Relative Confidence Sampling for Efficient On-Line Ranker Evaluation

Masrour Zoghi, Shimon Whiteson, Maarten de Rijke
ISLA, University of Amsterdam
Amsterdam, The Netherlands
m.zoghi, s.a.whiteson, derijke@uva.nl

Remi Munos
INRIA Lille - Nord Europe
Villeneuve d'Ascq, France
remi.munos@inria.fr

ABSTRACT

A key challenge in information retrieval is that of *on-line ranker evaluation*: determining which one of a finite set of rankers performs the best in expectation on the basis of user clicks on presented document lists. When the presented lists are constructed using *interleaved comparison methods*, which interleave lists proposed by two different candidate rankers, then the problem of minimizing the total *regret* accumulated while evaluating the rankers can be formalized as a *K-armed dueling bandits problem*. In this paper, we propose a new method called *relative confidence sampling* (RCS) that aims to reduce cumulative regret by being less conservative than existing methods in eliminating rankers from contention. In addition, we present an empirical comparison between RCS and two state-of-the-art methods, *relative upper confidence bound* and *SAVAGE*. The results demonstrate that RCS can substantially outperform these alternatives on several large learning to rank datasets.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords

Evaluation; implicit feedback; on-line learning

1. INTRODUCTION

An important challenge in information retrieval is that of *ranker evaluation*: determining which of a finite set of rankers performs best in expectation. In *off-line* ranker evaluation, which goes back to the early Cranfield experiments [8], rankers are assessed based on a fixed set of queries and documents manually judged by human assessors. Unfortunately, obtaining such judgments is expensive and error prone. Because the assessors typically did not formulate the queries on which they judge documents, their assessments may not accurately reflect how well those documents meet the needs of real users.

WSDM'14, Feb. 24–28, 2014, New York City, NY, USA.
ACM 978-1-4503-2351-2/14/02 ...$15.00.
http://dx.doi.org/10.1145/2556195.2556256.

These difficulties can be addressed by *on-line* ranker evaluation, in which rankers are assessed using click feedback from actual users. A common approach is to use *interleaved comparison methods* [6, 7, 13, 15, 19, 26, 28], in which the document lists proposed by two candidate rankers for a given query are interleaved and the resulting list presented to the user, whose clicks are used to infer a noisy preference for one ranker over the other. Recently, interleaving methods have been successfully applied in large-scale settings [6, 7].

The use of interleaved comparison methods gives rise to a key challenge in ranker evaluation: how to efficiently determine the best ranker from a set using only pairwise comparisons. This challenge can be formalized as a *K-armed dueling bandits* problem [37], wherein the best ranker is defined as the one that in expectation wins an interleaved comparison with each other ranker.

In the *explore-then-exploit* setting [37], the algorithm's goal is defined with respect to a fixed period of time called the *horizon*, at the end of which the algorithm must report what it thinks the best ranker is. Performance is then evaluated according to both the *accuracy* with which it picks the best ranker and the cumulative *regret* it accrues during that period. Regret is a measure of the suboptimality, with respect to the best ranker, of the rankers selected for interleaving. Several algorithms have been proposed for this setting, including *interleaved filter* (IF) [37], *beat the mean* (BTM) [36], and *sensitivity analysis of variables for generic exploration* (SAVAGE) [32].

By contrast, in the *ongoing regret minimization* setting [23], no horizon is specified in advance and the goal is to minimize cumulative regret for *all* horizons. The *relative upper confidence bounds* (RUCB) [38] algorithm was recently proposed for this setting.

In this paper, we propose and evaluate a new method for evaluating rankers, called *relative confidence sampling* (RCS). To determine which rankers to interleave at each iteration, RCS proceeds in two phases. First, it uses the results of the comparisons conducted so far to simulate a round-robin tournament among the rankers. Second, the champion of this tournament is compared against a challenger deemed to have the best chance of beating it. As more comparisons are conducted, the best ranker is increasingly likely to be selected as both champion and challenger, causing regret to fall steeply over time.

To validate RCS, we evaluate its performance on large Microsoft and Yahoo! learning to rank datasets using both the explore-then-exploit and the ongoing regret minimization settings. Our results show that RCS significantly and

substantially outperforms both SAVAGE, the state of the art in the explore-then-exploit setting, and RUCB, the state of the art in ongoing regret minimization.

The main contributions of this paper are thus:

1. We evaluate two existing K-armed dueling bandits algorithms on online ranker evaluation using two large learning to rank datasets, and
2. We propose and evaluate a new K-armed dueling bandits algorithm that performs better than existing state-of-the-art algorithms for online ranker evaluation.

In Section 2, we detail our problem setting and in Section 3 we discuss existing methods for ranker evaluation. Section 4 details relative confidence sampling, our new method for online ranker evaluation. Section 5 details our experimental setup and Section 6 presents our experimental results and discusses our findings. Section 7 contains our conclusions.

2. PROBLEM SETTING

One approach to ranker evaluation is to use manual expert annotations in a TREC-like setting [34]. However, collecting the necessary annotations is expensive and, because they are not based on real users, such annotations may not reflect real users' actual information needs. An attractive alternative is thus to evaluate rankers *on-line* using feedback from real users. One way to obtain such feedback is to measure the *click-through rate* [17, 18]. However, such feedback is often unreliable, since click-through rates can have substantial variance, particularly across users and topics [20, 21, 28, 29].

Fortunately, on-line feedback can also be obtained using *interleaved comparison methods*, which give *relative* feedback about how one ranker compares to another and has been shown to be more reliable [6, 26]. To compare two rankers on a given query, an interleaved comparison constructs a ranking that is an amalgamation of the rankings proposed by the two rankers for that query. Schemes for constructing the amalgamation include *balanced interleave* [19], *team draft* [28], *document constraints* [13], *probabilistic interleave* [15], and *optimized interleave* [27].

However, since interleaved comparison methods require feedback from real users, each comparison has significant real-world costs, i.e., if either of the compared rankers is poor, the interleaved ranking may also be poor, leaving the user dissatisfied. An important question is thus how to find the best ranker in a way that minimizes these costs.

This problem can be formalized as a *K-armed dueling bandits* problem [37], which is itself an extension of the *K-armed bandits* problem [2]. In the K-armed bandits problem, there are K rankers, $\{\rho_1, \ldots, \rho_K\}$. At each *time step*, a ranker ρ_i can be tried, generating a *reward* drawn from an unknown stationary distribution with expected value μ_i, which might be a quantity like click-through rate [5].

The K-armed *dueling* bandits problem is a variation that models the relative feedback available in settings like ranker evaluation with interleaved comparison methods. The problem is defined by a matrix of comparison probabilities $\mathbf{P} = [p_{ij}]$. At each *time step*, two rankers (ρ_i, ρ_j) are compared, e.g., using an interleaved comparison method, and with probability p_{ij} ranker ρ_i beats ρ_j. In this paper, we assume that there exists a *Condorcet winner* [32]: a ranker, which without loss of generality we label ρ_1, such that $p_{1i} > \frac{1}{2}$ for all $i > 1$. In other words, when interleaved with any other ranker, the Condorcet winner is expected to win.

The Condorcet winner is different in a subtle but important way from the *Borda winner* [32], which is a ranker ρ_b that satisfies

$$\sum_j p_{bj} \geq \sum_j p_{ij}, \text{ for all } i = 1, \ldots, K.$$

In other words, when averaged across all other rankers, the Borda winner is the ranker with the highest probability of winning a given comparison.

To better understand the distinction between these two types of winners, consider an example from the world of sports. Suppose we wish to use a K-armed dueling bandit method to efficiently identify the world's best tennis player. In addition, suppose that Rafael Nadal is likely to win, say with probability 0.55, a match against the other top players since, according to The New York Times (June 9, 2013) "Nadal is also the only member of the so-called Big Four to have a head-to-head edge over all the other members of that club: [Novak] Djokovic, Roger Federer and Andy Murray." Finally, suppose also that, though Federer loses in expectation to Nadal, he has a 0.75 probability of beating Djokavic and Murray. Thus, Nadal is the Condorcet winner but Federer is the Borda winner.

It is not immediately obvious which player should be considered the best. While Nadal is the only undominated player, a gambler who could only bet on one player before the start of a tournament should pick Federer. However, in ranker evaluation, the Condorcet winner is the more suitable choice. The reason is that, although Federer is more likely than Nadal to win a given match, the goal of ranker evaluation is not to win interleaved comparisons but to identify and use the best ranker. Interleaving is thus only a means to an end, whereas in tennis the match is an end in itself. Choosing anything other than the Condorcet winner as the best ranker is suboptimal because it means that by definition another ranker exists that would beat the chosen ranker. More precisely, given a distribution over user-query pairs encountered by the retrieval system and a ranker ρ_i other than the Condorcet winner ρ_1, then ρ_1 is morely likely to win an interleaved comparison against ρ_i for a given user-query pair than vice-versa. In Section 3, we compare the Condorcet assumption, i.e., the assumption of the existences of a Condorcet winner, with the typically much stronger assumptions made by other K-armed dueling bandits algorithms.

The goal of an algorithm for the K-armed dueling bandits problem can be formalized in several ways. In this paper, we consider two standard settings:

1. *Explore-then-exploit* [37]: In this setting, the algorithm is told in advance the exploration *horizon*: the number of time steps that the evaluation process is given to *explore* before it has to produce a single ranker as the best, which will be *exploited* thenceforth. The algorithm is assessed on its *accuracy*, the probability that a given run of the algorithm reports the Condorcet winner as the best ranker, as well as the amount of *regret* accumulated during the exploration phase, where regret for each time step is defined as follows: if rankers ρ_i and ρ_j were chosen for comparison at time t, then regret at that time is set to be

$$r_t := \frac{\Delta_{1i} + \Delta_{1j}}{2}, \tag{1}$$

with $\Delta_{1k} := p_{1k} - \frac{1}{2}$ for all $k \in \{1, \ldots, K\}$. Thus, regret measures the average advantage that the Condorcet winner has over the two interleaved rankers. Given our assumption on the probabilities p_{1k}, this implies that $r = 0$ if and only if the best ranker is interleaved with itself.

2. *Ongoing regret minimization* [23]: In this setting, no horizon is specified and the evaluation process continues indefinitely. Thus, it is no longer sufficient for the algorithm to maximize accuracy and minimize regret after a single horizon is reached. Instead, it must minimize regret across *all* horizons by rapidly decreasing the frequency of interleaved comparisons involving suboptimal rankers, particularly those that fare worse in comparison to the best ranker. This goal can be formulated as minimizing the cumulative regret over time, rather than with respect to a fixed horizon.

As we describe below in Section 3, most existing K-armed dueling bandits methods target the explore-then-exploit setting. However, we contend that the ongoing regret minimization setting better captures the key challenges of ranker evaluation on a deployed retrieval system for the following reason: explore-then-exploit methods require a horizon as input and behave differently for different horizons. This poses a practical problem because it is typically difficult to know in advance how many comparisons are required to determine the best ranker with confidence and thus how to set the horizon. If the horizon is set too long, the algorithm is too exploratory, increasing the number of evaluations needed to find the best ranker. If it is set too short, the best ranker remains unknown when the horizon is reached and the algorithm must be restarted with a longer horizon.

3. RELATED WORK

In this section, we briefly survey existing methods for the K-armed dueling bandits problem, considering both explore-then-exploit and ongoing regret minimization methods.

3.1 Explore-then-Exploit Methods

To our knowledge, the first method for the K-armed dueling bandits problem that was designed for an explore-then-exploit scenario is *interleaved filter* (IF) [37], which proceeds as follows: a ranker $\hat{\rho}$ is randomly chosen to be compared against all other rankers in sequence; these comparisons are repeated until another ranker ρ' either loses to or beats $\hat{\rho}$ by a wide margin, i.e., the winner scores so many more wins over the loser that one can conclude with a high level of certainty that the loser can be eliminated. If $\hat{\rho}$ is the winner, then ρ' is eliminated from the pool of rankers and is not compared against any other rankers. If $\hat{\rho}$ is the loser, then it is eliminated from the pool of rankers and ρ' becomes the new $\hat{\rho}$. This process continues until either all but one of the rankers is eliminated or the horizon is reached. A variation of the algorithm eliminates not just $\hat{\rho}$ when it loses badly to ρ', but also any other ranker that has lost to $\hat{\rho}$ on average (not necessarily by a wide margin).

More recently, the *beat the mean* (BTM) algorithm has been shown to perform better than IF [36]. BTM works by focusing exploration on the rankers that have been involved in the fewest comparisons. When it determines that a ranker fares on average too poorly in comparison to the remaining rankers, it removes it from consideration. More precisely, BTM considers the performance of each ranker against the *mean ranker* by averaging the ranker's scores against all other rankers and uses these estimates to decide which ranker should be eliminated.

IF and BTM require the comparison probabilities p_{ij} to satisfy conditions that are difficult to verify without specific knowledge about the dueling bandits problem at hand. Specifically, IF and BTM require a *total ordering* $\{\rho_1, \ldots, \rho_K\}$ of the rankers to exist such that $p_{ij} > \frac{1}{2}$ for all $i < j$. In Section 6, we show that, in the datasets we consider, the probability of a total ordering existing decreases quickly as the number of rankers increases, due to cyclical relationships among rankers other than the Condorcet winner.

Moreover, in addition to this total ordering, these two algorithms require a form of *stochastic transitivity*. In particular, IF requires *strong stochastic transitivity*: for any triple (i, j, k), with $i < j < k$, the following condition is satisfied:

$$p_{ik} \geq \max\{p_{ij}, p_{jk}\},$$

BTM requies the less restrictive *relaxed stochastic transitivity*: there exists a number $\gamma \geq 1$ such that for all pairs (j, k) with $1 < j < k$, we have

$$\gamma p_{1k} \geq \max\{p_{1j}, p_{jk}\}.$$

As pointed out in [36], strong stochastic transitivity is often violated in practice, a phenomenon also observed in our experiments: half of the K-armed dueling bandits on which we experimented require $\gamma > 1$.

BTM permits a broader class of K-armed dueling bandits problems, but it requires γ to be explicitly passed to it as a parameter, which poses substantial difficulties in practice. If γ is underestimated, the algorithm can in certain circumstances be misled with high probability into choosing the Borda winner instead of the Condorcet winner, e.g., when the Borda winner has a larger average advantage over the remaining arms than the Condorcet winner, as in the Nadal-Federer example. On the other hand, though overestimating γ does not cause the algorithm to choose the wrong ranker, it nonetheless results in a severe penalty: if γ is multiplied by θ, then the number of interleaved comparisons required to eliminate suboptimal rankers from further consideration is multiplied by θ^4. Hence, overestimating γ by a factor of 2 means requiring 16 times as many comparisons, which can substantially affect the amount of regret accumulated.

By contrast, the algorithm we propose in Section 4 makes only the Condorcet assumption, which is implied by the total ordering assumption of IF and BTM. However, we show in Section 6.1 that there are many K-armed dueling bandits problems that satisfy the Condorcet assumption but do not have a total ordering.

Sensitivity Analysis of VAriables for Generic Exploration (SAVAGE) [32] is a recently proposed algorithm that outperforms both IF and BTM by a wide margin when the number of rankers is of moderate size. Moreover, one version of SAVAGE, which we call *Condorcet SAVAGE*, makes only the Condorcet assumption and performed the best experimentally [32]. Condorcet SAVAGE compares pairs of rankers uniformly randomly until there exists a pair for which one of the rankers beats the other by a wide margin, in which case the loser is removed from the pool of rankers under consideration. So far, the performance of Condorcet SAVAGE in the context of online learning to rank has not been evaluated on large-scale datasets. We show in this paper that

our proposed algorithm for ranker evaluation substantially outperforms Condorcet SAVAGE on such datasets.

3.2 Ongoing Regret Minimization Methods

To our knowledge, the recently proposed *relative upper confidence bound* (RUCB) [38] is the only existing K-armed dueling bandits method that explicitly targets the ongoing regret minimization objective. RUCB is based on a well-known method for the K-armed bandit problem, called the *upper confidence bound* (UCB) algorithm [2], which maintains optimistic estimates of the mean reward for each ranker and selects the ranker with the largest optimistic estimate. By doing so, UCB ensures adequate exploration of the rankers to avoid fixating on the wrong ranker and accumulating regret indefinitely.

In a similar fashion, RUCB maintains both optimistic and pessimistic estimates of each p_{ij} and uses them to choose which rankers to compare against each other. More specifically, it proceeds in two stages. First, it determines which rankers have a very low chance of being the best ranker by optimistically comparing each ranker against the rest: if they lose despite this optimism, they are omitted; one of the remaining rankers is chosen at random as the *champion*. Second, a pessimistic estimate is computed of how this purported champion would do against all possible opponents. The most formidable opponent is then selected as the *challenger*. Finally, the champion and the challenger are compared, i.e., the rankers are interleaved.

RUCB has a parameter α that controls how exploratory it is. If $\alpha > \frac{1}{2}$, then the regret accumulated by RUCB is proven to grow at most logarithmically with high probability [38]. In addition, it has been shown empirically to accumulate orders of magnitude less regret than BTM. Moreover, RUCB makes only the Condorcet assumption.

So far, the performance of RUCB in the context of online learning to rank has not been evaluated on large-scale datasets. We show that our proposed algorithm for ranker evaluation substantially outperforms RUCB on such datasets.

4. RELATIVE CONFIDENCE SAMPLING

In this section, we propose the *relative confidence sampling* (RCS) algorithm, which aims to reduce cumulative regret by being less conservative than the existing methods about eliminating rankers from contention. Though designed for the ongoing regret minimization setting, RCS is also easily adapted to the explore-then-exploit setting, and we show in Section 6 that it substantially outperforms the state-of-the-art methods in both settings.

While RCS is related to RUCB, which is also designed for the ongoing regret minimization setting, it differs in one crucial respect: the use of *sampling* when conducting a round-robin tournament to select a champion. The goal in doing so is to exploit one of the key lessons that has been learned in the study of regular K-armed bandits: that much better performance can be obtained by maintaining posterior distributions over the expected value of each ranker and sampling from those posteriors to determine which ranker to select. This is evidenced by the superior performance of Thompson Sampling [1, 22, 31], a K-armed bandit method that employs such sampling, over various UCB algorithms [5, 22]. Unlike UCB algorithms, which rely on estimates of the expected value that can be far off, sampling-based methods rely on posteriors that tend not to gravitate toward the extremes, leading to more appropriate choices more often.

Algorithm 1 Relative Confidence Sampling (RCS)

Input: A set of rankers $\rho_1, \ldots, \rho_K$ and an oracle that can take a pair of rankers and return one as the winner (e.g., an interleaved comparison method)

1: Choose $\alpha > \frac{1}{2}$
2: $\mathbf{W} \leftarrow \mathbf{0}_{K \times K}$ // *2D array of wins:* $\mathbf{W}_{ij}$ *is the number of times* ρ_i *beat* ρ_j
3: **for** $t = 1, 2, \ldots$ **do**
4: // **I***: Run a simulated "tournament":*
5: $\mathbf{\Theta}(t) \leftarrow \frac{\mathbf{1}_{K \times K}}{2}$
6: **for** $i, j = 1, \ldots, K$ with $i < j$ **do**
7: $\mathbf{\Theta}_{ij}(t) \sim \text{Beta}(\mathbf{W}_{ij} + 1, \mathbf{W}_{ji} + 1)$
8: $\mathbf{\Theta}_{ji}(t) = 1 - \mathbf{\Theta}_{ij}(t)$
9: **end for**
10: Pick c such that $\mathbf{\Theta}_{cj}(t) \geq \frac{1}{2}$ for all j. If no such ranker exists, pick the ranker that has been chosen champion least often.
11: // **II***: Run UCB in relation to c:*
12: $\mathbf{U}(t) = \frac{\mathbf{W}}{\mathbf{W}+\mathbf{W}^T} + \sqrt{\frac{\alpha \ln t}{\mathbf{W}+\mathbf{W}^T}}$ // *All operations are element-wise, with* $\frac{x}{0} := 1$ *for any* x.
13: $\mathbf{U}_{ii}(t) \leftarrow \frac{1}{2}$ for each $i = 1, \ldots, K$.
14: $d \leftarrow \arg\max_j \mathbf{U}_{jc}(t)$
15: // **III***: Update* $\mathbf{W}$
16: Compare rankers ρ_c and ρ_d and increment either $\mathbf{W}_{cd}$ if ρ_c beat ρ_d or $\mathbf{W}_{dc}$ otherwise.
17: **end for**

As we show in the experiments in Section 6, the advantage of sampling from the posterior distribution also appears in ranker evaluation and is in fact even more pronounced. RUCB tends to be very conservative in its choice of a potential champion, so unless it is highly confident that a ranker ρ_i is inferior to another ranker, it will go on considering ρ_i as a potential champion. By contrast, RCS is less timid in its choices: the more a ranker beats the rest, the greater its chances of being chosen as a champion to be compared against the rest.

The RCS algorithm, described in Algorithm 1, takes as input a set of rankers and an oracle such as an interleaved comparison method that can compare these rankers and return a noisy estimate of which is the winner. As it is horizonless, RCS does not have an output: as time goes by it chooses the best ranker more and more frequently. RCS has one parameter α (Line 1), which controls how exploratory the algorithm's behavior is: the higher the value of α, the more slowly the algorithm settles on a single ranker. RCS maintains a *scoresheet* $\mathbf{W}$ (Line 2), in which it records the comparison results and proceeds in two phases:

I: A tournament is simulated based on the current scoresheet, i.e., samples $\mathbf{\Theta}_{ij}$ are collected for each pair of rankers (i, j) with $i > j$, from the posterior Beta distribution maintained on p_{ij}; Since $p_{ji} = 1 - p_{ij}$, RCS sets $\mathbf{\Theta}_{ji} = 1 - \mathbf{\Theta}_{ij}$ (Lines 6-9). Also, RCS sets $\mathbf{\Theta}_{ii} = \frac{1}{2}$ for each ranker i, since $p_{ii} = \frac{1}{2}$ and thus its posteriors are all concentrated at $\frac{1}{2}$ (Line 5). Given these sampled results, ranker i beats ranker j in the simu-

lated tournament if $\mathbf{\Theta}_{ij} > \frac{1}{2}$ for $i \neq j$. There are two possibilities at this stage (Line 10):

1. There is a champion ranker c that beats all other rankers in this tournament, i.e., $\mathbf{\Theta}_{cj} > \frac{1}{2}$ for all $j \neq c$.
2. No ranker beats all other rankers, in which case RCS sets $c = \text{argmin}_i N_i$, where N_i is the number of times ranker i was previously chosen as champion. In other words, c is the ranker that has been the champion least often.

Eventually, once the Condorcet winner has been compared against the rest of the rankers often enough, its superiority over the rest will cause their elimination in this phase of the algorithm. So, as times goes by, c will be the Condorcet winner more and more often.

II: The UCB algorithm is applied to the K-armed bandit problem with means $\{p_{1c}, \ldots, p_{Kc}\}$ (Lines 12-14). In other words, for each $j \in \{1, \ldots, K\}$, we calculate the optimistic estimate

$$u_{jc} := \frac{\mathbf{W}_{jc}}{\mathbf{W}_{jc} + \mathbf{W}_{cj}} + \sqrt{\frac{\alpha \ln t}{\mathbf{W}_{jc} + \mathbf{W}_{cj}}},$$

where the first term is our estimate of the comparison probability p_{jc} and the second term is a confidence radius that is added to ensure adequate exploration by allowing the other rankers to compare themselves against c optimistically. RCS picks the ranker d for which u_{dc} is higher than all other u_{jc}.

Finally, rankers c and d are compared against each other using a real interleaved comparison and $\mathbf{W}$ is updated accordingly (Line 16).

To better understand the rationale behind RCS, consider our Nadal-Federer example from Section 2. There, a key danger is that a K-armed dueling bandits algorithm may mistakenly conclude that Federer is the champion and stop comparing him against other players, most importantly Nadal. RCS avoids this pitfall in two ways. First, if Nadal has not been compared against the others enough times, then $\mathbf{\Theta}_{NF}$, where N is Nadal and F is Federer, will have high variance so that in Phase **I**, Federer will have difficulty beating the others in the simulated tournament in order to be chosen as the champion. In cases where no player beats everyone in the simulated tournament, RCS ensures that everyone gets a chance at being the champion, including Nadal. This ensures that our posterior belief in Nadal's chance of beating all other players will further concentrate above 0.5, making it even more likely that he will win future simulated tournaments. Second, even in cases where Federer does win the simulated tournament in Phase **I**, the fact that Nadal and Federer have not been compared often means that the upper bound $\mathbf{U}_{NF}$ will very likely be above $\mathbf{U}_{FF} = \frac{1}{2}$, hence preventing a fruitless comparison between Federer and himself. Instead, Federer will be compared to Nadal, ensuring that Nadal's superiority is eventually discovered.

RCS can be easily adapted to the explore-then-exploit setting: when the horizon is reached, it picks any ranker that beats the greatest number of other rankers according to the final scoresheet. We call this modification RCS also, since it is clear from context which version is meant.

5. EXPERIMENTAL SETUP

We seek to answer the following questions:

RQ1 How often does the Condorcet assumption hold in comparison to the total ordering assumption?

RQ2 How do SAVAGE and RCS perform at ranker evaluation on large-scale learning to rank datasets in the explore-then-exploit setting?

RQ3 How do RUCB and RCS perform on these datasets in the ongoing regret minimization setting?

RQ4 How does RUCB perform when the parameter α is too small for its theoretical guarantees to hold? Moreover, how does RCS perform in the same range?

RQ5 How do RUCB and RCS scale as the number of rankers grows?

We address these questions using an experimental setup built on two large-scale learning to rank datasets: the Microsoft Learning to Rank (MSLR) and the Yahoo! Learning to Rank Challenge (YLR) datasets. The latter consists of two distinct subsets, Set 1 and Set 2, both of which we use. The specifics of these datasets are described in Table 1.

Table 1: The specifics of the datasets used.

Datasets	Queries	URLs	Features	Reference
MSLR-WEB30K	31,531	3,771,125	136	[24]
YLR Set 1	19,944	473,134	519	[4]
YLR Set 2	1,266	34,815	596	[4]

Using these datasets, we create a finite set of rankers, each of which corresponds to a ranking feature provided in the dataset, e.g., PageRank or BM25, and from this set we choose a subset to test our algorithms on. The ranker evaluation task thus corresponds to determining which single feature constitutes the best ranker.

The subsets were chosen by listing the rankers by their numbers and sorting them *alphabetically* and then taking the first K rankers: given the fact that, at least with the features in the MSLR dataset, nearby features tend to be similar to each other,[1] this way of picking rankers ensures that not all the selected rankers are extremely similar to each other but there are nonetheless some similar rankers in the selected set. Consequently, some diversity is introduced without making the problem too easy, as would be the case if the rankers were picked completely randomly.

To compare a pair of rankers, we use *probabilistic interleave* (PI) [14], though any other interleaved comparison method could be used instead. To model the user's click behavior on the resulting interleaved lists, we employ a probabilistic user model [9, 14] that uses as input the manual labels (classifying documents as relevant or not for given queries) provided with both datasets. Queries are sampled randomly and clicks are generated probabilistically by conditioning on these assessments using a user model that resembles the behavior of an actual user [11, 12]. This approach follows an experimental paradigm that has previously been used for assessing the performance of rankers [13–16].

[1]We assume that a similar phenomenon also holds for the Yahoo! datasets, but there is no specific list of features for them that we could use to verify that claim.

Because we use probabilistic interleave, which already introduces noise into the interleaved comparison, we employ the *perfect click model* in our experiments. While another click model that injects additional noise, such as the *navigational click model* or the *informational click model* [15], could be used instead, doing so would only shift each p_{ij} closer to 0.5. As a result, all methods would need more time to identify the best ranker, making our experiments take even longer to run.

We use two evaluation metrics, corresponding to the two objectives outlined in Section 2. We use *accuracy* (or best ranker rate) and *cumulative regret*, both of which are used in the explore-then-exploit setting, while only the latter is used for the ongoing regret minimization objective. Cumulative regret is the total amount of regret encountered by the algorithm until a given time, where regret at each time step is as defined in (1). A ranker evaluation algorithm accumulates regret when it makes a suboptimal choice, meaning that it does not interleave the best ranker with itself. The more suboptimal the rankers in the interleaved comparisons, the higher the accumulated regret. Thus, according to the ongoing regret minimization objective, the goal of the ranker evaluation algorithm is to increase the frequency with which it chooses the best ranker as soon as possible; doing so results in lower regret curves: the flatter the curve, the lower the frequency of picking poor rankers.

In most of our experiments, we set $\alpha = 0.501$ for both RUCB and RCS, as this ensures that RUCB's theoretical results apply without it being too exploratory. However, in one experiment, in order to address **RQ4**, we set $\alpha = 0.1$. In the experiments addressing **RQ2**–**RQ4**, the number of rankers that we consider is set to 10. In the addressing **RQ5**, we contrast the performance of RUCB and RCS with sets of 10, 20, 30 and 40 rankers.

6. RESULTS AND DISCUSSION

In this section, we present and discuss results concerning each of our research questions.

6.1 The Condorcet Assumption

RCS, like RUCB and Condorcet SAVAGE, relies on the Condorcet assumption. Other methods such as IF and BTM, rely on the even stronger assumption of total ordering, among others. Therefore, it is of interest to know to what extent these assumptions hold in learning to rank datasets. In order to address **RQ1**, we consider the 136 feature rankers in the MSLR-WEB30K dataset.

Using the full matrix of comparison probabilities $\mathbf{P}$, which was estimated by carrying out $80,000$ interleaved comparisons between each pair of feature rankers, and a simple combinatorial calculation, we determined that the probability that a random subset of these features satisfies the Condorcet assumption is always above 0.91. For small and large subsets, the probability is even higher.

The probability that the total ordering assumption holds was estimated using a sampling approach: for each $K = 1, \ldots, 136$, we randomly sampled $10,000$ sets of K feature rankers and used the proportion of these subsets that satisfy total ordering as an approximation to the probability that total ordering holds for a subset of the given size K.

Figure 1 shows these probabilities as a function of the size of the subset of features considered: the blue curve shows the probability that the Condorcet assumption is satisfied for a subset of given size, while the red curve shows the same for the total ordering assumption. In the case of the Condorcet assumption, these probabilities are slightly smaller for intermediate subset sizes, since it is harder for larger subsets to avoid cycles in the *superiority graph* (the directed graph that has an edge from i to j if $p_{ij} > 0.5$), while for very large subsets, the total number of possible subsets is relatively small, so the probability becomes large again. By contrast, the larger the subset of rankers under consideration, the more likely it is to include a cycle, causing the probability of a total ordering existing to fall quickly.

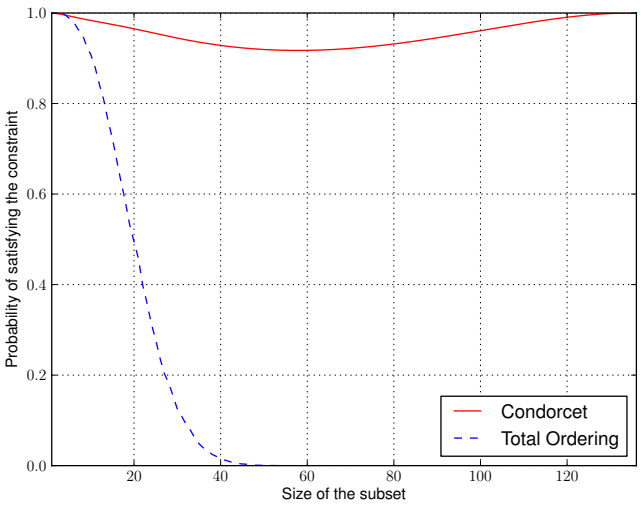

Figure 1: The probability that the Condorcet and the total ordering assumptions hold for subsets of the feature rankers in the MSLR-WEB30K dataset. The probability is shown as a function of the size of the subset.

One source of difficulty in this analysis are probabilities p_{ij} that are very close or equal to 0.5. Even with $80,000$ comparisons between each pair of rankers, our estimate of the preference matrix might contain fake loops that are simply due to the natural variance in frequentist estimates of such probabilities. This can lead to a reversal of the edges in the superiority graph. Indeed, for many cycles of size three in this graph, the "weakest link" is extremely close to 0.5, making it difficult to state with certainty that a cycle exists. However, the same variance can also break real cycles, and so these two phenomena should cancel each others' effects. Moreover, there are also many cycles whose edges are strong enough to deduce a genuine cycle.

6.2 Explore-then-Exploit Results

The performance of RCS in the explore-then-exploit setting was compared against Condorcet SAVAGE, which is the state of the art in this setting [32]. Recall from Section 2 that this setting considers both the accuracy and the cumulative regret of these algorithms at a given horizon. The top row of plots in Figure 2 compares the accuracy of RCS and Condorcet SAVAGE at 10 different horizons on three 10-armed bandits obtained from the three different datasets by considering 10 of the feature rankers. Note that the horizontal axis is in log scale and accuracy is the percentage of the runs that correctly produced the best ranker as the winner at the given horizon.

Note that, because Condorcet SAVAGE requires the horizon as input, the algorithm must be rerun from scratch for

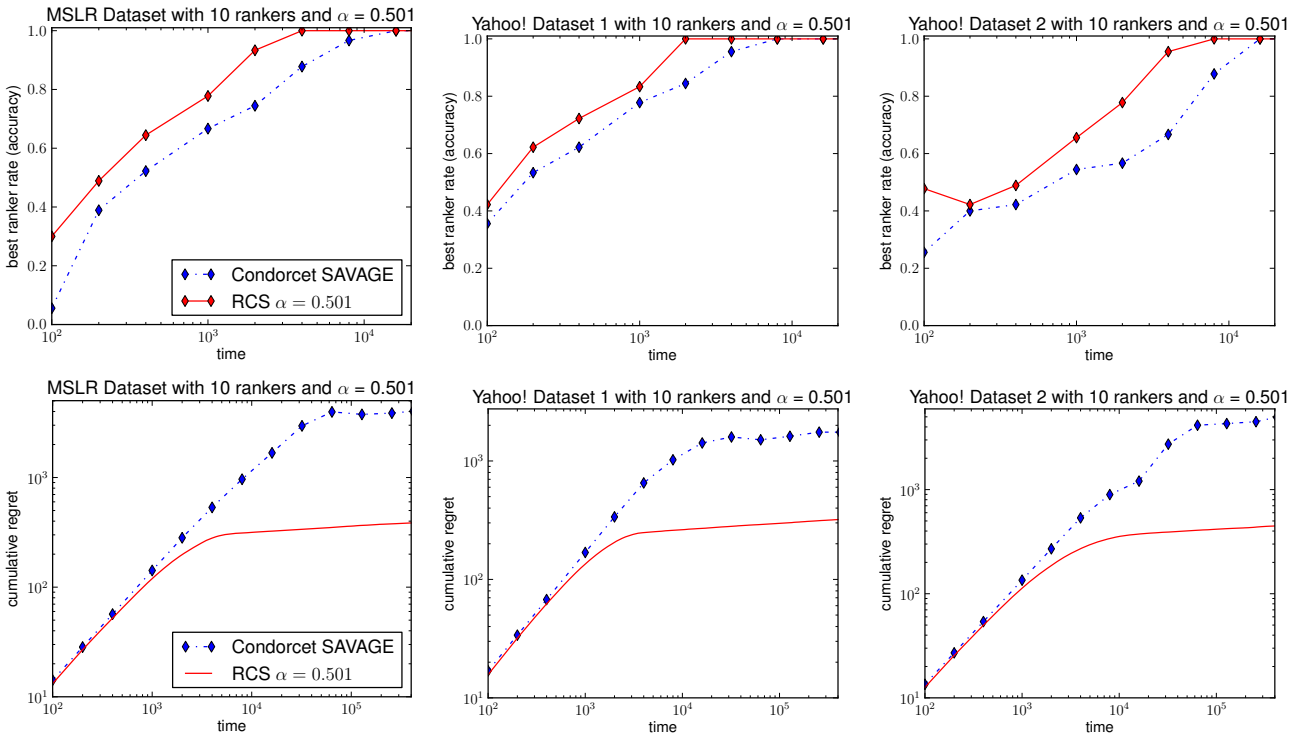

Figure 2: Accuracy (best ranker rate) and average cumulative regret over 90 runs; in top row of plots, the x-axis uses a log scale; in the bottom plots, both axes use log scales. Log scales were used for the x-axes to make the accuracy plots easier to read and in the y-axes of the regret plots since the regret accumulated by Condorcet SAVAGE is an order of magnitude higher than that of RCS.

each horizon. Thus, to obtain the plotted results, we conducted independent runs for each of the 10 horizons considered. By contrast, since RCS is a horizonless algorithm, we simply ran it until the longest horizon and then measured its accuracy at each of the 10 horizons.

Concerning **RQ2**, these results show that RCS has consistently higher accuracy than Condorcet SAVAGE on all datasets. This is a particularly striking result because RCS does not have any parameters optimized for the specific horizons in these experiments. Condorcet SAVAGE, by contrast, has the advantage that it receives the horizon as input and can thus adapt its behavior accordingly. Nonetheless, RCS outperforms Condorcet SAVAGE according to the metric for which Condorcet SAVAGE was designed. Though the lines appear close in the graph due to the log scale, the learning speed actually differs substantially: RCS reaches the same level of accuracy almost twice as quickly as Condorcet SAVAGE. For example, on the Yahoo! Set 1, RCS achieves an accuracy of 0.8 after 1000 steps while SAVAGE achieves it only after 2000 steps.

The other objective of the explore-then-exploit setting, which is to minimize the regret accumulated by the horizon, is addressed in the bottom row of plots in Figure 2. Note that these plots are in log-log scale. As with the accuracy plots, these plots select the performance of separate runs conducted at each horizon for Condorcet SAVAGE, while each RCS run was used to measure regret at all horizons. As is clear from these plots, RCS dramatically outperforms Condorcet SAVAGE despite being ignorant of the predetermined horizon. In fact, in each experiment, by the time Condorcet SAVAGE reaches 100% accuracy, it has accumulated at least 3 times as much regret as when RCS achieves that same accuracy.

The superior performance of RCS over Condorcet SAVAGE according to both accuracy and regret is due to related phenomena: the main advantage of RCS over Condorcet SAVAGE is that, instead of comparing pairs of rankers uniformly randomly, it rapidly focuses on comparing the Condorcet winner against the rest, thereby reducing regret, because one of the summands in the definition of regret (cf. (1)) is now zero and, more importantly, doing so leads to much better estimates of the probabilities p_{1j}, and thus greater confidence in the supremacy of the best ranker. This in turn leads to more frequent comparisons between the best ranker and itself, further reducing regret and increasing accuracy.

We also tested RUCB in the explore-then-exploit setting and found higher accuracy for RCS than RUCB, though the difference was relatively small. There was, however, a substantial difference in the regret performances of RUCB and RCS, as demonstrated in the next section. We omitted the accuracy curves for RUCB in order to increase readability.

6.3 Ongoing Regret Minimization Results

Turning now to the other method for evaluating K-armed dueling bandits algorithms, Figure 3 shows the expected cumulative regret obtained when applying RUCB and RCS to three 10-armed dueling bandits problems obtained from the three different datasets. For both RCS and RUCB, the parameter α is set to be 0.501 so that RUCB's theoretical guarantees [38] hold. The curves in the plots show the mean cumulative regret over 90 independent runs of each algorithm. The plots show results on the first 50,000 time steps, using linear scales on both axes.

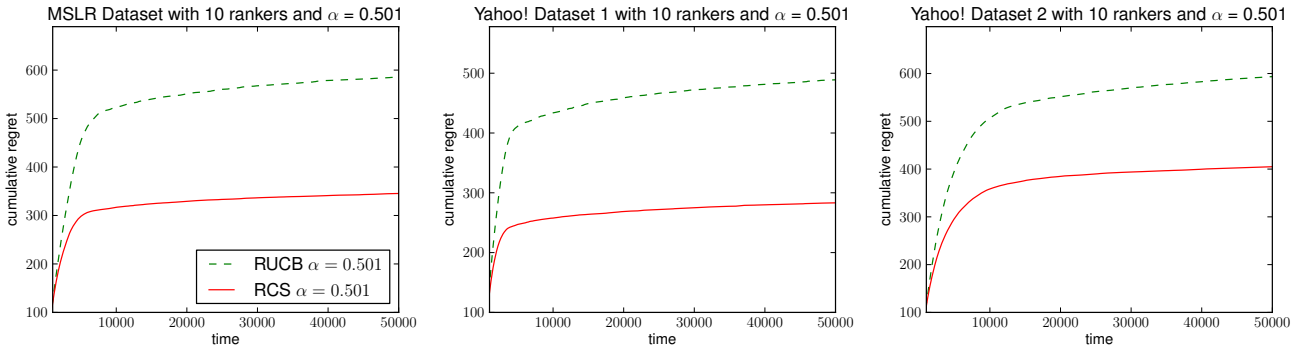

Figure 3: Cumulative regret averaged over 90 runs on the three datasets. All plots use axes with linear scale, since the two curves are much closer to each other than the ones in the regret plots in Figure 2.

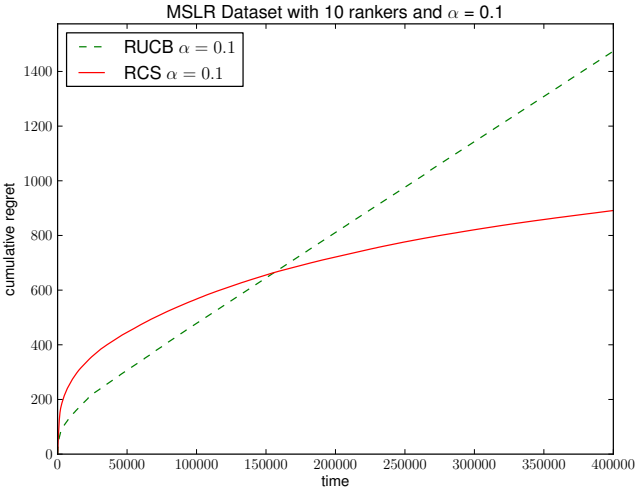

Figure 4: Cumulative regret averaged over 30 runs.

Regarding **RQ3**, the plots in Figure 3 clearly demonstrate that RCS accumulates substantially less regret than RUCB, with the former accumulating roughly a third less regret than latter. In other words, not only does RCS find the best ranker more quickly than RUCB, it also makes less severe errors in the process of doing so. In more concrete terms, the difference in the regret levels at which the two algorithms plateau is on the order of 200 in these three datasets, which roughly translates to an extra 2000 interleaved comparisons involving suboptimal rankers: this is because the probability with which the Condorcet winner beats the remaining arms is around 0.6. Needless to say, this performance difference can have a great impact on user satisfaction and engagement. Thus, these results highlight the benefits of a sampling-based approach to exploration.

Moreover, given the qualitative similarity between the performances of RUCB and RCS when $\alpha > 0.5$, we strongly suspect that similar regret bounds as those proven for RUCB [38] also hold for RCS. However, the use of sampling would necessitate a more intricate theoretical argument that we leave as future work.

Finally, in these experiments both algorithms had similar variance across runs; the best performing run of RUCB had a higher regret curve than the average regret curve of RCS.

6.4 Stability of RUCB and RCS

The improved performance of RCS over RUCB in the previous section can be attributed to the fact that RCS engages in less unnecessary exploration. Since lowering α makes both RCS and RUCB less exploratory, an important question, as posed in **RQ4**, is whether regret can be even further reduced by setting α to values below 0.5.

To address this question, we investigate the stability of RUCB and RCS by setting $\alpha = 0.1$, which lies outside the range permitted by RUCB's theoretical results. Figure 4 shows the cumulative regret results averaged over 30 runs of both RUCB and RCS on the MSLR dataset: fewer runs were used in this case to illustrate how easily RUCB can misbehave when α is below 0.5. These results show that the average cumulative regret for RUCB grows linearly, which was due to two of the runs never reaching the point where they keep interleaving the best ranker with itself. By contrast, though RCS accumulates almost twice as much regret at $\alpha = 0.1$ than $\alpha = 0.501$, the performance degradation is much less severe than for RUCB, with RCS's cumulative regret curve flattening much more. Similar results were also observed with the other datasets, but are not included here due to space constraints.

The performance difference is due to the fact that reducing α results in shrinking the confidence intervals maintained by RUCB, which results in the tournament phase of the algorithm not being exploratory enough. Hence, RUCB focuses prematurely on a single arm that has a temporary advantage over the others, preventing it from getting better estimates of the comparison probabilities between the Condorcet winner and the rest, which is necessary for the Condorcet winner to be chosen by the tournament. This, however, is not a stumbling block for RCS because there are no confidence intervals in the tournament phase, which relies on sampling instead. Figure 4 demonstrates that these samples ensure enough exploration to avoid getting stuck with a suboptimal ranker. Thus, while lower values of α are not beneficial to either algorithm, RCS remains stable while RUCB can experience the catastrophic negative performance associated with linear regret.

6.5 Size of the Set of Rankers

In order to study the issue of scalability, we compare RCS to RUCB on problems with 20, 30 and 40 rankers, all extracted from the MSLR dataset. See Figure 5.

Regarding **RQ5**, these results show that the cumulative regret curve of RCS flattens much sooner than that of RUCB. Thus, RCS starts focusing on the best ranker more quickly than RUCB: more specifically, where the two curves cross, RUCB was spending on average 6 to 9 times more iterations interleaving non-optimal rankers than RCS. For instance, in

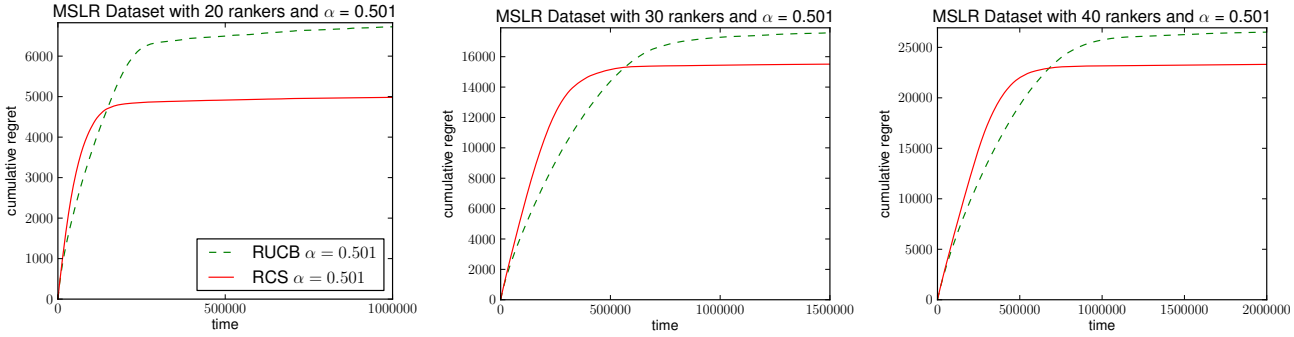

Figure 5: Cumulative regret averaged over 90 runs on the MSLR-WEB30K dataset, with 20, 30, 40 rankers. For comparison, the plot for 10 rankers is the leftmost plot in Figure 3; note that the scales differ between the four plots, which is necessary in order to illustrate the non-asymptotic portion of all of the results.

the experiments with 20 rankers, in the vicinity of the crossing point, 3.6% of RCS's comparisons involved subptimal rankers, whereas 27% of RUCB's did; the same quantities for the 30 and 40 ranker experiments are 1.9% vs. 17% and 3.3% vs. 21%. On the other hand, at time $T = \frac{T_0}{2}$, where T_0 is the time at which the two average regret curves cross, these differences are much smaller with the same numbers being 31% vs. 33%, 31% vs. 36% and 44% vs. 45% for the 20, 30 and 40 ranker experiments, respectively.

Note that this more rapid convergence to the best ranker requires more aggressive exploration early on. This can be deduced from the fact that, before plateauing, the red curves for RCS have slightly steeper slopes than the green curves for RUCB during the same period. This is due to the fact that RCS abstains from removing any rankers from consideration until it became clear which ranker is the best, whereas RUCB stops comparing poorer rankers earlier in the process. Nevertheless, RCS starts interleaving the best ranker with itself in substantially fewer iterations and thus accumulates much less regret in the long run. For instance, at time $T = 2T_0$, where T_0 is the time at which the two average regret curves cross, the average cumulative regret of RUCB is 30%, 12.3% and 12.6% higher than that of RCS for 20, 30 and 40 ranker experiments, respectively.

Moreover, the regret curves for RCS are much flatter after they plateau than those of RUCB, which means that, once the best ranker is identified, RCS is more likely to avoid futile interleaved comparisons with suboptimal rankers. More precisely, at time $T = 2T_0$, the percentage of comparisons RUCB devotes to suboptimal arms is still roughly 8 times higher than that of RCS, e.g., 2% and 0.3%, respectively, for the 20 ranker experiment.

7. CONCLUSIONS

We have proposed and experimentally evaluated a new method for addressing the problem of on-line ranker evaluation. Our method, *relative confidence sampling* (RCS), was evaluated against the existing methods on large scale learning to rank datasets, in two settings: explore-then-exploit and ongoing regret minimization. RCS significantly outperforms the existing state-of-the-art methods under both settings. In particular, given the need in online ranker evaluation scenarios to identify and compare the best ranker as quickly as possible, RCS has a large advantage over all existing algorithms, since when asked to return the best ranker in the explore-then-exploit setting, it has a higher probability of returning the best ranker, while minimizing the number of queries wasted on comparing suboptimal rankers (as evidenced by the lower regret curves for RCS), without requiring prior knowledge of the length of the evaluation or imposing restrictive assumptions such as a total ordering.

In future work, we aim to obtain theoretical guarantees for RCS, which require a considerably more delicate argument than that of RUCB because the use of sampling in RCS introduces a new dimension of uncertainty in the tournament phase of the algorithm.

Another important question is to design ranker evaluation algorithms that can deal with even larger (possibly infinite) numbers of rankers, without requiring convexity assumptions such as those in [35]. In general, one major technical difficulty facing interleaved comparisons, as opposed to approaches using click-through rate, is that the number of comparisons required to deal with a K-armed dueling bandits problem is of the order K^2, since every ranker needs to be compared against every other ranker. Some algorithms, such as Beat the Mean, get around this hurdle by averaging the comparisons between each arm and the rest. However, in order for these algorithms to find the Condorcet winner, the suboptimal arms need to satisfy a total ordering condition that is difficult to verify and is often violated in practice, as shown in Section 6.1. One potential remedy for dealing with this difficulty is to use correlation information between rankers and make use of continuous armed bandits algorithms [3, 10, 25, 30, 33], although both the algorithm and the theoretical analysis would be considerably more intricate in this setting, since the goal is not simply to find the maximum of a continuous function.

As explained in Section 5, we based our experiments on the *perfect* click model; in future experimental work we will confirm our experimental findings with alternative click models [15]. We expect such experiments to produce qualitatively similar results, since changing the click model simply makes the interleaved comparisons noisier without affecting whether a ranker beats another on average, i.e., the probability of one ranker beating the other is shifted closer to 0.5 without crossing it. This in turn will make the experiments take longer before finding the best ranker without affecting to which ranker the algorithms converge.

Finally, we remarked in Section 6.2 that RCS manages to beat Condorcet SAVAGE in the performance metrics that the latter was designed for. Thus, another possibility for future work is to investigate whether it is possible to further improve the performance of RCS (and RUCB, for that matter) by taking the horizon into account.

Acknowledgments. This research was supported by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nr 288024 (LiMoSINe project), the Netherlands Organisation for Scientific Research (NWO) under project nrs 640.004.802, 727.011.005, 612.001.116, HOR-11-10, the Center for Creation, Content and Technology (CCCT), the QuaMerdes project funded by the CLARIN-nl program, the TRO-Ve project funded by the CLARIAH program, the Dutch national program COMMIT, the ESF Research Network Program ELIAS, the Elite Network Shifts project funded by the Royal Dutch Academy of Sciences (KNAW), the Netherlands eScience Center under project number 027.012.105 and the Yahoo! Faculty Research and Engagement Program.

REFERENCES

[1] S. Agrawal and N. Goyal. Analysis of Thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory*, pages 1–26, 2012.

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[3] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvari. X-armed bandits. *Journal of Machine Learning Research*, 12:1655–1695, 2011.

[4] O. Chapelle and Y. Chang. Yahoo! learning to rank challenge overview. *Journal of Machine Learning Research-Proceedings Track*, 14:1–24, 2011.

[5] O. Chapelle and L. Li. An empirical evaluation of Thompson sampling. In *NIPS*, 2011.

[6] O. Chapelle, T. Joachims, F. Radlinski, and Y. Yue. Large-scale validation and analysis of interleaved search evaluation. *ACM Trans. Inf. Syst.*, 30(1): 6:1–6:41, 2012.

[7] A. Chuklin, A. Schuth, K. Hofmann, P. Serdyukov, and M. de Rijke. Evaluating aggregated search using interleaving. In *CIKM 2013*. ACM, October 2013.

[8] C. W. Cleverdon, J. Mills, and M. Keen. Factors determining the performance of indexing systems. In *ASLIB Cranfield project*. 1966.

[9] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM '08*, pages 87–94, 2008.

[10] N. de Freitas, A. Smola, and M. Zoghi. Exponential regret bounds for Gaussian process bandits with deterministic observations. In *ICML*, 2012.

[11] F. Guo, L. Li, and C. Faloutsos. Tailoring click models to user goals. In *WSCD '09*, pages 88–92, 2009.

[12] F. Guo, C. Liu, and Y. M. Wang. Efficient multiple-click models in web search. In *WSDM '09*, pages 124–131, New York, NY, USA, 2009. ACM.

[13] J. He, C. Zhai, and X. Li. Evaluation of methods for relative comparison of retrieval systems based on clickthroughs. In *CIKM '09*, pages 2029–2032, 2009.

[14] K. Hofmann, S. Whiteson, and M. de Rijke. A probabilistic method for inferring preferences from clicks. In *CIKM '11*, pages 249–258, USA, 2011.

[15] K. Hofmann, S. Whiteson, and M. de Rijke. Fidelity, soundness, and efficiency of interleaved comparison methods. *ACM Trans. Inf. Syst.*, 31(4), 2013.

[16] K. Hofmann, S. Whiteson, and M. de Rijke. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval*, 16(1):63–90, 2013.

[17] S. Ji, K. Zhou, C. Liao, Z. Zheng, G.-R. Xue, O. Chapelle, G. Sun, and H. Zha. Global ranking by exploiting user clicks. In *SIGIR '09*, pages 35–42, 2009.

[18] T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02*, pages 133–142, 2002.

[19] T. Joachims. Evaluating retrieval performance using clickthrough data. In J. Franke, G. Nakhaeizadeh, and I. Renz, editors, *Text Mining*, pages 79–96. 2003.

[20] S. Jung, J. L. Herlocker, and J. Webster. Click data as implicit relevance feedback in web search. *Information Processing & Management*, 43(3):791–807, 2007.

[21] J. Kamps, M. Koolen, and A. Trotman. Comparative analysis of clicks and judgments for IR evaluation. In *WSCD'09*, pages 80–87, 2009.

[22] E. Kauffmann, N. Korda, and R. Munos. Thompson sampling: an asymptotically optimal finite time analysis. In *ALT'12*, pages 199–213, 2012.

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

[24] Microsoft Learning to Rank Datasets, 2012. `http://research.microsoft.com/en-us/projects/mslr/default.aspx`.

[25] R. Munos. Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *NIPS*, 2011.

[26] F. Radlinski and N. Craswell. Comparing the sensitivity of information retrieval metrics. In *SIGIR '10*, pages 667–674, 2010.

[27] F. Radlinski and N. Craswell. Optimized interleaving for online retrieval evaluation. In *WSDM '13*, 2013.

[28] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *CIKM '08*, pages 43–52, 2008.

[29] F. Scholer, M. Shokouhi, B. Billerbeck, and A. Turpin. Using clicks as implicit judgments: expectations versus observations. In *ECIR'08*, pages 28–39, 2008.

[30] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *ICML*, 2010.

[31] W. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pages 285–294, 1933.

[32] T. Urvoy, F. Clerot, R. Féraud, and S. Naamane. Generic exploration and k-armed voting bandits. In *ICML*, 2013.

[33] M. Valko, A. Carpentier, and R. Munos. Stochastic simultaneous optimistic optimization. In *ICML*, 2013.

[34] E. M. Voorhees and D. K. Harman. *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press, 2005.

[35] Y. Yue and T. Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In *ICML*, 2009.

[36] Y. Yue and T. Joachims. Beat the mean bandit. In *ICML*, 2011.

[37] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The K-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, Sept. 2009.

[38] M. Zoghi, S. Whiteson, R. Munos, and M. de Rijke. Relative upper confidence bound for the k-armed dueling bandits problem. Techn. Report arXiv:1312.3393, 2013.