Online Learning to Rank in Stochastic Click Models

Masrour Zoghi [1] **Tomas Tunys** [2] **Mohammad Ghavamzadeh** [3] **Branislav Kveton** [4] **Csaba Szepesvari** [5] **Zheng Wen** [4]

Abstract

Online learning to rank is a core problem in information retrieval and machine learning. Many provably efficient algorithms have been recently proposed for this problem in specific click models. The click model is a model of how the user interacts with a list of documents. Though these results are significant, their impact on practice is limited, because all proposed algorithms are designed for specific click models and lack convergence guarantees in other models. In this work, we propose `BatchRank`, the first online learning to rank algorithm for a broad class of click models. The class encompasses two most fundamental click models, the cascade and position-based models. We derive a gap-dependent upper bound on the T-step regret of `BatchRank` and evaluate it on a range of web search queries. We observe that `BatchRank` outperforms ranked bandits and is more robust than `CascadeKL-UCB`, an existing algorithm for the cascade model.

1. Introduction

Learning to rank (LTR) is a core problem in information retrieval (Liu, 2011) and machine learning; with numerous applications in web search, recommender systems and ad placement. The goal of LTR is to present a list of K documents out of L that maximizes the satisfaction of the user. This problem has been traditionally solved by training supervised learning models on manually annotated relevance judgments. However, strong evidence suggests (Agichtein et al., 2006; Zoghi et al., 2016) that the feedback of users, that is clicks, can lead to major improvements over supervised LTR methods. In addition, billions of users interact daily with commercial LTR systems, and it is finally feasible to interactively and adaptive maximize the satisfaction of these users from clicks.

These observations motivated numerous papers on online LTR methods, which utilize user feedback to improve the quality of ranked lists. These methods can be divided into two groups: learning the best ranker in a family of rankers (Yue & Joachims, 2009; Hofmann et al., 2013), and learning the best list under some model of user interaction with the list (Radlinski et al., 2008a; Slivkins et al., 2013), such as a *click model* (Chuklin et al., 2015). The click model is a stochastic model of how the user examines and clicks on a list of documents. In this work, we focus on online LTR in click models and address a shortcoming of all past work on this topic.

More precisely, many algorithms have been proposed and analyzed for finding the optimal ranked list in the cascade model (CM) (Kveton et al., 2015a; Combes et al., 2015; Kveton et al., 2015b; Zong et al., 2016; Li et al., 2016), the dependent-click model (DCM) (Katariya et al., 2016), and the position-based model (PBM) (Lagree et al., 2016). The problem is that if the user interacts with ranked lists using a different click model, the theoretical guarantees cease to hold. Then, as we show empirically, these algorithms may converge to suboptimal solutions. This is a grave issue because it is well known that no single click model captures the behavior of an entire population of users (Grotov et al., 2015). Therefore, it is critical to develop efficient learning algorithms for multiple click models, which is the aim of this paper.

We make the following contributions:

- We propose *stochastic click bandits*, a learning framework for maximizing the expected number of clicks in online LTR in a broad class of click models, which includes both the PBM (Richardson et al., 2007) and CM (Craswell et al., 2008).
- We propose the first algorithm, `BatchRank`, that is guaranteed to learn the optimal solution in a diverse class of click models. This is of a great practical significance, as

[1]Independent Researcher, Vancouver, BC, Canada (Part of this work was done during an internship at Adobe Research) [2]Czech Technical University, Prague, Czech Republic [3]DeepMind, Mountain View, CA, USA (This work was done while the author was at Adobe Research) [4]Adobe Research, San Jose, CA, USA [5]University of Alberta, Edmonton, AB, Canada. Correspondence to: Branislav Kveton <kveton@adobe.com>, Masrour Zoghi <masrour@zoghi.org>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

it is often difficult or impossible to guess the underlying click model in advance.

- We prove a gap-dependent upper bound on the regret of `BatchRank` that scales well with all quantities of interest. The key step in our analysis is a KL scaling lemma (Section 5.4), which should be of a broader interest.
- We evaluate `BatchRank` on both CM and PBM queries. Our results show that `BatchRank` performs significantly better than `RankedExp3` (Radlinski et al., 2008a), an adversarial online LTR algorithm; and is more robust than `CascadeKL-UCB` (Kveton et al., 2015a), an optimal online LTR algorithm for the CM.

We define $[n] = \{1, \ldots, n\}$. For any sets A and B, we denote by A^B the set of all vectors whose entries are indexed by B and take values from A. We use boldface letters to denote important random variables.

2. Background

This section reviews two fundamental *click models* (Chuklin et al., 2015), models of how users click on an ordered list of K documents. The universe of all documents is represented by *ground set* $\mathcal{D} = [L]$ and we refer to the documents in $\mathcal{D}$ as *items*. The user is presented a *ranked list*, an ordered list of K documents out of L. We denote this list by $\mathcal{R} = (d_1, \ldots, d_K) \in \Pi_K(\mathcal{D})$, where $\Pi_K(\mathcal{D}) \subset \mathcal{D}^K$ is the set of all K-tuples with distinct elements from $\mathcal{D}$ and d_k is the k-th item in $\mathcal{R}$. We assume that the click model is parameterized by L *item-dependent attraction probabilities* $\alpha \in [0, 1]^L$, where $\alpha(d)$ is the probability that item d is *attractive*. The items attract the user independently. For simplicity and without loss of generality, we assume that $\alpha(1) \geq \ldots \geq \alpha(L)$. The reviewed models differ in how the user examines items, which leads to clicks.

2.1. Position-Based Model

The *position-based model (PBM)* (Richardson et al., 2007) is a model where the probability of clicking on an item depends on both its identity and position. Therefore, in addition to item-dependent attraction probabilities, the PBM is parameterized by K *position-dependent examination probabilities* $\chi \in [0, 1]^K$, where $\chi(k)$ is the examination probability of position k.

The user interacts with a list of items $\mathcal{R} = (d_1, \ldots, d_K)$ as follows. The user *examines* position $k \in [K]$ with probability $\chi(k)$ and then *clicks* on item d_k at that position with probability $\alpha(d_k)$. Thus, the expected number of clicks on list $\mathcal{R}$ is

$$r(\mathcal{R}) = \sum_{k=1}^{K} \chi(k) \alpha(d_k) \,.$$

In practice, it is often observed that $\chi(1) \geq \ldots \geq \chi(K)$ (Chuklin et al., 2015), and we adopt this assumption in this work. Under this assumption, the above function is maximized by the list of K most attractive items

$$\mathcal{R}^* = (1, \ldots, K) \,, \tag{1}$$

where the k-th most attractive item is placed at position k.

In this paper, we focus on the objective of maximizing the number of clicks. We note that the satisfaction of the user may not increase with the number of clicks, and that other objectives have been proposed in the literature (Radlinski et al., 2008b). The shortcoming of all of these objectives is that none directly measure the satisfaction of the user.

2.2. Cascade Model

In the *cascade model (CM)* (Craswell et al., 2008), the user scans a list of items $\mathcal{R} = (d_1, \ldots, d_K)$ from the first item d_1 to the last d_K. If item d_k is *attractive*, the user *clicks* on it and does not examine the remaining items. If item d_k is not attractive, the user *examines* item d_{k+1}. The first item d_1 is examined with probability one.

From the definition of the model, the probability that item d_k is examined is equal to the probability that none of the first $k - 1$ items are attractive. Since each item attracts the user independently, this probability is

$$\chi(\mathcal{R}, k) = \prod_{i=1}^{k-1} (1 - \alpha(d_i)) \,. \tag{2}$$

The expected number of clicks on list $\mathcal{R}$ is at most 1, and is equal to the probability of observing any click,

$$r(\mathcal{R}) = \sum_{k=1}^{K} \chi(\mathcal{R}, k) \alpha(d_k) = 1 - \prod_{k=1}^{K} (1 - \alpha(d_k)) \,.$$

This function is maximized by the list of K most attractive items $\mathcal{R}^*$ in (1), though any permutation of $[K]$ would be optimal in the CM. Note that the list $\mathcal{R}^*$ is optimal in both the PBM and CM.

3. Online Learning to Rank in Click Models

The PBM and CM (Section 2) are similar in many aspects. First, both models are parameterized by L item-dependent attraction probabilities. The items attract the user independently. Second, the probability of clicking on the item is a product of its attraction probability, which depends on the identity of the item; and the examination probability of its position, which is independent of the identity of the item. Finally, the optimal solution in both models is the list of K most attractive items $\mathcal{R}^*$ in (1), where the k-th most attractive item is placed at position k.

This suggests that the optimal solution in both models can be learned by a single learning algorithm, which does not know the underlying model. We propose this algorithm in Section 4. Before we discuss it, we formalize our learning problem as a *multi-armed bandit* (Auer et al., 2002; Lai & Robbins, 1985).

3.1. Stochastic Click Bandit

We refer to our learning problem as a *stochastic click bandit*. An instance of this problem is a tuple (K, L, P_α, P_χ), where K is the number of positions, L is the number of items, P_α is a distribution over binary vectors $\{0,1\}^L$, and P_χ is a distribution over binary matrices $\{0,1\}^{\Pi_K(\mathcal{D}) \times K}$.

The learning agent interacts with our problem as follows. Let $(\boldsymbol{A}_t, \boldsymbol{X}_t)_{t=1}^T$ be T i.i.d. random variables drawn from $P_\alpha \otimes P_\chi$, where $\boldsymbol{A}_t \in \{0,1\}^L$ and $\boldsymbol{A}_t(d)$ is the *attraction indicator* of item d at time t; and $\boldsymbol{X}_t \in \{0,1\}^{\Pi_K(\mathcal{D}) \times K}$ and $\boldsymbol{X}_t(\mathcal{R}, k)$ is the *examination indicator* of position k in list $\mathcal{R} \in \Pi_K(\mathcal{D})$ at time t. At time t, the agent chooses a list $\boldsymbol{\mathcal{R}}_t = (\boldsymbol{d}_1^t, \dots, \boldsymbol{d}_K^t) \in \Pi_K(\mathcal{D})$, which depends on past observations of the agent, and then observes *clicks*. These clicks are a function of $\boldsymbol{\mathcal{R}}_t$, $\boldsymbol{A}_t$, and $\boldsymbol{X}_t$. Let $\boldsymbol{c}_t \in \{0,1\}^K$ be the vector of *click indicators* on all positions at time t. Then

$$\boldsymbol{c}_t(k) = \boldsymbol{X}_t(\boldsymbol{\mathcal{R}}_t, k) \boldsymbol{A}_t(\boldsymbol{d}_k^t)$$

for any $k \in [K]$, the item at position k is clicked only if both $\boldsymbol{X}_t(\boldsymbol{\mathcal{R}}_t, k) = 1$ and $\boldsymbol{A}_t(\boldsymbol{d}_k^t) = 1$.

The goal of the learning agent is to maximize the number of clicks. Therefore, the number of clicks at time t is the *reward* of the agent at time t. We define it as

$$\boldsymbol{r}_t = \sum_{k=1}^K \boldsymbol{c}_t(k) = r(\boldsymbol{\mathcal{R}}_t, \boldsymbol{A}_t, \boldsymbol{X}_t)\,, \tag{3}$$

where $r : \Pi_K(\mathcal{D}) \times [0,1]^L \times [0,1]^{\Pi_K(\mathcal{D}) \times K} \to [0, K]$ is a *reward function*, which we define for any $\mathcal{R} \in \Pi_K(\mathcal{D})$, $A \in [0,1]^L$, and $X \in [0,1]^{\Pi_K(\mathcal{D}) \times K}$ as

$$r(\mathcal{R}, A, X) = \sum_{k=1}^K X(\mathcal{R}, k) A(d_k)\,.$$

We adopt the same independence assumptions as in Section 2. In particular, we assume that items attract the user independently.

Assumption 1. *For any $A \in \{0,1\}^L$,*

$$P(\boldsymbol{A}_t = A) = \textstyle\prod_{d \in \mathcal{D}} \mathrm{Ber}(A(d); \alpha(d))\,,$$

where $\mathrm{Ber}(\cdot; \theta)$ denotes the probability mass function of a Bernoulli distribution with mean $\theta \in [0,1]$, which we define as $\mathrm{Ber}(y; \theta) = \theta^y (1-\theta)^{1-y}$ for any $y \in \{0,1\}$.

Moreover, we assume that the attraction of any item is independent of the examination of its position.

Assumption 2. *For any list $\mathcal{R} \in \Pi_K(\mathcal{D})$ and position k,*

$$\mathbb{E}\left[\boldsymbol{c}_t(k) \,|\, \boldsymbol{\mathcal{R}}_t = \mathcal{R}\right] = \chi(\mathcal{R}, k) \alpha(d_k)\,,$$

where $\chi \in [0,1]^{\Pi_K(\mathcal{D}) \times K}$ and $\chi(\mathcal{R}, k) = \mathbb{E}\left[\boldsymbol{X}_t(\mathcal{R}, k)\right]$ is the examination probability *of position k in list $\mathcal{R}$.*

We do not make any independence assumptions among the entries of $\boldsymbol{X}_t$, and on other interactions of $\boldsymbol{A}_t$ and $\boldsymbol{X}_t$.

From our independence assumptions and the definition of the reward in (3), the *expected reward* of list $\mathcal{R}$ is

$$\mathbb{E}\left[r(\mathcal{R}, \boldsymbol{A}_t, \boldsymbol{X}_t)\right] = \sum_{k=1}^K \chi(\mathcal{R}, k) \alpha(d_k) = r(\mathcal{R}, \alpha, \chi)\,.$$

We evaluate the performance of a learning agent by its *expected cumulative regret*

$$R(T) = \mathbb{E}\left[\sum_{t=1}^T R(\boldsymbol{\mathcal{R}}_t, \boldsymbol{A}_t, \boldsymbol{X}_t)\right]\,,$$

where $R(\boldsymbol{\mathcal{R}}_t, \boldsymbol{A}_t, \boldsymbol{X}_t) = r(\mathcal{R}^*, \boldsymbol{A}_t, \boldsymbol{X}_t) - r(\boldsymbol{\mathcal{R}}_t, \boldsymbol{A}_t, \boldsymbol{X}_t)$ is the *instantaneous regret* of the agent at time t and

$$\mathcal{R}^* = \arg\max_{\mathcal{R} \in \Pi_K(\mathcal{D})} r(\mathcal{R}, \alpha, \chi)$$

is the *optimal list* of items, the list that maximizes the expected reward. To simplify exposition, we assume that the optimal solution, as a set, is unique.

3.2. Position Bandit

The learning variant of the PBM in Section 2.1 can be formulated in our setting when

$$\forall \mathcal{R}, \mathcal{R}' \in \Pi_K(\mathcal{D}):\ \boldsymbol{X}_t(\mathcal{R}, k) = \boldsymbol{X}_t(\mathcal{R}', k) \tag{4}$$

at any position $k \in [K]$. Under this assumption, the probability of clicking on item $\boldsymbol{d}_k^t$ at time t is

$$\mathbb{E}\left[\boldsymbol{c}_t(k) \,|\, \boldsymbol{\mathcal{R}}_t\right] = \chi(k) \alpha(\boldsymbol{d}_k^t)\,,$$

where $\chi(k)$ is defined in Section 2.1. The expected reward of list $\boldsymbol{\mathcal{R}}_t$ at time t is

$$\mathbb{E}\left[\boldsymbol{r}_t \,|\, \boldsymbol{\mathcal{R}}_t\right] = \sum_{k=1}^K \chi(k) \alpha(\boldsymbol{d}_k^t)\,.$$

3.3. Cascading Bandit

The learning variant of the CM in Section 2.2 can be formulated in our setting when

$$\boldsymbol{X}_t(\mathcal{R}, k) = \prod_{i=1}^{k-1} (1 - \boldsymbol{A}_t(d_i)) \tag{5}$$

for any list $\mathcal{R} \in \Pi_K(\mathcal{D})$ and position $k \in [K]$. Under this assumption, the probability of clicking on item $\boldsymbol{d}_k^t$ at time t is

$$\mathbb{E}\left[\boldsymbol{c}_t(k) \,|\, \boldsymbol{\mathcal{R}}_t\right] = \left[\prod_{i=1}^{k-1} (1 - \alpha(\boldsymbol{d}_i^t))\right] \alpha(\boldsymbol{d}_k^t) \,.$$

The expected reward of list $\boldsymbol{\mathcal{R}}_t$ at time t is

$$\mathbb{E}\left[\boldsymbol{r}_t \,|\, \boldsymbol{\mathcal{R}}_t\right] = \sum_{k=1}^{K} \left[\prod_{i=1}^{k-1} (1 - \alpha(\boldsymbol{d}_i^t))\right] \alpha(\boldsymbol{d}_k^t) \,.$$

3.4. Additional Assumptions

The above assumptions are not sufficient to guarantee that the optimal list $\mathcal{R}^*$ in (1) is learnable. Therefore, we make four additional assumptions, which are quite natural.

Assumption 3 (Order-independent examination)**.** *For any lists $\mathcal{R} \in \Pi_K(\mathcal{D})$ and $\mathcal{R}' \in \Pi_K(\mathcal{D})$, and position $k \in [K]$ such that $d_k = d'_k$ and $\{d_1, \ldots, d_{k-1}\} = \{d'_1, \ldots, d'_{k-1}\}$, $\boldsymbol{X}_t(\mathcal{R}, k) = \boldsymbol{X}_t(\mathcal{R}', k)$.*

The above assumption says that the examination indicator $\boldsymbol{X}_t(\mathcal{R}, k)$ only depends on the identities of $d_1, \ldots, d_{k-1}$. Both the CM and PBM satisfy this assumption, which can be validated from (4) and (5).

Assumption 4 (Decreasing examination)**.** *For any list $\mathcal{R} \in \Pi_K(\mathcal{D})$ and positions $1 \leq i \leq j \leq K$, $\chi(\mathcal{R}, i) \geq \chi(\mathcal{R}, j)$.*

The above assumption says that a lower position cannot be examined more than a higher position, in any list $\mathcal{R}$. Both the CM and PBM satisfy this assumption.

Assumption 5 (Correct examination scaling)**.** *For any list $\mathcal{R} \in \Pi_K(\mathcal{D})$ and positions $1 \leq i \leq j \leq K$, let $\alpha(d_i) \leq \alpha(d_j)$ and $\mathcal{R}' \in \Pi_K(\mathcal{D})$ be the same list as $\mathcal{R}$ except that d_i and d_j are exchanged. Then $\chi(\mathcal{R}, j) \geq \chi(\mathcal{R}', j)$.*

The above assumption says that the examination probability of a position cannot increase if the item at that position is swapped for a less-attractive higher-ranked item, in any list $\mathcal{R}$. Both the CM and PBM satisfy this assumption. In the CM, the inequality follows directly from the definition of examination in (2). In the PBM, $\chi(\mathcal{R}, j) = \chi(\mathcal{R}', j)$.

Assumption 6 (Optimal examination)**.** *For any list $\mathcal{R} \in \Pi_K(\mathcal{D})$ and position $k \in [K]$, $\chi(\mathcal{R}, k) \geq \chi(\mathcal{R}^*, k)$.*

This assumption says that any position k is least examined if the first $k-1$ items are optimal. Both the CM and PBM satisfy this assumption. In the CM, the inequality follows from the definition of examination in (2). In the PBM, we have that $\chi(\mathcal{R}, k) = \chi(\mathcal{R}^*, k)$.

4. Algorithm `BatchRank`

The design of `BatchRank` (Algorithm 1) builds on two key ideas. First, we *randomize* the placement of items to avoid biases due to the click model. Second, we *divide and conquer*; recursively divide the batches of items into more and less attractive items. The result is a sorted list of K items, where the k-th most attractive item is placed at position k.

Algorithm 1 `BatchRank`

// Initialization
for $b = 1, \ldots, 2K$ **do**
  for $\ell = 0, \ldots, T-1$ **do**
    for all $d \in \mathcal{D}$ **do**
      $\boldsymbol{c}_{b,\ell}(d) \leftarrow 0,\ \boldsymbol{n}_{b,\ell}(d) \leftarrow 0$
$\mathcal{A} \leftarrow \{1\},\ b_{\max} \leftarrow 1$
$\boldsymbol{I}_1 \leftarrow (1, K),\ \boldsymbol{B}_{1,0} \leftarrow \mathcal{D},\ \ell_1 \leftarrow 0$
for $t = 1, \ldots, T$ **do**
  for all $b \in \mathcal{A}$ **do**
    `DisplayBatch`(b, t)
  for all $b \in \mathcal{A}$ **do**
    `CollectClicks`(b, t)
  for all $b \in \mathcal{A}$ **do**
    `UpdateBatch`(b, t)

`BatchRank` explores items in batches, which are indexed by integers $b > 0$. A *batch* b is associated with the initial set of items $\boldsymbol{B}_{b,0} \subseteq \mathcal{D}$ and a range of *positions* $\boldsymbol{I}_b \in [K]^2$, where $\boldsymbol{I}_b(1)$ is the *highest position* in batch b, $\boldsymbol{I}_b(2)$ is the *lowest position* in batch b, and $\mathrm{len}(b) = \boldsymbol{I}_b(2) - \boldsymbol{I}_b(1) + 1$ is *number of positions* in batch b. The batch is explored in *stages*, which we index by integers $\ell > 0$. The remaining items in stage ℓ of batch b are $\boldsymbol{B}_{b,\ell} \subseteq \boldsymbol{B}_{b,0}$. The lengths of the stages quadruple. More precisely, any item $d \in \boldsymbol{B}_{b,\ell}$ in stage ℓ is explored n_ℓ times, where $n_\ell = \left\lceil 16 \tilde{\Delta}_\ell^{-2} \log T \right\rceil$ and $\tilde{\Delta}_\ell = 2^{-\ell}$. The current stage of batch b is ℓ_b.

Method `DisplayBatch` (Algorithm 2) explores batches as follows. In stage ℓ of batch b, we randomly choose $\mathrm{len}(b)$ least observed items in $\boldsymbol{B}_{b,\ell}$ and display them at randomly chosen positions in $\boldsymbol{I}_b$. If the number of these items is less than $\mathrm{len}(b)$, we mix them with randomly chosen more observed items, which are not explored. This exploration has two notable properties. First, it is uniform in the sense that no item in $\boldsymbol{B}_{b,\ell}$ is explored more than once than any other item in $\boldsymbol{B}_{b,\ell}$. Second, any item in $\boldsymbol{B}_{b,\ell}$ appears in any list over $\boldsymbol{B}_{b,\ell}$ with that item with the same probability. This is critical to avoid biases due to click models.

Method `CollectClicks` (Algorithm 3) collects feedback. We denote the *number of observations* of item d in stage ℓ of batch b by $\boldsymbol{n}_{b,\ell}(d)$ and the *number of clicks* on that item by $\boldsymbol{c}_{b,\ell}(d)$. At the end of the stage, all items $d \in \boldsymbol{B}_{b,\ell}$ are observed exactly n_ℓ times and we estimate the probability of clicking on item d as

$$\hat{\boldsymbol{c}}_{b,\ell}(d) = \boldsymbol{c}_{b,\ell}(d) / n_\ell \,. \tag{6}$$

Algorithm 2 `DisplayBatch`

Input: batch index b, time t

$\ell \leftarrow \ell_b,\ n_{\min} \leftarrow \min_{d \in \boldsymbol{B}_{b,\ell}} \boldsymbol{n}_{b,\ell}(d)$
Let $d_1, \ldots, d_{|\boldsymbol{B}_{b,\ell}|}$ be a random permutation of items $\boldsymbol{B}_{b,\ell}$ such that $\boldsymbol{n}_{b,\ell}(d_1) \leq \ldots \leq \boldsymbol{n}_{b,\ell}(d_{|\boldsymbol{B}_{b,\ell}|})$
Let $\pi \in \Pi_{\text{len}(b)}([\text{len}(b)])$ be a random permutation of position assignments
for $k = \boldsymbol{I}_b(1), \ldots, \boldsymbol{I}_b(2)$ **do**
  $\boldsymbol{d}_k^t \leftarrow d_{\pi(k - \boldsymbol{I}_b(1) + 1)}$

Algorithm 3 `CollectClicks`

Input: batch index b, time t

$\ell \leftarrow \ell_b,\ n_{\min} \leftarrow \min_{d \in \boldsymbol{B}_{b,\ell}} \boldsymbol{n}_{b,\ell}(d)$
for $k = \boldsymbol{I}_b(1), \ldots, \boldsymbol{I}_b(2)$ **do**
  if $\boldsymbol{n}_{b,\ell}(\boldsymbol{d}_k^t) = n_{\min}$ **then**
    $\boldsymbol{c}_{b,\ell}(\boldsymbol{d}_k^t) \leftarrow \boldsymbol{c}_{b,\ell}(\boldsymbol{d}_k^t) + \boldsymbol{c}_t(k)$
    $\boldsymbol{n}_{b,\ell}(\boldsymbol{d}_k^t) \leftarrow \boldsymbol{n}_{b,\ell}(\boldsymbol{d}_k^t) + 1$

Method `UpdateBatch` (Algorithm 4) updates batches and has three main parts. First, we compute `KL-UCB` upper and lower confidence bounds (Garivier & Cappe, 2011) for all items $d \in \boldsymbol{B}_{b,\ell}$ (lines 5–6),

$$\boldsymbol{U}_{b,\ell}(d) \leftarrow \operatorname*{arg\,max}_{q \in [\hat{\boldsymbol{c}}_{b,\ell}(d),\, 1]} \left\{ n_\ell \, D_{\text{KL}}(\hat{\boldsymbol{c}}_{b,\ell}(d) \,\|\, q) \leq \delta_T \right\},$$

$$\boldsymbol{L}_{b,\ell}(d) \leftarrow \operatorname*{arg\,min}_{q \in [0,\, \hat{\boldsymbol{c}}_{b,\ell}(d)]} \left\{ n_\ell \, D_{\text{KL}}(\hat{\boldsymbol{c}}_{b,\ell}(d) \,\|\, q) \leq \delta_T \right\},$$

where $D_{\text{KL}}(p \,\|\, q)$ denotes the *Kullback-Leibler divergence* between Bernoulli random variables with means p and q, and $\delta_T = \log T + 3 \log \log T$. Then we test whether batch b can be safely divided into s more attractive items and the rest (lines 7–15). If it can, we split the batch into two new batches (lines 21–27). The first batch contains s items and is over positions $\boldsymbol{I}_b(1), \ldots, \boldsymbol{I}_b(1) + s - 1$; and the second batch contains the remaining items and is over the remaining positions. The stage indices of new batches are initialized to 0. If the batch can be divided at multiple positions s, we choose the highest s. If the batch is not divided, we eliminate items that cannot be at position $\boldsymbol{I}_b(2)$ or higher with a high probability (lines 17–19).

The set of active batches is denoted by $\mathcal{A}$, and we explore and update these batches in parallel. The highest index of the latest added batch is $b_{\max}$. Note that $b_{\max} \leq 2K$, because any batch with at least two items is split at a unique position into two batches. `BatchRank` is initialized with a single batch over all positions and items (lines 6–7).

Note that by the design of `UpdateBatch`, the following invariants hold. First, the positions of active batches $\mathcal{A}$ are a partition of $[K]$ at any time t. Second, any batch contains at least as many items as is the number of the positions in that batch. Finally, when $\boldsymbol{I}_b(2) < K$, the number of items in batch b is equal to the number of its positions.

Algorithm 4 `UpdateBatch`

Input: batch index b, time t

// End-of-stage elimination
$\ell \leftarrow \ell_b$
if $\min_{d \in \boldsymbol{B}_{b,\ell}} \boldsymbol{n}_{b,\ell}(d) = n_\ell$ **then**
  for all $d \in \boldsymbol{B}_{b,\ell}$ **do**
    Compute $\boldsymbol{U}_{b,\ell}(d)$ and $\boldsymbol{L}_{b,\ell}(d)$
  Let $d_1, \ldots, d_{|\boldsymbol{B}_{b,\ell}|}$ be any permutation of items $\boldsymbol{B}_{b,\ell}$ such that $\boldsymbol{L}_{b,\ell}(d_1) \geq \ldots \geq \boldsymbol{L}_{b,\ell}(d_{|\boldsymbol{B}_{b,\ell}|})$
  for $k = 1, \ldots, \text{len}(b)$ **do**
    $B_k^+ \leftarrow \{d_1, \ldots, d_k\}$
    $B_k^- \leftarrow \boldsymbol{B}_{b,\ell} \setminus B_k^+$
  // Find a split at the position with the highest index
  $s \leftarrow 0$
  for $k = 1, \ldots, \text{len}(b) - 1$ **do**
    if $\boldsymbol{L}_{b,\ell}(d_k) > \max_{d \in B_k^-} \boldsymbol{U}_{b,\ell}(d)$ **then**
      $s \leftarrow k$
  if $(s = 0)$ and $(|\boldsymbol{B}_{b,\ell}| > \text{len}(b))$ **then**
    // Next elimination stage
    $\boldsymbol{B}_{b,\ell+1} \leftarrow \left\{ d \in \boldsymbol{B}_{b,\ell} : \boldsymbol{U}_{b,\ell}(d) \geq \boldsymbol{L}_{b,\ell}(d_{\text{len}(b)}) \right\}$
    $\ell_b \leftarrow \ell_b + 1$
  else if $s > 0$ **then**
    // Split
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{b_{\max} + 1, b_{\max} + 2\} \setminus \{b\}$
    $\boldsymbol{I}_{b_{\max}+1} \leftarrow (\boldsymbol{I}_b(1), \boldsymbol{I}_b(1) + s - 1)$
    $\boldsymbol{B}_{b_{\max}+1,0} \leftarrow B_s^+,\ \ell_{b_{\max}+1} \leftarrow 0$
    $\boldsymbol{I}_{b_{\max}+2} \leftarrow (\boldsymbol{I}_b(1) + s, \boldsymbol{I}_b(2))$
    $\boldsymbol{B}_{b_{\max}+2,0} \leftarrow B_s^-,\ \ell_{b_{\max}+2} \leftarrow 0$
    $b_{\max} \leftarrow b_{\max} + 2$

5. Analysis

In this section, we state our regret bound for `BatchRank`. Before we do so, we discuss our estimator of clicks in (6). In particular, we show that (6) is the attraction probability of item d scaled by the average examination probability in stage ℓ of batch b. The examination scaling preserves the order of attraction probabilities, and therefore `BatchRank` can operate on (6) in place of $\alpha(d)$.

5.1. Confidence Radii

Fix batch b, positions $\boldsymbol{I}_b$, stage ℓ, and items $\boldsymbol{B}_{b,\ell}$. Then for any item $d \in \boldsymbol{B}_{b,\ell}$, we can write the estimator in (6) as

$$\hat{\boldsymbol{c}}_{b,\ell}(d) = \frac{1}{n_\ell} \sum_{t \in \mathcal{T}} \sum_{k = \boldsymbol{I}_b(1)}^{\boldsymbol{I}_b(2)} \boldsymbol{c}_t(k) \mathbb{1}\{\boldsymbol{d}_k^t = d\} \tag{7}$$

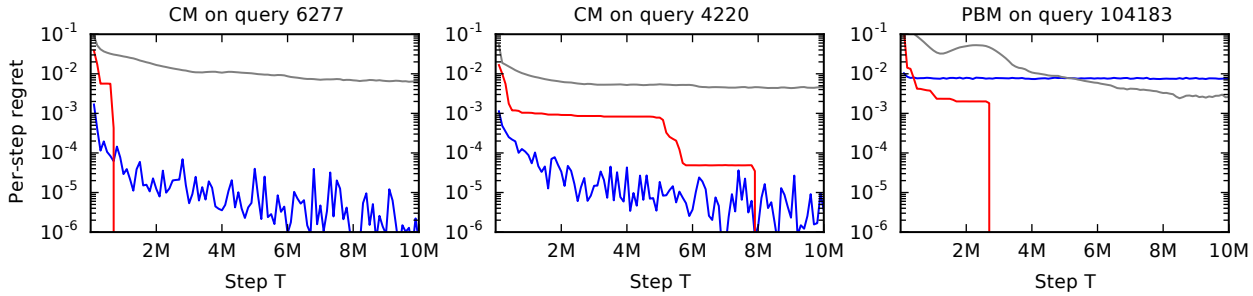

Figure 1. The expected per-step regret of `BatchRank` (red), `CascadeKL-UCB` (blue), and `RankedExp3` (gray) on three problems. The results are averaged over 10 runs.

for some set of n_ℓ time steps $\mathcal{T}$ and its expected value is

$$\bar{\boldsymbol{c}}_{b,\ell}(d) = \mathbb{E}\left[\hat{\boldsymbol{c}}_{b,\ell}(d)\right] . \tag{8}$$

The key step in the design of `BatchRank` is that we maintain confidence radii around (7). This is sound because the observations in (7),

$$\{\boldsymbol{X}_t(\boldsymbol{\mathcal{R}}_t, k)\boldsymbol{A}_t(d)\}_{t \in \{t \in \mathcal{T}: \boldsymbol{d}_k^t = d\}} \tag{9}$$

at any position k, are i.i.d. in time. More precisely, by the design of `DisplayBatch`, all displayed items from batch b are chosen randomly from $\boldsymbol{B}_{b,\ell}$; and independently of the realizations of $\boldsymbol{X}_t(\boldsymbol{\mathcal{R}}_t, k)$ and $\boldsymbol{A}_t(d)$, which are random as well. The last problem is that the policy for placing items at positions $1, \ldots, \boldsymbol{I}_b(1) - 1$ can change independently of batch b because `BatchRank` splits batches independently. But this has no effect on $\boldsymbol{X}_t(\boldsymbol{\mathcal{R}}_t, k)$ because the examination of position k does not depend on the order of higher ranked items (Assumption 3).

5.2. Correct Examination Scaling

Fix batch b, positions $\boldsymbol{I}_b$, stage ℓ, and items $\boldsymbol{B}_{b,\ell}$. Since the examination of a position does not depend on the order of higher ranked items, and does not depend on lower ranked items at all (Assumption 3), we can express the probability of clicking on any item $d \in \boldsymbol{B}_{b,\ell}$ in (7) as

$$\bar{\boldsymbol{c}}_{b,\ell}(d) = \frac{\alpha(d)}{|\mathcal{S}_d|} \sum_{\mathcal{R} \in \mathcal{S}_d} \sum_{k=\boldsymbol{I}_b(1)}^{\boldsymbol{I}_b(2)} \chi(\mathcal{R}, k) \mathbb{1}\{d_k = k\} , \tag{10}$$

where

$$\mathcal{S}_d = \big\{(e_1, \ldots, e_{\boldsymbol{I}_b(2)}) : d \in \{e_{\boldsymbol{I}_b(1)}, \ldots, e_{\boldsymbol{I}_b(2)}\} , \\ (e_{\boldsymbol{I}_b(1)}, \ldots, e_{\boldsymbol{I}_b(2)}) \in \Pi_{\text{len}(b)}(\boldsymbol{B}_{b,\ell})\big\}$$

is the set of all lists with permutations of $\boldsymbol{B}_{b,\ell}$ on positions $\boldsymbol{I}_b$ that contain item d, for some fixed higher ranked items $e_1, \ldots, e_{\boldsymbol{I}_b(1)-1} \notin \boldsymbol{B}_{b,\ell}$. Let $d^* \in \boldsymbol{B}_{b,\ell}$ be any item such that $\alpha(d^*) \geq \alpha(d)$, and $\bar{\boldsymbol{c}}_{b,\ell}(d^*)$ and $\mathcal{S}_{d^*}$ be defined analogously to $\bar{\boldsymbol{c}}_{b,\ell}(d)$ and $\mathcal{S}_d$ above. Then we argue that

$$\bar{\boldsymbol{c}}_{b,\ell}(d^*)/\alpha(d^*) \geq \bar{\boldsymbol{c}}_{b,\ell}(d)/\alpha(d) , \tag{11}$$

the examination scaling of a less attractive item d is never higher than that of a more attractive item d^*.

Before we prove (11), note that for any list $\mathcal{R} \in \mathcal{S}_d$, there exists one and only one list in $\mathcal{S}_{d^*}$ that differs from $\mathcal{R}$ only in that items d and d^* are exchanged. Let this list be $\mathcal{R}^*$. We analyze three cases. First, suppose that list $\mathcal{R}$ does not contain item d^*. Then by Assumption 3, the examination probabilities of d in $\mathcal{R}$ and d^* in $\mathcal{R}^*$ are the same. Second, let item d^* be ranked higher than item d in $\mathcal{R}$. Then by Assumption 5, the examination probability of d in $\mathcal{R}$ is not higher than that of d^* in $\mathcal{R}^*$. Third, let item d^* be ranked lower than item d in $\mathcal{R}$. Then by Assumption 3, the examination probabilities of d in $\mathcal{R}$ and d^* in $\mathcal{R}^*$ are the same, since they do not depend on lower ranked items. Finally, from the definition in (10) and that $|\mathcal{S}_d| = |\mathcal{S}_{d^*}|$, we have that (11) holds.

5.3. Regret Bound

For simplicity of exposition, let $\alpha(1) > \ldots > \alpha(L) > 0$. Let $\alpha_{\max} = \alpha(1)$, and $\chi^*(k) = \chi(\mathcal{R}^*, k)$ for all $k \in [K]$. The regret of `BatchRank` is bounded below.

Theorem 1. *For any stochastic click bandit in Section 3.1 that satisfies Assumptions 1 to 6 and $T \geq 5$, the expected T-step regret of* `BatchRank` *is bounded as*

$$R(T) \leq \frac{192K^3L}{(1 - \alpha_{\max})\Delta_{\min}} \log T + 4KL(3e + K) ,$$

where $\Delta_{\min} = \min_{k \in [K]} \{\alpha(k) - \alpha(k+1)\}$.

Proof. The key idea is to bound the expected T-step regret in any batch (Lemma 7 in Appendix). Since the number of batches is at most $2K$, the regret of `BatchRank` is at most $2K$ times larger than that of in any batch.

The regret in a batch is bounded as follows. Let all confidence intervals hold, $\boldsymbol{I}_b$ be the positions of batch b, and the maximum gap in batch b be

$$\Delta_{\max} = \max_{d \in \{\boldsymbol{I}_b(1), \ldots, \boldsymbol{I}_b(2)-1\}} [\alpha(d) - \alpha(d+1)] .$$

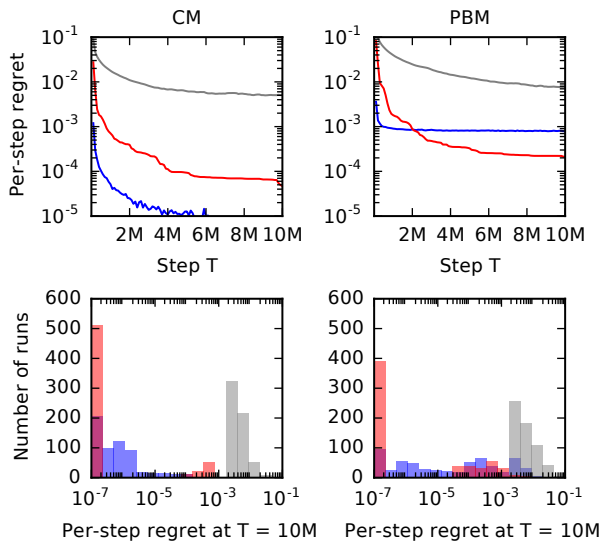

Figure 2. The comparison of `BatchRank` (red), `CascadeKL-UCB` (blue), and `RankedExp3` (gray) in the CM and PBM. In the top plots, we report the per-step regret as a function of time T, averaged over 60 queries and 10 runs per query. In the bottom plots, we show the distribution of the regret at $T = 10$M.

If the gap of item d in batch b is $O(K\Delta_{\max})$, its regret is dominated by the time that the batch splits, and we bound this time in Lemma 6 in Appendix. Otherwise, the item is likely to be eliminated before the split, and we bound this time in Lemma 5 in Appendix. Now take the maximum of these upper bounds. ■

5.4. Discussion

Our upper bound in Theorem 1 is logarithmic in the number of steps T, linear in the number of items L, and polynomial in the number of positions K. To the best of our knowledge, this is the first gap-dependent upper bound on the regret of a learning algorithm that has sublinear regret in both the CM and PBM. The gap $\Delta_{\min}$ characterizes the hardness of sorting $K + 1$ most attractive items, which is sufficient for solving our problem. In practice, the maximum attraction probability $\alpha_{\max}$ is bounded away from 1. Therefore, the dependence on $(1 - \alpha_{\max})^{-1}$ is not critical. For instance, in most queries in Section 6, $\alpha_{\max} \leq 0.9$.

We believe that the cubic dependence on K is not far from being optimal. In particular, consider the problem of learning the most clicked item-position pair in the PBM (Section 2.1), which is easier than our problem. This problem can be solved as a stochastic rank-1 bandit (Katariya et al., 2017b) by `Rank1Elim`. Now consider the following PBM. The examination probability of the first position is close to one and the examination probabilities of all other positions are close to zero. Then the T-step regret of `Rank1Elim` is $O([K^3 + K^2 L \Delta_{\min}^{-1}] \log T)$ because $\mu = O(1/K)$, where μ is defined in Katariya et al. (2017b). Note that the gap-dependent term nearly matches our upper bound.

The KL confidence intervals in `BatchRank` are necessary to achieve sample efficiency. The reason is, as we prove in Lemma 9 in Appendix, that

$$\chi(1 - m) D_{\mathrm{KL}}(\alpha \,\|\, \alpha^*) \leq D_{\mathrm{KL}}(\chi\alpha \,\|\, \chi\alpha^*)$$

for any $\chi, \alpha, \alpha^* \in [0, 1]$ and $m = \max\{\alpha, \alpha^*\}$. This implies that any two items with the expected rewards of $\chi\alpha$ and $\chi\alpha^*$ can be distinguished in $O(\chi^{-1}(\alpha^* - \alpha)^{-2})$ observations for any scaling factor χ, when m is bounded away from 1. Suppose that $\alpha < \alpha^*$. Then the expected per-step regret for choosing the suboptimal item is $\chi(\alpha^* - \alpha)$, and the expected cumulative regret is $O((\alpha^* - \alpha)^{-1})$. The key observation is that the regret is independent of χ. This is a major improvement over `UCB1` confidence intervals, which only lead to $O(\chi^{-1}(\alpha^* - \alpha)^{-1})$ regret. Because χ can be exponentially small, such a dependence is undesirable.

The elimination of items in `UpdateBatch` (lines 17–19) is necessary. The regret of `BatchRank` would be quadratic in L otherwise.

6. Experiments

We experiment with the *Yandex* dataset (Yandex), a dataset of 35 million (M) search sessions, each of which may contain multiple search queries. Each query is associated with displayed documents at positions 1 to 10 and their clicks. We select 60 frequent search queries, and learn their CMs and PBMs using PyClick (Chuklin et al., 2015), which is an open-source library of click models for web search. In each query, our goal it to rerank $L = 10$ most attractive items with the objective of maximizing the expected number of clicks at the first $K = 5$ positions. This resembles a real-world setting, where the learning agent would only be allowed to rerank highly attractive items, and not allowed to explore unattractive items (Zoghi et al., 2016).

`BatchRank` is compared to two methods, `CascadeKL-UCB` (Kveton et al., 2015a) and `RankedExp3` (Radlinski et al., 2008a). `CascadeKL-UCB` is an optimal algorithm for learning to rank in the cascade model. `RankedExp3` is a variant of ranked bandits (Section 7) where the base bandit algorithm is `Exp3` (Auer et al., 1995). This approach is popular in practice and does not make any independence assumptions on the attractions of items.

Many solutions in our queries are near optimal, and therefore the optimal solutions are hard to learn. Therefore, we decided to evaluate the performance of algorithms by their *expected per-step regret*, in up to 10M steps. If a solution is suboptimal and does not improve over time, its expected per-step regret remains constant and is bounded away from

zero, and this can be easily observed even if the gap of the solution is small. We expect this when CascadeKL-UCB is applied to the PBM because CascadeKL-UCB has no guarantees in this model. The reported regret is averaged over periods of 100k steps to reduce randomness.

We report the performance of all compared algorithms on two CMs and one PBM in Figure 1. The plots are chosen to represent general trends in this experiment. In the CM, CascadeKL-UCB performs very well on most queries. This is not surprising since CascadeKL-UCB is designed for the CM. BatchRank often learns the optimal solution quickly (Figure 1a), but sometimes this requires close to $T = 10$M steps (Figure 1b). In the PBM, CascadeKL-UCB may converge to a suboptimal solution. Then its expected per-step regret remains constant and even RankedExp3 can learn a better solution over time (Figure 1c). We also observe that BatchRank outperforms RankedExp3 in all experiments.

We report the average performance of all compared algorithms in both click models in Figure 2. These trends confirm our earlier findings. In the CM, CascadeKL-UCB outperforms BatchRank; while in the PBM, BatchRank outperforms CascadeKL-UCB at $T = 2$M steps. The regret of CascadeKL-UCB in the PBM flattens and is bounded away from zero. This trend can be explained by the histograms in Figure 2. They show that CascadeKL-UCB converges to suboptimal solutions, whose regret is at least 10^{-3}, in one sixth of its runs. The performance of BatchRank is more robust and we do not observe many runs whose regret is of that magnitude.

We are delighted with the performance of BatchRank. Although it is not designed to be optimal (Section 5.4), it is more robust than CascadeKL-UCB and clearly outperforms RankedExp3. The performance of CascadeKL-UCB is unexpectedly good. Although it does not have guarantees in the PBM, it performs very well on many queries. We plan to investigate this in our future work.

7. Related Work

A popular approach to online learning to rank are *ranked bandits* (Radlinski et al., 2008a; Slivkins et al., 2013). The key idea in ranked bandits is to model each position in the recommended list as an individual bandit problem, which is then solved by a *base bandit algorithm*. This algorithm is typically adversarial (Auer et al., 1995) because the distribution of clicks on lower positions is affected by higher positions. We compare to ranked bandits in Section 6.

Online learning to rank in click models (Craswell et al., 2008; Chuklin et al., 2015) was recently studied in several papers (Kveton et al., 2015a; Combes et al., 2015; Kveton et al., 2015b; Katariya et al., 2016; Zong et al., 2016; Li et al., 2016; Lagree et al., 2016). In all of these papers, the attraction probabilities of items are estimated from clicks and the click model. The learning agent has no guarantees beyond this model.

The problem of finding the most clicked item-position pair in the PBM, which is arguably easier than our problem of finding K most clicked item-position pairs, can be solved as a stochastic rank-1 bandit (Katariya et al., 2017b;a). We discuss our relation to these works in Section 5.4.

Our problem can be also viewed as an instance of partial monitoring, where the attraction indicators of items are unobserved. General partial-monitoring algorithms (Agrawal et al., 1989; Bartok et al., 2012; Bartok & Szepesvari, 2012; Bartok et al., 2014) are unsuitable for our setting because their computational complexity is polynomial in the number of actions, which is exponential in K.

The click model is a model of how the user interacts with a list of documents (Chuklin et al., 2015), and many such models have been proposed (Becker et al., 2007; Richardson et al., 2007; Craswell et al., 2008; Chapelle & Zhang, 2009; Guo et al., 2009a;b). Two fundamental click models are the CM (Craswell et al., 2008) and PBM (Richardson et al., 2007). These models have been traditionally studied separately. In this work, we show that learning to rank problems in these models can be solved by the same algorithm, under reasonable assumptions.

8. Conclusions

We propose stochastic click bandits, a framework for online learning to rank in a broad class of click models that encompasses two most fundamental click models, the cascade and position-based models. In addition, we propose a computationally and sample efficient algorithm for solving our problems, BatchRank, and derive an upper bound on its T-step regret. Finally, we evaluate BatchRank on web search queries. Our algorithm outperforms ranked bandits (Radlinski et al., 2008a), a popular online learning to rank approach; and is more robust than CascadeKL-UCB (Kveton et al., 2015a), an existing algorithm for online learning to rank in the cascade model.

The goal of this work is not to propose the optimal algorithm for our setting, but to demonstrate that online learning to rank in multiple click models is possible with theoretical guarantees. We strongly believe that the design of BatchRank, as well as its analysis, can be improved. For instance, BatchRank resets its estimators of clicks in each batch, which is wasteful. In addition, based on the discussion in Section 5.4, our analysis may be loose by a factor of K. We hope that the practically relevant setting, which is introduced in this paper, will spawn new enthusiasm in the community and lead to more work in this area.

References

Agichtein, Eugene, Brill, Eric, and Dumais, Susan. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th Annual International ACM SIGIR Conference*, pp. 19–26, 2006.

Agrawal, Rajeev, Teneketzis, Demosthenis, and Anantharam, Venkatachalam. Asymptotically efficient adaptive allocation schemes for controlled i.i.d. processes: Finite parameter space. *IEEE Transactions on Automatic Control*, 34(3):258–267, 1989.

Auer, Peter, Cesa-Bianchi, Nicolo, Freund, Yoav, and Schapire, Robert. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pp. 322–331, 1995.

Auer, Peter, Cesa-Bianchi, Nicolo, and Fischer, Paul. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.

Bartok, Gabor and Szepesvari, Csaba. Partial monitoring with side information. In *Proceedings of the 23rd International Conference on Algorithmic Learning Theory*, pp. 305–319, 2012.

Bartok, Gabor, Zolghadr, Navid, and Szepesvari, Csaba. An adaptive algorithm for finite stochastic partial monitoring. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Bartok, Gabor, Foster, Dean, Pal, David, Rakhlin, Alexander, and Szepesvari, Csaba. Partial monitoring - classification, regret bounds, and algorithms. *Mathematics of Operations Research*, 39(4):967–997, 2014.

Becker, Hila, Meek, Christopher, and Chickering, David Maxwell. Modeling contextual factors of click rates. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pp. 1310–1315, 2007.

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

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

Combes, Richard, Magureanu, Stefan, Proutiere, Alexandre, and Laroche, Cyrille. Learning to rank: Regret lower bounds and efficient algorithms. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 2015.

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

Garivier, Aurelien and Cappe, Olivier. The KL-UCB algorithm for bounded stochastic bandits and beyond. In *Proceeding of the 24th Annual Conference on Learning Theory*, pp. 359–376, 2011.

Grotov, Artem, Chuklin, Aleksandr, Markov, Ilya, Stout, Luka, Xumara, Finde, and de Rijke, Maarten. A comparative study of click models for web search. In *Proceedings of the 6th International Conference of the CLEF Association*, 2015.

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

Guo, Fan, Liu, Chao, and Wang, Yi Min. Efficient multiple-click models in web search. In *Proceedings of the 2nd ACM International Conference on Web Search and Data Mining*, pp. 124–131, 2009b.

Hoeffding, Wassily. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Hofmann, Katja, Schuth, Anne, Whiteson, Shimon, and de Rijke, Maarten. Reusing historical interaction data for faster online learning to rank for IR. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining*, pp. 183–192, 2013.

Katariya, Sumeet, Kveton, Branislav, Szepesvari, Csaba, and Wen, Zheng. DCM bandits: Learning to rank with multiple clicks. In *Proceedings of the 33rd International Conference on Machine Learning*, pp. 1215–1224, 2016.

Katariya, Sumeet, Kveton, Branislav, Szepesvari, Csaba, Vernade, Claire, and Wen, Zheng. Bernoulli rank-1 bandits for click feedback. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017a.

Katariya, Sumeet, Kveton, Branislav, Szepesvari, Csaba, Vernade, Claire, and Wen, Zheng. Stochastic rank-1 bandits. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017b.

Kveton, Branislav, Szepesvari, Csaba, Wen, Zheng, and Ashkan, Azin. Cascading bandits: Learning to rank in the cascade model. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015a.

Kveton, Branislav, Wen, Zheng, Ashkan, Azin, and Szepesvari, Csaba. Combinatorial cascading bandits. In *Advances in Neural Information Processing Systems 28*, pp. 1450–1458, 2015b.

Lagree, Paul, Vernade, Claire, and Cappe, Olivier. Multiple-play bandits in the position-based model. In *Advances in Neural Information Processing Systems 29*, pp. 1597–1605, 2016.

Lai, T. L. and Robbins, Herbert. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

Li, Shuai, Wang, Baoxiang, Zhang, Shengyu, and Chen, Wei. Contextual combinatorial cascading bandits. In *Proceedings of the 33rd International Conference on Machine Learning*, pp. 1245–1253, 2016.

Liu, Tie-Yan. *Learning to Rank for Information Retrieval*. Springer, 2011.

Radlinski, Filip, Kleinberg, Robert, and Joachims, Thorsten. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 784–791, 2008a.

Radlinski, Filip, Kurup, Madhu, and Joachims, Thorsten. How does clickthrough data reflect retrieval quality? In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, pp. 43–52, 2008b.

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

Slivkins, Aleksandrs, Radlinski, Filip, and Gollapudi, Sreenivas. Ranked bandits in metric spaces: Learning diverse rankings over large document collections. *Journal of Machine Learning Research*, 14(1):399–436, 2013.

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

Yue, Yisong and Joachims, Thorsten. Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 1201–1208, 2009.

Zoghi, Masrour, Tunys, Tomas, Li, Lihong, Jose, Damien, Chen, Junyan, Chin, Chun Ming, and de Rijke, Maarten. Click-based hot fixes for underperforming torso queries. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 195–204, 2016.

Zong, Shi, Ni, Hao, Sung, Kenny, Ke, Nan Rosemary, Wen, Zheng, and Kveton, Branislav. Cascading bandits for large-scale recommendation problems. In *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence*, 2016.