Learning Diverse Rankings with Multi-Armed Bandits

Filip Radlinski FILIP@CS.CORNELL.EDU
Robert Kleinberg RDK@CS.CORNELL.EDU
Thorsten Joachims TJ@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14853 USA

Abstract

Algorithms for learning to rank Web documents usually assume a document's relevance is independent of other documents. This leads to learned ranking functions that produce rankings with redundant results. In contrast, user studies have shown that diversity at high ranks is often preferred. We present two online learning algorithms that directly learn a diverse ranking of documents based on users' clicking behavior. We show that these algorithms minimize abandonment, or alternatively, maximize the probability that a relevant document is found in the top k positions of a ranking. Moreover, one of our algorithms asymptotically achieves optimal worst-case performance even if users' interests change.

1. Introduction

Web search has become an essential component of the Internet infrastructure, and has hence attracted significant interest from the machine learning community (e.g. Herbrich et al., 2000; Burges et al., 2005; Radlinski & Joachims, 2005; Chu & Ghahramani, 2005; Metzler & Croft, 2005; Yue et al., 2007; Taylor et al., 2008). The conventional approach to this learning-to-rank problem has been to assume the availability of manually labeled training data. Usually, this data consists of a set of documents judged as relevant or not to specific queries, or of pairwise judgments comparing the relative relevance of pairs of documents. These judgments are used to optimize a ranking function offline, to a standard information retrieval metric, then deploying the learned function in a live search engine.

We propose a new learning to rank problem formulation that differs in three fundamental ways. First, unlike most previous methods, we learn from usage data rather than manually labeled relevance judgments. Usage data is available in much larger quantities and at much lower cost. Moreover, unlike manual judgments, which need to be constantly updated to stay relevant, usage data naturally reflects current users' needs and the documents currently available. Although some researchers have transformed usage data into relevance judgments, or used it to generate features (e.g. Joachims, 2002; Radlinski & Joachims, 2005; Agichtein et al., 2006), we go one step further by directly optimizing a usage-based metric.

Second, we propose an online learning approach for learning from usage data. As training data is being collected, it immediately impacts the rankings shown. This means the learning problem we address is regret minimization, where the goal is to minimize the total number of poor rankings displayed over *all* time. In particular, in this setting there is a natural tradeoff between exploration and exploitation: It may be valuable in the long run to present some rankings with unknown documents, to allow training data about these documents to be collected. In contrast, in the short run exploitation is typically optimal. With only few exceptions (e.g. Radlinski & Joachims, 2007), previous work does not consider such an online approach.

Third and most importantly, except for (Chen & Karger, 2006), previous algorithms for learning to rank have considered the relevance of each document independently of other documents. This is reflected in the performance measures typically optimized, such as Precision, Recall, Mean Average Precision (MAP) (Baeza-Yates & Ribeiro-Neto, 1999) and Normalized Discounted Cumulative Gain (NDCG) (Burges et al., 2006). In fact, recent work has shown that these measures do not necessarily correlate with user satisfaction (Turpin & Scholer, 2006). Additionally, it intuitively stands to reason that presenting many slight variations of the same relevant document in web search results may increase the MAP or NDCG score, yet would be suboptimal for users. Moreover, web queries often have different meanings for different users (a canonical example is the query *jaguar*) suggesting that a ranking with diverse documents may be preferable.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

We will show how clickthrough data can be used to learn rankings maximizing the probability that any new user will find at least one relevant document high in the ranking.

2. Related Work

The standard approach for learning to rank uses training data, in the form of judgments assessing the relevance of individual documents to a query, to learn parameters θ for a scoring function $f(q, d_i, \theta)$. Given a new query q, this function computes $f(q, d_i, \theta)$ for each document d_i *independently* and ranks documents by decreasing score (e.g. Herbrich et al., 2000; Joachims, 2002; Burges et al., 2005; Chu & Ghahramani, 2005). This also applies to recent algorithms that learn θ to maximize nonlinear performance measures such as MAP (Metzler & Croft, 2005; Yue et al., 2007) and NDCG (Burges et al., 2006; Taylor et al., 2008).

The theoretical model that justifies ranking documents in this way is the probabilistic ranking principle (Robertson, 1977). It suggests that documents should be ranked by their probablility of relevance to the query. However, the optimality of such a ranking relies on the assumption that there are no statistical dependencies between the probabilities of relevance among documents – an assumption that is clearly violated in practice. For example, if one document about jaguar cars is not relevant to a user who issues the query *jaguar*, other car pages become less likely to be relevant. Furthermore, empirical studies have shown that given a fixed query, the same document can have different relevance to different users (Teevan et al., 2007). This undermines the assumption that each document has a single relevance score that can be provided as training data to the learning algorithm. Finally, as users are usually satisfied with finding a small number of, or even just one, relevant document, the usefulness and relevance of a document does depend on other documents ranked higher.

As a result, most search engines today attempt to eliminate redundant results and produce *diverse* rankings that include documents that are potentially relevant to the query for different reasons. However, learning optimally diverse rankings using expert judgments would require document relevance to be measured for different possible meanings of a query. While the TREC interactive track[1] provides some documents labeled in this way for a small number of queries, such document collections are even more difficult to create than standard expert labeled collections.

[1] http://trec.nist.gov/data/t11_interactive/t11i.html

Several non-learning algorithms for obtaining a diverse ranking of documents from a non-diverse ranking have been proposed. One common one is Maximal Marginal Relevance (MMR) (Carbonell & Goldstein, 1998). Given a similarity (relevance) measure between documents and queries $sim_1(d, q)$ and a similarity measure between pairs of documents $sim_2(d_i, d_j)$, MMR iteratively selects documents by repeatedly finding $d_i = \text{argmax}_{d \in \mathcal{D}} \lambda sim_1(d, q) - (1 - \lambda) \max_{d_j \in S} sim_2(d, d_j)$ where S is the set of documents already selected and λ is a tuning parameter. In this way MMR selects the most relevant documents that are also different from any documents already selected.

Critically, MMR requires that the relevance function $sim_1(d, q)$, and the similarity function $sim_2(d_i, d_j)$ is known. It is usual to obtain sim_1 and sim_2 using algorithms such as those discussed above. The goal of MMR is to rerank an already learned ranking (that of ranking documents by decreasing sim_1 score) to improve diversity. All previous approaches of which we are aware that optimize diversity similarly require a relevance function to be learned prior to performing a diversification step (Zhu et al., 2007; Zhang et al., 2005; Zhai et al., 2003), with the exception of Chen and Karger (2006). Rather, they require that a model for estimating the probability a document is relevant, given a query and other non-relevant documents, is available. In contrast, we directly learn a diverse ranking of documents using users' clicking behavior.

3. Problem Formalization

We address the problem of learning an optimally diversified ranking of documents $\mathcal{D} = \{d_1, \ldots, d_n\}$ for one fixed query. Suppose we have a population of users, where each user u_i considers some subset of documents $A_i \subset \mathcal{D}$ as relevant to the query, and the remainder of the documents as non-relevant. Intuitively, users with different interpretations for the query would have different relevant sets, while users with similar interpretations would have similar relevant sets.

At time t, we interact with user u_t with relevant set A_t. We present an ordered set of k documents, $B_t = (b_1(t), \ldots, b_k(t))$. The user considers the results in order, and clicks on up to one document. The probability of user u_t clicking on document d_i (conditional on the user not clicking on a document presented earlier in the ranking) is assumed to be $p_{ti} \in [0, 1]$. We refer to the vector of probabilities $(p_{ti})_{i \in \mathcal{D}}$ as the *type* of user u_t. In the simplest case, we could take $p_{ti} = 1$ if $d_i \in A_t$ and 0 otherwise, in which case the user clicks on the first relevant document or does not click if no documents in B_t are relevant. However, in reality clicks tend to be noisy although more relevant docu-

Algorithm 1 Ranked Explore and Commit

input: Documents $(d_1, .., d_n)$, parameters ϵ, δ, k.
$x \leftarrow \lceil 2k^2/\epsilon^2 \log(2k/\delta) \rceil$
$(b_1, \ldots, b_k) \leftarrow k$ arbitrary documents.
for i=1 ... k **do** *At every rank*
  $\forall j.\ p_j \leftarrow 0$
  for counter=1 ... x **do** *Loop x times*
    for j=1 ... n **do** *over every document d_j*
      $b_i \leftarrow d_j$
      display $\{b_1, \ldots, b_k\}$ to user; record clicks
      if user clicked on b_i **then** $p_j \leftarrow p_j + 1$
    end for
  end for
  $j^* \leftarrow \text{argmax}_j\, p_j$ *Commit to best document at this rank*
  $b_i \leftarrow d_{j^*}$
end for

ments are more likely to be clicked on. In our analysis, we will take $p_{ti} \in [0, 1]$.

We get payoff 1 if the user clicks, 0 if not. The goal is to maximize the total payoff, summing over all time. This payoff represents the number of users who clicked on any result, which can be interpreted as the user finding at least one potentially relevant document (so long as p_{ti} is higher when $d_i \in A_t$ than when $d_i \notin A_t$).

The event that a user does not click is called *abandonment* since the user abandoned the search results. Abandonment is an important measure of user satisfaction because it indicates that users were presented with search results of no potential interest.

4. Learning Algorithms

We now present two algorithms that directly minimize the abandonment rate. At a high level, both algorithms learn a marginal utility for each document at each rank, displaying documents to maximize the probability that a new user of the search system would find at least one relevant document within the top k positions. The algorithms differ in their assumptions.

4.1. Ranked Explore and Commit

The first algorithm we present is a simple greedy strategy that assumes that user interests and documents do not change over time. As we will see, after T time steps this algorithm achieves a payoff of at least $(1-1/e-\epsilon)OPT - O(k^3 n/\epsilon^2 \ln(k/\delta))$ with probability at least $1-\delta$. OPT denotes the maximal payoff that could be obtained if the click probabilities p_{ti} were known ahead of time for all users and documents, and $(1-1/e)OPT$ is the best obtainable polynomial time approximation, as will be explained in Section 5.1.

As described in Algorithm 1, Ranked Explore and Commit (REC) iteratively selects documents for each rank. At each rank position i, every document d_j is presented a fixed number x times, and the number of clicks it receives during these presentations is recorded. After nx presentations, the algorithm permanently assigns the document that received the most clicks to the current rank, and moves on to the next rank.

Algorithm 2 Ranked Bandits Algorithm

initialize $\mathsf{MAB}_1(n), \ldots, \mathsf{MAB}_k(n)$ *Initialize MABs*
for t = 1 ... T **do**
  for i = 1 ... k **do** *Sequentially select documents*
    $\hat{b}_i(t) \leftarrow$ select-arm (MAB_i)
    if $\hat{b}_i(t) \in \{b_1(t), .., b_{i-1}(t)\}$ **then** *Replace repeats*
      $b_i(t) \leftarrow$ arbitrary unselected document
    else
      $b_i(t) \leftarrow \hat{b}_i(t)$
    end if
  end for
  display $\{b_1(t), \ldots, b_k(t)\}$ to user; record clicks
  for i = 1 ... k **do** *Determine feedback for MAB_i*
    if user clicked $b_i(t)$ **and** $\hat{b}_i(t) = b_i(t)$ **then**
      $f_{it} = 1$
    else
      $f_{it} = 0$
    end if
    update $(\mathsf{MAB}_i, arm = \hat{b}_i(t), reward = f_{it})$
  end for
end for

4.2. Ranked Bandits Algorithm

Ranked Explore and Commit is purely greedy, meaning that after each document is selected, this decision is never revisited. In particular, this means that if user interests or documents change, REC can perform arbitrarily poorly. In contrast, the Ranked Bandits Algorithm (RBA) achieves a combined payoff of $(1-1/e)OPT - O(k\sqrt{Tn \log n})$ after T time steps even if documents and user interests change over time.

This algorithm leverages standard theoretical results for multi-armed bandits. Multi-armed bandits (MAB) are modeled on casino slot machines (sometimes called one-armed bandits). The goal of standard MAB algorithms is to select the optimal sequence of slot machines to play to maximize the expected total reward collected. For further details, refer to (Auer et al., 2002a). The ranked bandits algorithm runs an MAB instance MAB_i for *each rank* i. Each of the k copies of the multi-armed bandit algorithm maintains a value (or index) for every document. When selecting the ranking to display to users, the algorithm MAB_1 is responsible for choosing which document is shown at rank 1. Next, the algorithm MAB_2 determines which

document is shown at rank 2, unless the same document was selected at the highest rank. In that case, the second document is picked arbitrarily. This process is repeated to select all top k documents.

Next, after a user considers up to the top k documents in order and clicks on one or none, we need to update the indices. If the user clicks on a document actually selected by an MAB instance, the reward for the arm corresponding to that document for the multi-armed bandit at that rank is 1. The reward for the arms corresponding to all other selected documents is 0. In particular, note that the RBA treats the bandits corresponding to each rank independently. Precise pseudocode for the algorithm is presented in Algorithm 2. A generalization of this algorithm, in an abstract setting without the application to Information Retrieval, was discovered independently by Streeter and Golovin (2007).

The actual MAB algorithm used for each MAB_i instance is not critical, and in fact any algorithm for the non-stochastic multi-armed bandit problem will suffice. Our theoretical analysis only requires that:

- The algorithm has a set S of n strategies.
- In each period t a payoff function $f_t : S \to [0, 1]$ is defined. This function is not revealed to the algorithm, and may depend on the algorithm's choices before time t.
- In each period the algorithm chooses a (random) element $y_t \in S$ based on the feedback revealed in prior periods.
- The feedback revealed in period t is $f_t(y_t)$.
- The expected payoffs of the chosen strategies satisfy:
$$\sum_{t=1}^{T} \mathbf{E}[f_t(y_t)] \;\geq\; \max_{y \in S} \sum_{t=1}^{T} \mathbf{E}[f_t(y)] \;-\; R(T)$$
where $R(T)$ is an explicit function in $o(T)$ which depends on the particular multi-armed bandit algorithm chosen, and the expectation is over any randomness in the algorithm. We will use the **Exp3** algorithm in our analysis, where $R(T) = O\left(\sqrt{Tn \log n}\right)$ (Auer et al., 2002b).

We will also later see that although these conditions are needed to bound worst-case performance, better practical performance may be obtained at the expense of worst-case performance if they are relaxed.

5. Theoretical Analysis

We now present a theoretical analysis of the algorithms presented in Section 4. First however, we discuss the offline version of this optimization problem.

5.1. The Offline Optimization Problem

The problem of choosing the optimum set of k documents for a given user population is NP-hard, even if all the information about the user population (i.e. the set of relevant documents for each user) is given offline and we restrict ourselves to $p_{ij} \in \{0, 1\}$. This is because selecting the optimal set of documents is equivalent to the maximum coverage problem: Given a positive integer k and a collection of subsets $S_1, S_2, \ldots, S_n$ of an m-element set, find k of the subsets whose union has the largest possible cardinality.

The standard greedy algorithm for the maximum coverage problem, translated to our setting, iteratively chooses the document that is relevant to the most users for whom a relevant document has not yet been selected. This algorithm is a $(1 - 1/e)$-approximation algorithm for this maximization problem (Nemhauser et al., 1978). The $(1 - 1/e)$ factor is optimal and no better worst-case approximation ratio is achievable in polynomial time unless $NP \subseteq DTIME\left(n^{\log \log n}\right)$ (Khuller et al., 1997).

5.2. Analysis of Ranked Bandits Algorithm

We start by analyzing the Ranked Bandits Algorithm. This algorithm works by simulating the offline greedy algorithm, using a separate instance of the multi-armed bandit algorithm for each step of the greedy algorithm. Except for the sublinear regret term, the combined payoff is as high as possible without violating the hardness-of-approximation result stated in the preceding paragraph.

To analyze the RBA, we first restrict ourselves to users who click on any given document with probability either 0 or 1. We refer to this restricted type of user as a *deterministic user*; we will relax the requirement later. Additionally, this analysis applies to a worst case (and hence fixed) sequence of users.

Further, it is useful to introduce some notation. For a set A and a sequence $B = (b_1, b_2, \ldots, b_k)$, let

$$G_i(A, B) = \begin{cases} 1 & \text{if } A \text{ intersects } \{b_1, \ldots, b_i\} \\ 0 & \text{otherwise} \end{cases}$$

$$g_i(A, B) = G_i(A, B) - G_{i-1}(A, B)$$

Recalling that A_t is the set of documents relevant to user u_t, we see that $G_k(A_t, B)$ is the payoff of presenting B to the user u_t. Let

$$B^* = \operatorname*{argmax}_B \sum_{t=1}^{T} G_k(A_t, B),$$

$$OPT = \sum_{t=1}^{T} G_k(A_t, B^*).$$

Recall that $(\hat{b}_1(t), \ldots, \hat{b}_k(t))$ is the sequence of documents chosen by the algorithms $\mathsf{MAB}_1, \ldots, \mathsf{MAB}_k$ at time t, and that $(b_1(t), \ldots, b_k(t))$ is the sequence of documents presented to the user. Define the feedback function f_{it} for algorithm MAB_i at time t, as follows:

$$f_{it}(b) = \begin{cases} 1 & \text{if } G_{i-1}(A_t, B_t) = 0 \text{ and } b \in A_t \\ 0 & \text{otherwise} \end{cases}.$$

Note that the value of f_{it} defined in the pseudocode for the Ranked Bandits Algorithms is equal to $f_{it}(\hat{b}_i(t))$.

Lemma 1. *For all i,*

$$\begin{aligned} &\mathbf{E}\left[\sum_{t=1}^{T} g_i(A_t, B_t)\right] \\ &\geq \frac{1}{k}\mathbf{E}\left[\sum_{t=1}^{T} (G_k(A_t, B^*) - G_{i-1}(A_t, B_t))\right] - R(T) \\ &= \frac{1}{k}OPT - \frac{1}{k}\mathbf{E}\left[\sum_{t=1}^{T} G_{i-1}(A_t, B_t)\right] - R(T). \end{aligned}$$

Proof. First, note that

$$g_i(A_t, B_t) \geq f_{it}(\hat{b}_i(t)). \tag{1}$$

This is trivially true when $f_{it}(\hat{b}_i(t)) = 0$. When $f_{it}(\hat{b}_i(t)) = 1$, $G_{i-1}(A_t, B_t) = 0$ and $\hat{b}_i(t) \in A_t$. This implies that $b_i(t) = \hat{b}_i(t)$ and that $g_i(A_t, B_t) = 1$.

Now using the regret bound for MAB_i we obtain

$$\begin{aligned} \sum_{t=1}^{T} \mathbf{E}[f_{it}(\hat{b}_i(t))] &\geq \max_b \sum_{t=1}^{T} \mathbf{E}[f_{it}(b)] - R(T) \\ &\geq \frac{1}{k}\mathbf{E}\left[\sum_{b \in B^*} \sum_{t=1}^{T} f_{it}(b)\right] - R(T). \end{aligned} \tag{2}$$

To complete the proof of the lemma, we will prove that

$$\sum_{b \in B^*} f_{it}(b) \geq G_k(A_t, B^*) - G_{i-1}(A_t, B_t). \tag{3}$$

The lemma follows immediately by combining (1)-(3). Observe that the left side of (3) is a non-negative integer, while the right side takes one of the values $\{-1, 0, 1\}$. Thus, to prove (3) it suffices to show that the left side is greater than or equal to 1 whenever the right side is equal to 1. The right side equals 1 only when $G_{i-1}(A_t, B_t) = 0$ and A_t intersects B^*. In this case it is clear that there exists at least one $b \in B^*$ such that $f_{it}(b) = 1$, hence the left side is greater than or equal to 1. □

Theorem 1. *The algorithm's combined payoff after T rounds satisfies:*

$$\mathbf{E}\left[\sum_{t=1}^{T} G_k(A_t, B_t)\right] \geq \left(1 - \frac{1}{e}\right) OPT - kR(T). \tag{4}$$

Proof. We will prove, by induction on i, that

$$OPT - \mathbf{E}\left[\sum_{t=1}^{T} G_i(A_t, B_t)\right] \leq \left(1 - \frac{1}{k}\right)^i OPT + iR(T). \tag{5}$$

The theorem follows by taking $i = k$ and using the inequality $\left(1 - \frac{1}{k}\right)^k < \frac{1}{e}$.

In the base case $i = 0$, inequality (5) is trivial. For the induction step, let

$$Z_i = OPT - \mathbf{E}\left[\sum_{t=1}^{T} G_i(A_t, B_t)\right].$$

We have

$$Z_i = Z_{i-1} - \mathbf{E}\left[\sum_{t=1}^{T} g_i(A_t, B_t)\right], \tag{6}$$

and Lemma 1 says that

$$\mathbf{E}\left[\sum_{t=1}^{T} g_i(A_t, B_t)\right] \geq \frac{1}{k} Z_{i-1} - R(T). \tag{7}$$

Combining (6) with (7), we obtain

$$Z_i \leq \left(1 - \frac{1}{k}\right) Z_{i-1} + R(T).$$

Combining this with the induction hypothesis proves (5). □

The general case, in which user u_i's type vector $(p_{ij})_{j \in \mathcal{D}}$ is an arbitrary element of $[0, 1]^{\mathcal{D}}$, can be reduced via a simple transformation to the case of deterministic users analyzed above. We replace user u_i with a *random* deterministic user $\hat{u}_i$ whose type vector $\hat{p}_i \in \{0, 1\}^{\mathcal{D}}$ is sampled using the following rule: the random variable $\hat{p}_{ij}$ has distribution

$$\hat{p}_{ij} = \begin{cases} 1 & \text{with probability } p_{ij} \\ 0 & \text{with probability } 1 - p_{ij}, \end{cases}$$

and these random variables are mutually independent. Note that the clicking behavior of user u_i when presented with a ranking B is *identical* to the clicking behavior observed when a random user type $\hat{u}_i$ is sampled from the above distribution, and the ranking B is presented to this random user. Thus, if we apply the specified transformation to users $u_1, u_2, \ldots, u_T$, obtaining a random sequence $\hat{u}_1, \hat{u}_2, \ldots, \hat{u}_T$ of deterministic users, this transformation changes neither the algorithm's expected payoff nor that of the optimum ranking B^*. Thus, Theorem 1 for general users can be deduced by applying the same theorem to the random sequence $\hat{u}_1, \ldots, \hat{u}_T$ and taking the expectation of the left and right sides of (4) over the random choices involved in sampling $\hat{u}_1, \ldots, \hat{u}_T$.

Note also that B^* is defined as the optimal *subset* of k documents, and OPT is the payoff of presenting B^*, without specifying the order in which documents are presented. However, the Ranked Bandits Algorithm learns an order for the documents in addition to identifying a set of documents. In particular, given $k' < k$, RBA(k') would receive exactly the same feedback as the first k' instances of MAB_i receive when running RBA(k). Hence any k' sized prefix of the learned ranking also has the same performance bound with respect the appropriate smaller set B'^*.

Finally, it is worth noting that this analysis *cannot* be trivially extended to non-binary payoffs, for example when learning a ranking of web advertisements. In particular, the greedy algorithm on which RBA is based in the non-binary payoff case can obtain a payoff that is a factor of $k - \varepsilon$ below optimal, for any $\varepsilon > 0$.

5.3. Analysis of Ranked Explore and Commit

The analysis of the Ranked Explore and Commit (REC) algorithm is analogous to that of the Ranked Bandits algorithm, except that the equivalents of Lemma 1 and Theorem 1 are only true with high probability after $t_0 = nxk$ time steps of exploration have occurred. Let B denote the ranking selected by REC.

Lemma 2. *Let $x = 2k^2/\epsilon^2 \log(2k/\delta)$. Assume A_t is drawn i.i.d. from a fixed distribution of user types. For any i, with probability $1 - \delta/k$,*

$$\mathbf{E}\left[\sum_{t=t_0}^{T} g_i(A_t, B)\right] \geq \frac{1}{k}\mathbf{E}\left[\sum_{t=t_0}^{T} (G_k(A_t, B^*) - G_{i-1}(A_t, B))\right] - \frac{\epsilon}{k}T.$$

Proof Outline. First note that in this setting, B^* and OPT are defined in expectation over the A_t drawn. For any document, by Hoeffding's inequality, with probability $1 - \delta/2k$ the true payoff of that document explored at rank i is within $\epsilon/2k$ of the observed mean payoff. Hence the document selected at rank i is within ϵ/k of the payoff of the best document available at rank i. Now, the same proof as for Lemma 1 applies, although with a different regret $R(T)$. □

Theorem 2. *With probability $(1 - \delta)$, the algorithm's combined payoff after T rounds satisfies:*

$$\mathbf{E}\left[\sum_{t=1}^{T} G_k(A_t, B)\right] \geq \left(1 - \frac{1}{e}\right) OPT - \epsilon T - nkx \quad (8)$$

Proof Outline. Applying Lemma 2 for all $i \in \{1, .., k\}$, with probability $(1 - k\delta/k) = (1 - \delta)$ the conclusion of the Lemma holds for all i.

Next, an analogous proof as for Theorem 1 applies, except replacing $R(T)$ with $\frac{\epsilon}{k}T$ and noting that the regret during the nkx exploration steps is at most 1 for every time step. □

It is interesting to note that, in contrast to the Ranked Bandits Algorithm, this algorithm can be adapted to the case where clicked documents provide real valued payoffs. The only modification necessary is that documents should always be presented by decreasing payoff value. However, we do not address this extension further due to space constraints.

6. Evaluation

In this section, we evaluate the Ranked Bandits and Ranked Explore and Commit algorithms, as well as two variants of RBA, with simulations using a user and document model.

We chose a model that produces a user population and document distribution designed to be realistic yet allow us to evaluate the performance of the presented algorithms under different levels of noise in user clicking behavior. Our model first assigns each of 20 users to topics of interest using a Chinese Restaurant Process (Aldous, 1985) with parameter $\theta = 3$. This led to a mean of 6.5 unique topics, with topic popularity decaying according to a power law. Taking a collection of 50 documents, we then randomly assigned as many documents to each topic as there were users assigned to the topic, leading to topics with more users having more relevant documents. We set each document assigned to a topic as relevant to all users assigned to that topic, and all other documents as non relevant. The probabilities of a user clicking on relevant and non-relevant documents were set to constants p_R and p_{NR} respectively.

We tested by drawing one user uniformly from the user population at each time step, and presented this user with the ranking selected by each algorithm, using $k = 5$. We report the average number of time steps where the user clicked on a result, and the average number of time steps where at least one of the presented documents was relevant to the user. All numbers we report are averages over 1,000 algorithm runs.

6.1. Performance Without Click Noise

We start by evaluating how well the REC and RBA algorithms maximize the clickthrough rate in the simplest case when $p_R = 1$ and $p_{NR} = 0$. We also compare their performance to the clickthrough rate that the same users would generate if presented with a static system that orders documents by decreasing true prob-

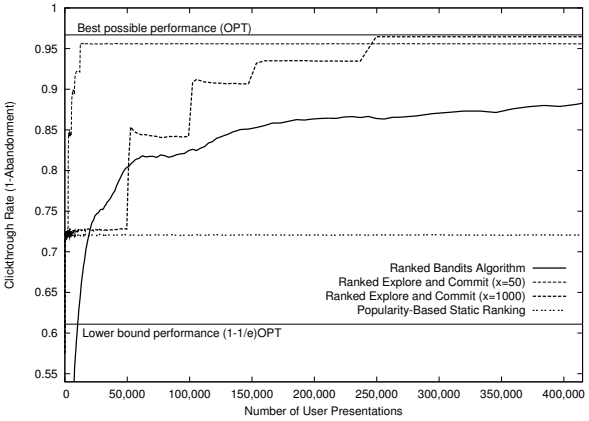

Figure 1. Clickthrough rate of the learned ranking as a function of the number of times the ranking was presented to users.

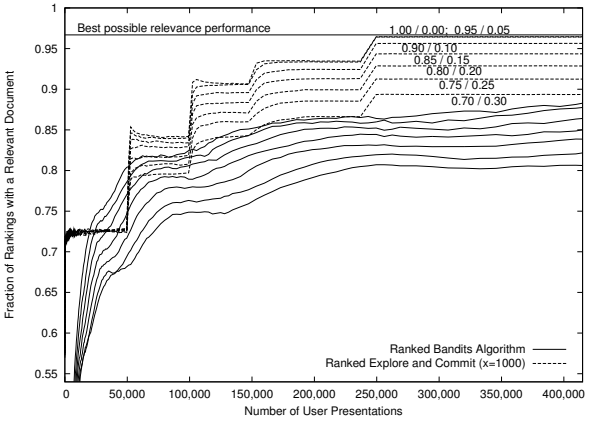

Figure 2. Effect of noise in clicking behavior on the quality of the learned ranking.

ability of relevance to the users assuming document relevances are independent. Figure 1 shows that both REC and RBA perform well above the static baseline and well above the performance guarantee provided by the theoretical results. This is not surprising, as the $(1 - 1/e)OPT$ bound is a worst-case bound. In fact, we see that REC with $x = 1000$ nearly matches the performance of the best possible ranking after finishing its initial exploration phase. We also see that the exploration parameter of REC plays a significant role in the performance, with lower exploration leading to faster convergence but slightly lower final performance. Note that despite REC performing best here, the ranking learned by REC is fixed after the exploration steps have been performed. If user interests and documents change over time, the performance of REC could fall arbitrarily. In contrast, RBA is guaranteed to remain near or above the $(1 - 1/e)OPT$ bound.

6.2. Effect of Click Noise

In Figure 1, the clickthrough rate and fraction of users who found a relevant document in the top k positions is identical (since users click if and only if they are presented with a relevant document). In contrast, Figure 2 shows how the fraction of users who find a relevant document decays as the probability of a user clicking becomes noisier. The figure presents the performance lines for REC and RBA across a range of click probabilities, from $(p_R = 1, p_{NR} = 0)$ to $(p_R = 0.7, p_{NR} = 0.3)$. We see that both algorithms decay gracefully: as the clicks become noisier noisy, the fraction of users presented with a relevant documents decays slowly.

6.3. Optimizing Practical Effectiveness

Despite the theoretical results shown earlier, it would be surprising if an algorithm designed for the worst case had best average case performance. Figure 3 shows the clickthrough rate (which the algorithms optimize), and fraction of users who find relevant documents (which is of more interest to information retrieval practitioners), for variants building on the insights of the ranked bandits idea. Specifically, two variants of RBA that have the best performance we could obtain in our simulation are shown. We found that using a UCB1-based multi-armed bandit algorithm (Auer et al., 2002a) in place of EXP3 improves the performance of RBA substantially when user interests are static. Note however, that UCB1 does not satisfy the constraints presented in Section 4.2 because it assumes rewards are identically distributed over time, an assumption violated in our setting when changes in the documents presented above rank i alter the reward distribution at rank i. Nevertheless, we see that this modification substantially improves the performance of RBA. We expect such an algorithm to perform best when few documents are prone to radical shifts in popularity.

7. Conclusions and Extensions

We have presented a new formulation of the learning to rank problem that explicitly takes into account the relevance of different documents being interdependent. We presented, analyzed and evaluated two algorithms and two variants for this learning setting. We have shown that the learning problem can be solved in a theoretically sound manner, and that our algorithms can be expected to perform reasonably in practice.

We plan to extend this work by addressing the non-binary document relevance settings, and perform empirical evaluations using real users and real documents. Furthermore, we plan to investigate how prior knowledge can be incorporated into the algorithms to improve speed of convergence. Finally, we plan to inves-

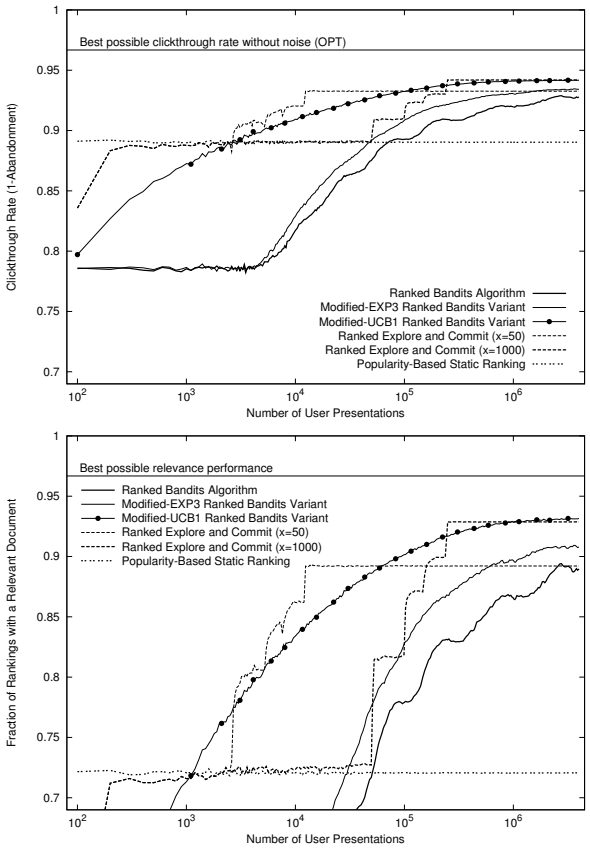

Figure 3. In a practical setting, it may be beneficial to use a variant of RBA to obtain improved performance at the cost of weaker theoretical guarantees. Performance is shown in realistic settings $p_R = 0.8$, $p_{NR} = 0.2$.

tigate if the bandits at different ranks can be coupled to improve the rate at which RBA converges.

Acknowledgments

We would like to thank the reviewers for helpful comments. This work was supported by NSF Career Award CCF-0643934, NSF Award CCF-0729102, NSF Career Award 0237381 and a gift from Google. The first author was supported by a Microsoft Research Fellowship.

References

Agichtein, E., Brill, E., & Dumais, S. (2006). Improving web search ranking by incorporating user behavior. *In SIGIR* (pp. 19–26).

Aldous, D. J. (1985). Exchangeability and related topics. *École d'Été de Probabilités de Saint-Flour XIII* (pp. 1–198).

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002a). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, *47*, 235–256.

Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (2002b). The non-stochastic multi-armed bandit problem. *SIAM Journal of Computing*, *32*, 48–77.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval*. New York, NY: Addison Wesley.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. *In ICML* (pp. 89–96).

Burges, C. J. C., Ragno, R., & Le, Q. V. (2006). Learning to rank with nonsmooth cost functions. *In NIPS* (pp. 193–200). MIT Press.

Carbonell, J., & Goldstein, J. (1998). The use of MMR, diversity-based reranking for reordering documents and producing summaries. *In SIGIR* (pp. 335–336).

Chen, H., & Karger, D. R. (2006). Less is more: Probabilistic models for retrieving fewer relevant documents. *In SIGIR* (pp. 429–436).

Chu, W., & Ghahramani, Z. (2005). Gaussian processes for ordinal regression. *Journal of Machine Learning Research*, *6*, 1019–1041.

Herbrich, R., Graepel, T., & Obermayer, K. (2000). Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers* (pp. 115–132).

Joachims, T. (2002). Optimizing search engines using clickthrough data. *In KDD* (pp. 132–142).

Khuller, S., Moss, A., & Naor, J. (1997). The budgeted maximum coverage problem. *Information Processing Letters*, *70*, 39–45.

Metzler, D., & Croft, W. B. (2005). A markov random field model for term dependencies. *In SIGIR* (pp. 472–479).

Nemhauser, G. L., Wolsey, L. A., & Fisher, M. L. (1978). An analysis of approximation for maximizing submodular set functions. *Mathematical Programming*, *14*, 265–294.

Radlinski, F., & Joachims, T. (2005). Query chains: Learning to rank from implicit feedback. *In KDD* (pp. 239–248).

Radlinski, F., & Joachims, T. (2007). Active exploration for learning rankings from clickthrough data. *In KDD* (pp. 570–579).

Robertson, S. E. (1977). The probability ranking principle in IR. *Journal of Documentation*, *33*, 294–304.

Streeter, M., & Golovin, D. (2007). *An online algorithm for maximizing submodular functions* (Technical Report CMU-CS-07-171). Carnegie Mellon University.

Taylor, M. J., Guiver, J., Robertson, S. E., & Minka, T. (2008). Softrank: Optimizing non-smooth ranking metrics. *In WSDM* (pp. 77–86).

Teevan, J., Dumais, S. T., & Horvitz, E. (2007). Characterizing the value of personalizing search. *In SIGIR* (pp. 757–758).

Turpin, A., & Scholer, F. (2006). User performance versus precision measures for simple search tasks. *In SIGIR* (pp. 11–18).

Yue, Y., Finley, T., Radlinski, F., & Joachims, T. (2007). A support vector method for optimizing average precision. *In SIGIR* (pp. 271–278).

Zhai, C., Cohen, W. W., & Lafferty, J. (2003). Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval. *In SIGIR* (pp. 10–17).

Zhang, B., Li, H., Liu, Y., Ji, L., Xi, W., Fan, W., Chen, Z., & Ma, W.-Y. (2005). Improving web search results using affinity graph. *In CIKM* (pp. 504–511).

Zhu, X., Goldberg, A. B., Gael, J. V., & Andrzejewski, D. (2007). Improving diversity in ranking using absorbing random walks. *Proceedings of NAACL HLT*.