \right) (a_k - b_k) \left(\prod_{i=k+1}^{K} b_i \right) ,$$

where (a) is by recursively applying the same argument to $\prod_{k=2}^{K} a_k - \prod_{k=2}^{K} b_k$. Now we apply the absolute value and get

$$\left| \prod_{k=1}^{K} a_k - \prod_{k=1}^{K} b_k \right| = \left| \sum_{k=1}^{K} \left(\prod_{i=1}^{k-1} a_i \right) (a_k - b_k) \left(\prod_{i=k+1}^{K} b_i \right) \right| \leq \sum_{k=1}^{K} |a_k - b_k| ,$$

since $\prod_{i=1}^{k-1} a_i \in [0,1]$ and $\prod_{i=k+1}^{K} b_i \in [0,1]$.

□

Now we prove our main claim. For any context X and list $A = (a_k)_{k \in [K]}$, we have from the definition of the CM and from the bound in Lemma 4 that

$$\left| V_{\text{CM}}(A, X) - \hat{V}_{\text{CM}}(A, X) \right| = \left| 1 - \prod_{k=1}^{K} (1 - \theta_{a_k, X}) - 1 + \prod_{k=1}^{K} (1 - \hat{\theta}_{a_k, X}) \right| = \left| \prod_{k=1}^{K} (1 - \hat{\theta}_{a_k, X}) - \prod_{k=1}^{K} (1 - \theta_{a_k, X}) \right|$$
$$\leq \sum_{k=1}^{K} |\theta_{a_k, X} - \hat{\theta}_{a_k, X}| = \sum_{a \in A} |\theta_{a, X} - \hat{\theta}_{a, X}| \leq \sum_{a \in A} \sqrt{\log(|\mathcal{E}| \, |\mathcal{X}| / \delta) / (2 n_{a, X})} \, .$$

The last inequality is under the assumption that (12) holds, which holds with probability at least $1 - \delta$.

DCM: For any context X and list $A = (a_k)_{k \in [K]}$, we have from the definition of the DCM and from the bound in Lemma 4 that

$$\begin{aligned} \left| V_{\text{DCM}}(A, X) - \hat{V}_{\text{DCM}}(A, X) \right| &= \left| 1 - \prod_{k=1}^{K} (1 - (1 - \lambda_{k, X}) \theta_{a_k, X}) - 1 + \prod_{k=1}^{K} (1 - (1 - \lambda_{k, X}) \hat{\theta}_{a_k, X}) \right| \\ &= \left| \prod_{k=1}^{K} (1 - (1 - \lambda_{k, X}) \hat{\theta}_{a_k, X}) - \prod_{k=1}^{K} (1 - (1 - \lambda_{k, X}) \theta_{a_k, X}) \right| \\ &\leq \sum_{k=1}^{K} |\theta_{a_k, X} - \hat{\theta}_{a_k, X}| = \sum_{a \in A} |\theta_{a, X} - \hat{\theta}_{a, X}| \leq \sum_{a \in A} \sqrt{\log(|\mathcal{E}| \, |\mathcal{X}| / \delta) / (2 n_{a, X})} \, . \end{aligned}$$

The first inequality holds as $\lambda_{k, X} \in [0, 1]$, and the last inequality is under the assumption that (12) holds, which holds with probability at least $1 - \delta$. This completes the proof.

Proof of Theorem 3:

Proof. Let $L(A, X) = \hat{V}(A, X) - c(A, X)$. Suppose that $\left| \hat{V}(A, X) - V(A, X) \right| \leq c(A, X)$ holds for all A and X. Note that this also implies $L(A, X) \leq V(A, X)$. Then, for any X,

$$\begin{aligned} V(A_{*, X}, X) - V(\hat{A}_X, X) &= V(A_{*, X}, X) - L(A_{*, X}, X) + L(A_{*, X}, X) - V(\hat{A}_X, X) \\ &\leq V(A_{*, X}, X) - L(A_{*, X}, X) + L(\hat{A}_X, X) - V(\hat{A}_X, X) \\ &\leq V(A_{*, X}, X) - L(A_{*, X}, X) \\ &= V(A_{*, X}, X) - \hat{V}(A_{*, X}, X) + c(A_{*, X}, X) \\ &\leq 2 c(A_{*, X}, X) \, . \end{aligned}$$

The first inequality holds because $\hat{A}_X$ maximizes $L(\cdot, X)$. The second inequality holds because $L(\hat{A}_X, X) \leq V(\hat{A}_X, X)$. The last inequality holds because $V(A_{*, X}, X) - \hat{V}(A_{*, X}, X) \leq c(A_{*, X}, X)$.

It remains to prove that $\left| \hat{V}(A, X) - V(A, X) \right| \leq c(A, X)$ holds for all A and X. By Lemma 2, this occurs with probability at least $1 - \delta$, jointly over all A and X, for $c(A, X)$ in Lemma 2. This completes the proof. □

C Other Experiments

Most Frequent Query: In Figure 7 we show the simplest case when using only the most frequent query to observe the effect of LCBs without possible skewing due to averaging over multiple queries. Results found in the most frequent query support those already discussed in *Top 10 Queries*.

Less Frequent Queries: We study how less frequent queries impact the performance of LCBs. We use the same setup as in the *Top 10 Queries* experiment and observe how the error changes as the number of queries increases. In Figure 8, we show comparable improvements to those of DCM in Figure 1, showing that LCBs perform well even in less frequent queries. We performed this experiment with CM and PBM as well and observed a similar behavior.

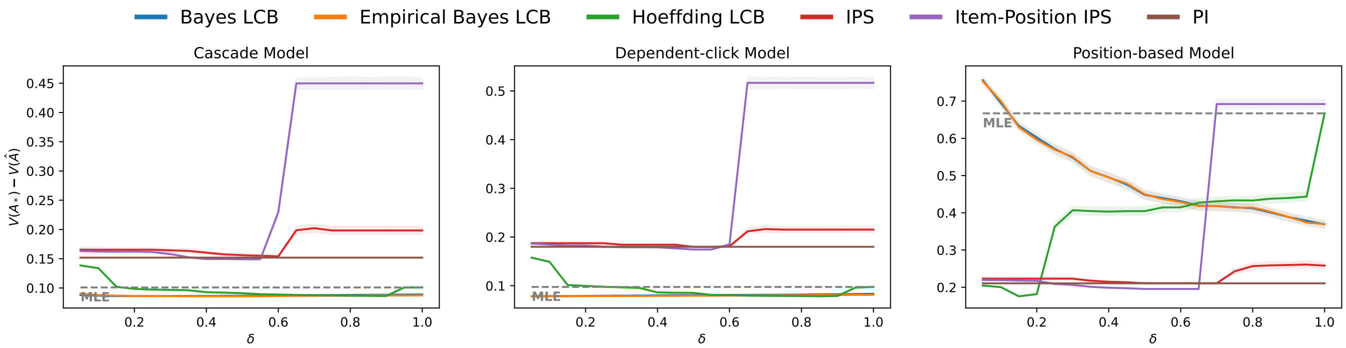


Figure 7: Error of the estimated lists $\hat{A}$ compared to optimal lists A_* on the most frequent query.

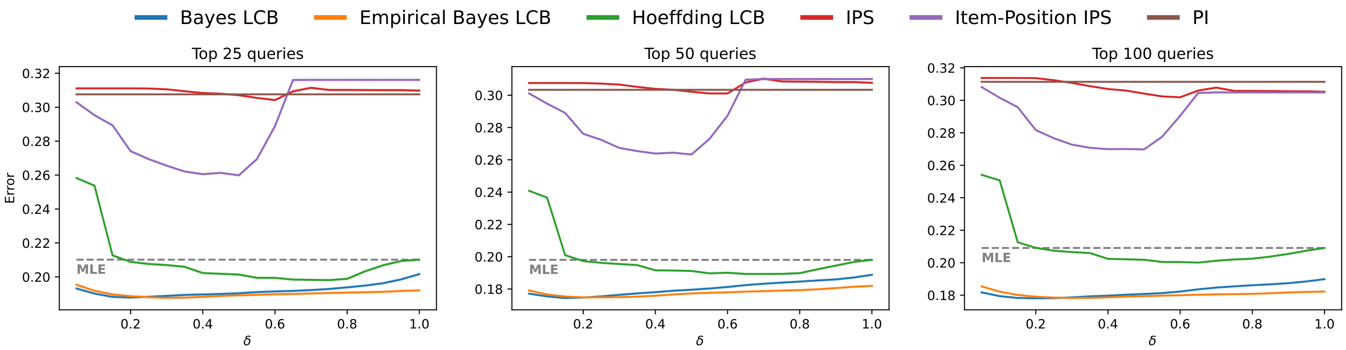


Figure 8: Comparison of our methods to baselines on DCM and less frequent queries.

Cross-Validation Setting: In all of the previous experiments, we fit a model on logged data $\mathcal{D}$, use that model to generate new clicks, and then learn a new model using those clicks. One can argue that data generated under right model yields better results and under a wrong model LCBs do not hold. Therefore, we split logged data $\mathcal{D}$ at the 23rd day to train and test set and use first 23 days to train model parameters. We randomly select 1000 samples for each query from the remaining 4 days to fit models using LCB while varying δ. The rest of the setting is the same as in the *Top 10 Queries* experiment. The reason for this split is that using the remaining four days provides enough data in the top 10 queries to sample from. We tested other split ratios and observed results do not differ significantly. In Figure 9, we observe the Bayesian LCBs outperform all other baselines, and reasoning under uncertainty is still better than using MLE. Similarly to the previous experiments, we omit other linear baselines in CM and DCM plots as they perform considerably worse.

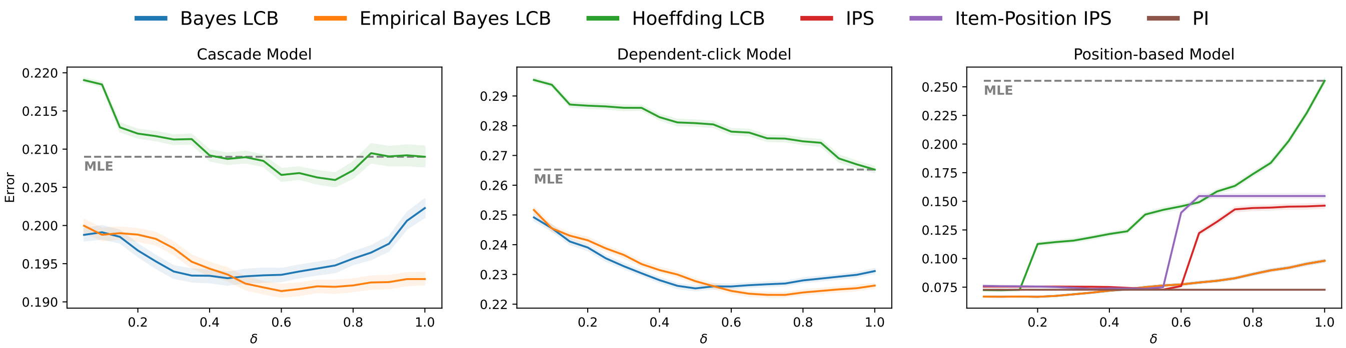


Figure 9: Comparison of our methods against other baselines in setting, where the fitted ground truth model does not generate clicks.

Finding Prior: When we estimated prior in the previous experiments, we did so on grid $(\alpha, \beta) \in \mathcal{G}^2,\ \mathcal{G} = \{2^i\}_{i=1}^{10}$ and we observed significant improvements over fixed prior $\mathrm{Beta}(1,1)$ mostly when $\delta = 1$. In this experiment, the goal is to measure an improvement with the increasing grid size. We evaluate DCM using Bayesian LCBs and keep the rest of the setting the same as in the *Top 10 Queries* experiment. We then estimate prior from these queries by searching over grid $\mathcal{G}^2,\ \mathcal{G} = \{2^{i-1}\}_{i=1}^{m},\ m \in [1, 2, 5, 10, 20]$. In Figure 10, we see how the results get more robust against the δ hyperparameter until $|\mathcal{G}| = 10$, and after that, the method still finds the same optimal prior. Therefore in our case, it is sufficient to search over $\mathcal{G} = \{2^i\}_{i=1}^{10}$. The optimal values found on $\mathcal{G}$ are $\theta \sim \mathrm{Beta}(1, 8)$ and $\lambda \sim \mathrm{Beta}(1, 64)$.

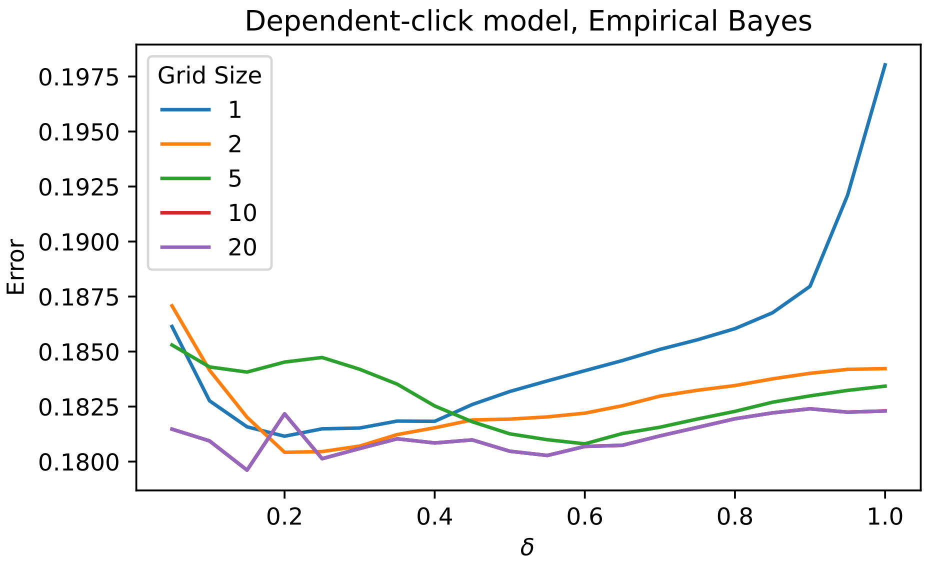


Figure 10: Empirical Bayes showing increasing performance with larger grid size. Red and purple lines are overlapping.