Balancing Exploration and Exploitation in Learning to Rank Online

Katja Hofmann, Shimon Whiteson and Maarten de Rijke
{K.Hofmann,S.A.Whiteson,deRijke}@uva.nl

ISLA, University of Amsterdam

Abstract. As retrieval systems become more complex, *learning to rank* approaches are being developed to automatically tune their parameters. Using *online* learning to rank approaches, retrieval systems can learn directly from implicit feedback, while they are running. In such an online setting, algorithms need to both explore new solutions to obtain feedback for effective learning, and exploit what has already been learned to produce results that are acceptable to users. We formulate this challenge as an *exploration-exploitation dilemma* and present the first online learning to rank algorithm that works with implicit feedback and balances exploration and exploitation. We leverage existing learning to rank data sets and recently developed click models to evaluate the proposed algorithm. Our results show that finding a balance between exploration and exploitation can substantially improve online retrieval performance, bringing us one step closer to making online learning to rank work in practice.

1 Introduction

Information retrieval (IR) systems are becoming increasingly complex. For example, web search engines combine hundreds of ranking features that each capture a particular aspect of a query, candidate documents, and the match between the two.In heavily used search engines these combinations are carefully tuned to fit users' needs.

For automatically tuning the parameters of such a system, machine learning algorithms are invaluable [13]. Most methods employ supervised learning, i.e., algorithms are trained on examples of relevant and non-relevant documents for particular queries.

While for some applications, such as web search, large amounts of data are available for training, for many environments such data is not available. For example, when deploying a search engine for a local library or company intranet, collecting large amounts of training data required for supervised learning may not be feasible [19]. Even in environments where training data is available, it may not capture typical information needs and user preferences perfectly [15], and cannot anticipate future changes in user needs.

A promising direction for addressing this problem are online approaches for learning to rank [9, 25, 26]. These work in settings where no training data is available before deployment. They learn directly from implicit feedback inferred from user interactions, such as clicks, making it possible to adapt to users throughout the lifetime of the system.

However, collecting a broad enough set of implicit feedback to enable effective online learning is difficult in practice. An online algorithm can observe feedback only on the document lists it presents to the user. This feedback is strongly biased towards the top results, because users are unlikely to examine lower ranked documents [20]. Therefore, effective learning is possible only if the system experiments with new rankings.

Recent online learning to rank approaches address this problem through exploration, for example by interleaving a document list produced by the current best solution with that of a (randomly selected) exploratory solution [25, 26]. However, this purely exploratory behavior may harm the quality of the result list presented to the user. For example, once the system has found a reasonably good solution, most exploratory document lists will be worse than the current solution.

In this paper we frame this fundamental problem as an *exploration–exploitation dilemma*. If the system presents only document lists that it expects will satisfy the user, it cannot obtain feedback on other, potentially better, solutions. However, if it presents document lists from which it can gain a lot of new information, it risks presenting bad results to the user during learning. Therefore, to perform optimally, the system must *explore* new solutions, while also maintaining satisfactory performance by *exploiting* existing solutions. To make online learning to rank for IR work in a realistic setting, we need to find ways to balance exploration and exploitation.

We present the first algorithm that balances exploration and exploitation in a setting where only implicit feedback is available. Our approach augments a recently developed purely exploratory algorithm that learns from implicit feedback [25] with a mechanism for controlling the rate of exploration. We assess the resulting algorithm using a novel evaluation framework that leverages standard learning to rank datasets and models of users' click behavior. Our experiments are the first to confirm that finding a proper balance between exploration and exploitation can improve online performance. We also find that surprisingly little exploration is needed for effective learning. These results bring us one step closer to making online learning to rank work in practice.

2 Related Work

While our method is the first to balance exploration and exploitation in a setting where only implicit feedback is available, a large body of research addresses related problems.

Most work in learning to rank has focused on supervised learning approaches that learn from labeled training examples [13]. A limitation of these approaches is that they cannot use the copious data that can be easily collected while users interact with the search engine. Such implicit feedback directly captures information from the actual users of the system [15]. In particular, preferences between documents [10] and between document lists [18] can be inferred and have been shown to contain sufficient information for learning effective ranking functions [9].

To make learning from implicit feedback effective, online approaches need to explore. Methods based on active learning systematically select document pairs so as to maximize the expected information gain [16, 24]. Two recently developed stochastic methods use interleaved document lists to infer relative preferences between an exploratory and an exploitative ranking function [25, 26]. One algorithm compares a fixed set of ranking functions and selects the best one [26]. The other algorithm, on which our approach is based, uses relative feedback about two ranking functions for stochastic gradient descent [25].

While recent online learning to rank methods provide ways to explore, they do not address how to balance exploration and exploitation. Related research on ad placement and result list diversification has investigated how to balance these factors, but assumes explicit feedback and does not generalize over queries and documents [12, 17].

3 Method

In this section, we formalize the problem of online learning to rank for IR, describe a baseline learning algorithm, and extend it to balance exploration and exploitation.

Problem formulation Our formulation of learning to rank for IR differs from most other work in learning to rank in that we consider a continuous cycle of interactions between users and the search engine. A natural fit for this problem are formalizations from reinforcement learning (RL), a branch of machine learning in which an algorithm learns by trying out actions (e.g., document lists) that generate rewards (e.g., an evaluation measure such as AP or NDCG) from its environment (e.g., users) [21]. Using this formalization allows us to describe this problem in a principled way and to apply concepts and solutions from this well-studied area.

Figure 1 shows the interaction cycle. A user submits a query to a retrieval system, which generates a document list and presents it to the user. The user interacts with the list, e.g., by clicking on links, from which the retrieval system infers feedback about the quality of the presented document list. This problem formulation directly translates to an RL problem (cf., Figure 1, terminology in italics) in which the retrieval system tries, based only on implicit feedback, to maximize a hidden reward signal that corresponds to some evaluation measure. We make the simplifying assumption that queries are independent, i.e., queries are submitted by different users and there are no sessions. This renders the problem a *contextual bandit problem*, a well-studied type of RL problem [1, 11].

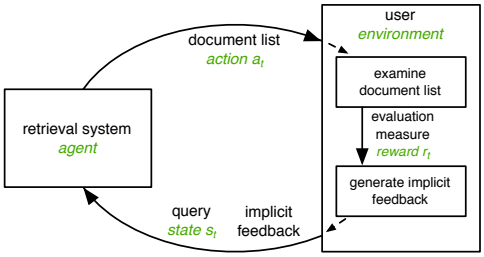

Fig. 1. The IR problem modeled as a contextual bandit problem, with IR terminology in black and corresponding RL terminology in green and italics.

Since our hypothesis is that balancing exploration and exploitation improves retrieval performance *while learning*, we need to measure this aspect of performance. Previous work in learning to rank for IR has considered only final performance, i.e., performance on unseen data after training is completed [13], and, in the case of active learning, learning speed in terms of the number of required training samples [24].

As is common in RL, we measure *cumulative reward*, i.e., the sum of rewards over all queries addressed during learning [21]. Many definitions of cumulative reward are possible, depending on the modeling assumptions. We assume an *infinite horizon problem*, a model that is appropriate for IR learning to rank problems that run indefinitely. Such problems include a *discount factor* $\gamma \in [0, 1)$ that weights immediate rewards higher than future rewards. One way to interpret the discount factor is to suppose that there is a $1 - \gamma$ probability that the task will terminate at each timestep (e.g., users may abandon the retrieval system). Rewards are thus weighted according to the probability that the task will last long enough for them to occur. Then, cumulative reward is defined as the discounted infinite sum of rewards r_i: $C = \sum_{i=1}^{\infty} \gamma^{i-1} r_i$.

Baseline learning approach Our approach builds off a gradient-based policy search algorithm called Dueling Bandit Gradient Descent (DBGD) [25]. This algorithm is particularly suitable for online learning to rank for IR because it generalizes over queries,

requires only relative evaluations of the quality of two document lists, and infers such comparisons from implicit feedback [18].

This approach learns a ranking function consisting of a weight vector w for a linear weighted combinations of feature vectors. Thus, to rank a set of documents D given a query q, feature vectors $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_D\}$ that describe the relation between D and q are produced. Next, scores S for each document are produced using $S = wX$. Finally, documents are ranked by their scores to generate a ranked document list l_w.

Algorithm 1 summarizes this approach. It takes as input a comparison method $f(l_1, l_2)$, that compares two document lists, and three parameters, the step sizes α[1] and δ, and an initial weight vector w_0. At each timestep t, the algorithm observes a query q_t from which two document lists are produced: one exploitative, one exploratory. The exploitative list is produced from the current exploitative weight vector w_t, found to perform best up to the current timestep t. The exploratory list is produced from an exploratory weight vector w'_t, which is generated by moving w_t in a random direction u_t by a step of size δ. The exploitative and exploratory lists are then compared using a function $f(l_1, l_2)$. If the exploratory weight vector w'_t is judged to have produced the better document ranking, the current exploitative weight vector w_t is updated by moving it towards w'_t by a step size α.

Algorithm 1 Baseline algorithm, based on [25].

Input: $f(l_1, l_2)$, α, δ, w_0
for query q_t ($t = 1..T$) **do**
  Sample unit vector u_t uniformly.
  $w'_t \leftarrow w_t + \delta u_t$ *// generate exploratory w*
  if $f(l(w_t), l(w'_t))$ **then**
    $w_{t+1} \leftarrow w_t + \alpha u_t$ *// update exploitative w*
  else
    $w_{t+1} \leftarrow w_t$
return w_{t+1}

For the comparison method $f(l_1, l_2)$, several implementations have been suggested [8, 18]. We chose a variant of the *balanced interleave method* as it is efficient, easy to implement, and was found to be more reliable than the similar *team-draft method* both in [8] and in our own preliminary experiments. This method takes as input two document lists and constructs an interleaved result list by randomly selecting a starting list and then interleaving the two lists so that presentation bias between the two lists is minimized. After observing user clicks on the result list, a preference between the lists is inferred as follows. The rank of the lowest clicked document N is identified. Then, for each list the number of clicked documents within the top N is counted. The list that received more clicks in its top N is preferred. Ties are ignored.

Balancing exploration and exploitation Given an appropriate function for comparing document lists, the baseline algorithm described above learns effectively from implicit feedback. However, the algorithm always explores, i.e., it constructs the result list in a way that minimizes bias between the exploratory and exploitative document lists, which is assumed to produce the best feedback for learning. We now present a comparison function $f(l_1, l_2)$ that does allow balancing exploration and exploitation.

In contrast to previous work, we alter the balanced interleave function to interleave documents probabilistically. Instead of randomizing only the starting position and then interleaving documents deterministically, we randomly select the list to contribute the document at each rank of the result list. In expectation, each list contributes documents to each rank equally often.

[1] In [25], γ denotes the exploitation step size. We use α to avoid confusion with the discount factor γ.

We employ a method for balancing exploration and exploitation that is inspired by ϵ-*greedy*, a commonly used exploration strategies in RL [22]. In ϵ-greedy exploration, the agent selects an action with probability ϵ at each timestep. With probability $1-\epsilon$, it selects the greedy action, i.e., the action with the highest currently estimated value.

Our probabilistic interleave algorithm, which supplies the comparison method required by DBGD, is shown in Algorithm 2. The algorithm takes as input two document lists l_1 and l_2, and an exploration rate k. For each rank of the result list to be filled, the algorithm randomly picks one of the two result lists (biased by the exploration rate k). From the selected list, the highest-ranked document that is not yet in the combined result list is added at this rank. The result list is displayed to the user and clicks C are observed. Then, for each clicked document, a click is attributed to that list if the document is in the top N of the list, where N is the lowest-ranked click.

Algorithm 2 $f(l_1, l_2) - k$-greedy comparison of document lists

Input: l_1, l_2, k
initialize empty result list I
// construct result list
for rank r in (1..10) **do**
  $L \leftarrow l_1$ with probability k, l_2 with probability $1-k$
  $I[r] \leftarrow$ first element of $L \notin I$
display I and observe clicked elements C
$N = length(C)$; $c_1 = c_2 = 0$
for i in $(1..N)$ **do**
  if $C[i] \in l_1[1:N]$ **then**
    $c_1 = c_1 + 1$ *// count clicks on* l_1
  if $C[i] \in l_2[1:N]$ **then**
    $c_2 = c_2 + 1$ *// count clicks on* l_2
$n_1 = |l_1[1:N] \cap I[1:N]|$ *// compensate for bias*
$n_2 = |l_2[1:N] \cap I[1:N]|$
$c_2 = \frac{n_1}{n_2} * c_1$
return $c_1 < c_2$

The exploration rate $k \in [0.0, 0.5]$ controls the relative amount of exploration and exploitation, similar to ϵ. It determines the probability with which a list is selected to contribute a document to the interleaved result list at each rank. When $k = 0.5$, an equal number of documents are presented to the user in expectation. As k decreases, more documents are contributed by the exploitative list, which is expected to improve the quality of the result list but produce noisier feedback.

As k decreases, more documents from the exploitative list are presented, which introduces bias for inferring feedback. The bias linearly increases the expected number of clicks on the exploitative list and reduces the expected number of clicks on the exploratory list. We can partially compensate for this bias since $E[c_2] = \frac{n_1}{n_2} * E[c_1]$, where $E[c_i]$ is the expected number of clicks within the top N of list l_i, and n_i is the number of documents from l_i that were displayed in the top N of the interleaved result list. This compensates for the expected number of clicks, but leaves some bias in the expected number of times each document list is preferred. While perfectly compensating for bias is possible, it would require making probabilistic updates based on the observed result. This would introduce additional noise, creating a bias/variance trade-off. Preliminary experiments show that the learning algorithm is less susceptible to increased bias than to increased noise. Therefore we use this relatively simple, less noisy bias correction.

4 Experiments

Evaluating the ability of an algorithm to maximize cumulative performance in an online IR setting poses unique experimental challenges. The most realistic experimental setup – in a live setting with actual users – is risky because users may get frustrated with bad

search results. The typical TREC-like setup used in supervised learning to rank for IR is not sufficient because information on user behavior is missing.

To address these challenges, we propose an evaluation setup that simulates user interactions. This setup combines datasets with explicit relevance judgments that are typically used for supervised learning to rank with recently developed click models. Given a dataset with queries and explicit relevance judgments, interactions between the retrieval system and the user are simulated (c.f., the box labeled "user/environment" in Figure 1). Submitting a query is simulated by random sampling from the set of queries. After the system has generated a result list for the query, feedback is generated using a click model and the relevance judgments provided with the dataset. Note that the explicit judgments from the dataset are not directly shown to the retrieval system but rather used to simulate the user feedback and measure cumulative performance.

Click model Our click model is based on the Dependent Click Model (DCM) [6, 7], a generalization of the cascade model [3]. The model posits that users traverse result lists from top to bottom, examining each document as it is encountered. Based on this examination, the user decides whether to click on the document or skip it. After each clicked document the user decides whether or not to continue examining the document list. Since the DCM has been shown to effectively predict users' click behavior [7], we believe it is a good model for generating implicit feedback.

When a user examines a document in the result list, they do not know the true relevance label of the document. However, aspects of the document's representation in the result list (e.g., title) make it more likely that a document is clicked if it is relevant. Using this assumption, the ground truth relevance judgments provided in explicitly annotated learning to rank datasets, and the process put forward by the DCM, we define the following model parameters. Relevant documents are clicked with a probability $p(c|R)$, the probability of a click given that a document is relevant. Non-relevant documents can attract (noisy) clicks, with probability $p(c|NR)$. After clicking a document, the user may be satisfied with the results and stop examination with probability $p(s|R)$, the probability of stopping examination after clicking on a relevant document. The probability of stopping after visiting a non-relevant document is denoted by $p(s|NR)$.

To instantiate this click model we need to define click and stop probabilities. When DCM is trained on large click logs, probabilities are estimated for individual query-document pairs, while marginalizing over the position at which documents were presented in the training data. In our setting, learning these probabilities directly is not possible, because no click log data is available. Therefore we instantiate the model heuristically, making choices that allow us to study the behavior of our approach in various settings. Setting these probabilities heuristically is reasonable because learning outcomes for the gradient descent algorithm used in this paper are influenced mainly by how much more likely users are to click on relevant and non-relevant documents. Thus, this ratio is more important than the actual numbers used to instantiate the model.

Table 1 gives an overview of the click models used in our experiments. First, to obtain an upper bound on the performance that could be obtained if feedback was deterministic, we define a *perfect* model, where all relevant documents are clicked and no non-relevant documents are clicked. The two realistic models are based on typ-

Table 1. Overview of the click models used.

model	$p(c\|R)$	$p(c\|NR)$	$p(s\|R)$	$p(s\|NR)$
perfect	1.0	0.0	0.0	0.0
navigational	0.95	0.05	0.9	0.2
informational	0.9	0.4	0.5	0.1

ical user behavior in web search [2, 6], because 8 of the 9 datasets we use implement web search tasks (see below). In a navigational task, users look for a specific document they know to exist in a collection, e.g., a company's homepage. Typically, it is easy to distinguish relevant and non-relevant documents and the probability of stopping examination after a relevant hit is high. Therefore, our *navigational* model is relatively reliable, with a high difference between $p(c|R)$ and $p(c|NR)$. In an informational task, users look for information about a topic, which can be distributed over several pages. Here, users generally know less about what page(s) they are looking for and clicks tend to be noisier. This behavior leads to the *informational* model, which is much noisier than the navigational model.

Data We conducted our experiments using two standard collections for learning to rank: letor 3.0 and letor 4.0 [14]. In total, these two collections comprise 9 datasets. Each consists of queries for which features were extracted from a document collection, together with relevance judgements for the considered query-document pairs.

The datasets were compiled from different sources: the 106 queries in OHSUMED are based on a log of a search engine for scientific abstracts drawn from the MedLine database. The remaining datasets are based on Web Track collections run between 2003 and 2008 at TREC. HP2003, HP2004, NP2003, NP2004, TD2003 and TD2004 implement homepage finding, named-page finding, and topic distillation tasks, using a crawl of web pages within the .gov domain. These datasets contain between 50–150 queries each, with about 1000 judged documents per query. MQ2007 and MQ2008 are based on the 2007 and 2008 Million Query track at TREC and use the ".GOV2" collection. These two datasets contain substantially more queries, 1700 and 800 respectively, but much fewer judged documents per query.

The datasets based on the TREC Web track use binary relevance judgments, while OHSUMED, MQ2007 and MQ2008 are judged on a 3-point scale from 0 (non-relevant) to 2 (highly relevant). In all experiments we use binary relevance judgments. For the three datasets that originally contain graded judgments, we treat all judgments greater than zero as relevant. In preliminary experiments with graded relevance, we obtained results nearly identical to those with the simpler binary judgments.[2]

Each dataset comes split up for machine learning experiments using 5-fold cross-validation. We use the training sets for training during the learning cycle and for calculating cumulative performance, and the test sets for measuring final performance.

Runs In all experiments we initialize the starting weight vector w_0 randomly, and use the best performing parameter settings from [25]: $\delta = 1$ and $\alpha = 0.01$. Our *baseline* is Algorithm 1, based on [25], which corresponds to a purely exploratory setting of $k = 0.5$ in our extended method. Against this baseline we compare *exploit* runs that balance exploration and exploration by varying the exploration rate k between 0.4 and 0.1 as shown in Algorithm 2. All experiments are run for 1000 iterations.

Discounting Because our problem formulation assumes an infinite horizon, cumulative performance is defined as an infinite sum of discounted rewards (cf. §3). Since experiments are necessarily finite, we cannot compute this infinite sum exactly. However, because the sum is discounted, rewards in the far future have little impact and cumulative performance can be approximated with a sufficiently long finite experiment.

[2] The reason appears to be that the learning algorithm works with very coarse feedback, so more finely grained feedback has little influence on the reliability of inferred judgments.

In our experiments, we set the discount factor $\gamma = 0.995$. This choice can be justified in two ways. First, it is typical of discount factors used when evaluating RL methods [21]. Choosing a value close to 1 ensures that future rewards have significant weight and thus the system must explore in order to perform well. Second, at this value of γ, cumulative performance can be accurately estimated with the number of queries in our datasets. Since rewards after 1000 iterations have a weight of 1% or less, our finite runs are good approximations of true cumulative performance.

Evaluation measures We use cumulative NDCG on the result list presented to the user to measure cumulative performance of the system. We define cumulative reward as the discounted sum of NDCG that the retrieval system accrues throughout the length of the experiment. Final performance is reported in terms of NDCG on the test set. Though omited here due to lack of space, we also conducted experiments measuring cumulative and final performance based on MAP and MRR and observed similar results.

For each dataset we repeat all runs 25 times and report results averaged over folds and repetitions. We test for significant differences with the baseline runs ($k = 0.5$, the first column of Table 3) using a two-sided student's t-test. Runs that significantly outperform the exploratory baseline are marked with $^{\triangle}$ ($p < 0.05$) or $^{\blacktriangle}$ ($p < 0.01$).

5 Results and Discussion

The main goal of this paper is to show that balancing exploration and exploitation in online learning to rank for IR can improve cumulative performance. However, such a result is meaningful only if our baseline learning algorithm learns effectively in this setting. Therefore, before turning to our main results, we assess our baseline algorithm.

Baseline learning approach Figure 2 shows example learning curves for the dataset *NP2003* at different settings of k and for all click models. Learning curves for all other datasets are qualitatively similar, and we omit those due to space restrictions. The figure shows final performance in terms of NDCG on the test sets after each learning step. We see that final performance improves over time in all settings. As expected, learning is faster when feedback is more reliable. For the idealized *perfect* click model, final performance after 1000 iterations ranges between 0.777 and 0.785 for different settings of k. For the noisy *informational* click model at the same settings, final performance is between 0.412 and 0.546. Although final performance drops substantially as implicit feedback becomes extremely noisy, we find that as long as there is a signal, i.e., relevant documents are more likely to be clicked than non-relevant ones, performance improves over time for all datasets.

We find an interaction effect between click model and exploration rate. When the click model is reliable, there is no significant difference between the final performance at different settings of k. However, in the *informational* click model, variance increases, and there is a large difference between final performance at different settings of k. This is a direct and expected consequence of the noise in inferred feedback. More surprising is that final performance improves for smaller k, since we expected feedback to be the most reliable for the fully exploratory setting $k = 0.5$. Instead, it appears that, since bias is only partially compensated for (cf., §3), the remaining bias at lower values of k smoothes over some of the noise in the click model. At lower exploration rates, fewer

results from the exploratory list are presented and it becomes harder for the exploratory list to win the comparison. Thus, instead of noisier updates, the algorithm makes fewer, more reliable updates that on average result in greater performance gains.

As a final sanity check, we calculate standard evaluation measures that are typically used to evaluate supervised learning to rank methods. Results for the *perfect* click model and $k = 0.5$ after 1000 iterations are listed in Table 2. Despite the limited information available to the algorithm (relative quality of the result list instead of explicit relevance judgment per document), performance is competitive with current supervised learning to rank algorithms [13]. Note that we did not tune parameters of the algorithm for final performance, so further improvements may be possible.

Balancing exploration and exploitation We now turn to our main research question: *Can balancing exploration and exploitation improve online performance?* Our results are shown in Table 3. Cumulative performance for all *exploit* runs ($k \in [0.1, 0.4]$) is compared to the purely exploratory baseline ($k = 0.5$). Best runs per row are highlighted in bold and significant differences are marked as described above.

Our focus is on comparing relative performance per dataset. Starting with the *perfect* click model, we see that for all datasets the baseline is outperformed by all lower settings of k. For $k < 0.4$ all improvements over the baseline are statistically significant. The improvements range from 4.1% (*OHSUMED*) to 12.35% (*NP2004*).

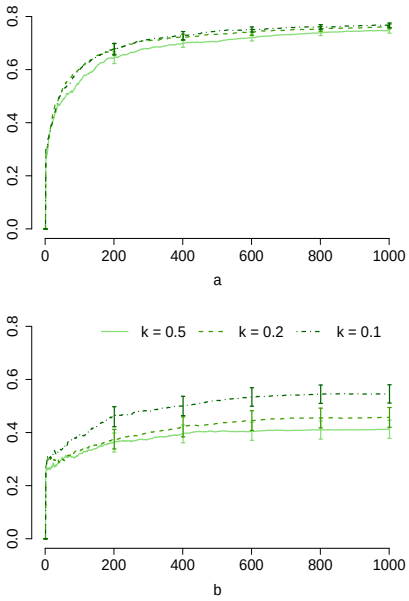

Fig. 2. Final performance (with 5% confidence intervals) over time for the dataset *NP2003* for a) *navigational*, and b) *informational* click models and $k \in \{0.1, 0.2, 0.5\}$

Table 2. NDCG@10, Precision@10, and MAP for the baseline algorithm.

	NDCG@10	*P@10*	*MAP*
HP2003	0.792	0.102	0.721
HP2004	0.770	0.096	0.676
NP2003	0.761	0.090	0.649
NP2004	0.787	0.093	0.659
TD2003	0.296	0.152	0.231
TD2004	0.298	0.236	0.206
OHSUMED	0.422	0.488	0.437
MQ2007	0.375	0.335	0.410
MQ2008	0.488	0.238	0.447

We observe similar results for the *navigational* click model. For all datasets, there are several lower settings of k where performance improves over the baseline. For all but one dataset (*MQ2007*), these improvements are statistically significant. Improvements range from 0.54% (*MQ2007*) to 21.9% (*NP2003*).

The trend continues for the *informational* click model. Again, the purely exploratory baseline is outperformed by more exploitative settings in all cases. For 7 out of 9 cases the improvements are statistically significant. The improvement ranges up to 35.9% for the dataset *HP2004*.

We find that for all click models and all datasets balancing exploration and exploitation can significantly improve online performance over the purely exploratory baseline.

Comparing cumulative performance listed in Table 3 with final performance in Table 2, we find that cumulative performance does not depend only on final performance. For example, NDCG@10 and MAP for *HP2003* are much higher than for *OHSUMED*, but cumulative performance is very similar (precision scores are low for *HP2003* because there are few relevant documents in general, and are not a good indicator of the relative quality of result rankings). We find that the main factors affecting cumulative performance are the speed of learning and how effectively early learning gains are exploited. This confirms that measuring final performance is not enough when evaluating online learning to rank algorithms.

The best setting for exploration rate k is 0.1 or 0.2 in all but two cases. A setting of $k = 0.2$ means that by injecting, on average, only two documents from an exploratory list, the algorithm learns effectively and achieves good cumulative performance while learning. This means that surprisingly little exploration is sufficient for good performance and that the original algorithm (our baseline) explores too much.

Table 3. Results. Cumulative NDCG for *baseline* ($k = 0.5$) and *exploit* ($k \in [0.1, 0.4]$) runs.

k	*0.5*	*0.4*	*0.3*	*0.2.*	*0.1*
click model: perfect					
HP2003	119.91	125.71▲	129.99▲	**130.55**▲	128.50▲
HP2004	109.21	111.57	118.54▲	**119.86**▲	116.46▲
NP2003	108.74	113.61▲	117.44▲	**120.46**▲	119.06▲
NP2004	112.33	119.34▲	124.47▲	**126.20**▲	123.70▲
TD2003	82.00	84.24	88.20▲	**89.36**▲	86.20▲
TD2004	85.67	90.23▲	91.00▲	**91.71**▲	88.98△
OHSUMED	128.12	130.40▲	131.16▲	**133.37**▲	131.93▲
MQ2007	96.02	97.48	98.54▲	**100.28**▲	98.32▲
MQ2008	90.97	92.99▲	94.03▲	**95.59**▲	95.14▲
click model: navigational					
HP2003	102.58	109.78▲	**118.84**▲	116.38▲	117.52▲
HP2004	89.61	97.08▲	99.03▲	103.36▲	**105.69**▲
NP2003	90.32	100.94▲	105.03▲	108.15▲	**110.12**▲
NP2004	99.14	104.34△	110.16▲	112.05▲	**116.00**▲
TD2003	70.93	75.20▲	**77.64**▲	77.54▲	75.70△
TD2004	78.83	80.17	82.40△	**83.54**▲	80.98
OHSUMED	125.35	126.92△	127.37▲	**127.94**▲	127.21
MQ2007	95.50	94.99	95.70	**96.02**	94.94
MQ2008	89.39	90.55	91.24△	**92.36**▲	92.25▲
click model: informational					
HP2003	59.53	63.91	61.43	70.11△	**71.19**▲
HP2004	41.12	52.88▲	48.54△	**55.88**▲	55.16▲
NP2003	53.63	53.64	57.60	58.40	**69.90**▲
NP2004	60.59	63.38	64.17	63.23	**69.96**△
TD2003	52.78	52.95	51.58	55.76	**57.30**
TD2004	58.49	61.43	59.75	62.88△	**63.37**
OHSUMED	121.39	123.26	124.01△	**126.76**▲	125.40▲
MQ2007	91.57	**92.00**	91.66	90.79	90.19
MQ2008	86.06	87.26	85.83	**87.62**	86.29

While balancing exploration and exploitation improves performance for all datasets, the magnitude of these improvements differs substantially. For example, for the *navigational* click model, the relative improvement between baseline and best setting for *NP2003* is 21.9%, while for *MQ2007* the difference is only 0.54%. This reflects a general difference between datasets obtained from the 2003 and 2004 TREC web tracks and the remaining datasets. The first contain 1000 candidate documents per query, but few relevant documents. Therefore, it is relatively difficult to find a good ranking and minor changes in weights can result in substantially worse result lists. Consequently, the differences between exploratory and exploitative document lists are large, leading to large improvements when more documents from the exploitative list are selected. In contrast, the datasets *MQ2007*, *MQ2008*, and *OHSUMED* contain fewer candidate documents but many more relevant ones. Therefore, exploring does not hurt as much as in other settings and differences between exploratory and exploitative settings tend to be smaller. Note that in realistic settings it is likely that more candidate documents are considered, so the effect of exploiting more is likely to be stronger.

The different instantiations of the click model also result in qualitative differences in cumulative performance. Performance is higher with *perfect* feedback, and decreases

as feedback becomes noisier. Performance on some datasets is more strongly affected by noisy feedback. For the *HP*, *NP*, and *TD* datasets, performance for the *informational* model drops substantially. This may again be related to the large number of non-relevant documents in these datasets. As finding a good ranking is harder, noise has a stronger effect. Despite this drop in performance, balancing exploration and exploitation consistently leads to better cumulative performance than the purely exploratory baseline.

6 Conclusion

In this paper, we formulated online learning to rank for IR as an RL problem, casting the task as a contextual bandit problem in which only implicit feedback is available. We argued that, in such problems, learning to rank algorithms must balance exploration and exploitation in order to maximize cumulative performance. We proposed the first algorithm to do so in an IR setting with only implicit feedback. This algorithm extends a stochastic gradient descent algorithm with a mechanism for controlling the rate of exploration. Since assessing the performance of such algorithms poses unique challenges, we introduced a evaluation framework based on simulated interaction that can measure the cumulative performance of online methods.

The performance of our method was compared to a purely exploratory baseline, using three click models and nine datasets. We demonstrate that a proper balance between exploration and exploitation can significantly and substantially improve cumulative performance, which confirms our hypothesis. Surprisingly little exploration is needed for good performance, and we analyzed how the reliability of user feedback and differences between datasets affect the balance between exploration and exploitation.

Given this initial evidence of the benefit of balancing exploration and exploitation in online IR, developing new algorithms for this problem is an important goal for future research. In addition to the approach developed in this paper, approaches combining active learning [4, 23] with exploration strategies from RL could be developed.

Our evaluation framework is based on existing learning to rank data collections and a probabilistic model of how users examine result lists and decide whether to follow a link to a document. An interesting future direction is to leverage click log data for evaluation. The main challenge is to account for the fact that not all possible rankings are contained in logs. Possible solutions could be based on click models estimated from such data, like the one underlying the click model used in this paper [5, 7].

Like all simulations, our experiments are based on specific modeling assumptions. In the future, experiments in real-life settings will be indispensable for verifying these assumptions. Interesting questions include how a discount factor should be set for the requirements of specific search environments. In preliminary experiments, we found that our results do not depend on the infinite horizon model, as we obtained qualitatively similar results in a finite horizon setting. Also, while our click model is based on specific validated models, particular instantiations of this model had to be chosen heuristically. Therefore, it would be useful to investigate what instantiations of the click model best reflect the behavior of real users in such environments.

Acknowledgements. This research was partially supported by the European Union's ICT Policy Support Programme as part of the Competitiveness and Innovation Framework Programme, CIP ICT-PSP under grant agreement nr 250430, by the PROMISE Network of Excellence co-funded by the 7th Framework Programme of the European

Commission, grant agreement no. 258191, by the DuOMAn project carried out within the STEVIN programme which is funded by the Dutch and Flemish Governments under project nr STE-09-12, by the Netherlands Organisation for Scientific Research (NWO) under project nrs 612.061.814, 612.061.815, 640.004.802, 380-70-011, and by the Center for Creation, Content and Technology (CCCT).

7 References

[1] A. G. Barto, R. S. Sutton, and P. S. Brouwer. Associative search network: A reinforcement learning associative memory. *IEEE Trans. Syst., Man, and Cybern.*, 40:201–211, 1981.

[2] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.

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

[4] P. Donmez and J. Carbonell. Active sampling for rank learning via optimizing the area under the ROC curve. In *ECIR'09*, pages 78–89, 2009.

[5] G. E. Dupret and B. Piwowarski. A user browsing model to predict search engine click data from past observations. In *SIGIR '08*, pages 331–338, 2008.

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

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

[8] 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.

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

[10] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR '05*, pages 154–161. ACM Press, 2005.

[11] J. Langford and T. Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *NIPS'08*, pages 817–824, 2008.

[12] J. Langford, A. Strehl, and J. Wortman. Exploration scavenging. In *ICML '08*, pages 528–535, 2008.

[13] T. Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[14] T.-Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *LR4IR '07*, 2007.

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

[16] F. Radlinski and T. Joachims. Active exploration for learning rankings from clickthrough data. In *KDD '07*, pages 570–579, 2007.

[17] F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *ICML '08*, pages 784–791. ACM, 2008.

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

[19] M. Sanderson. Test collection based evaluation of information retrieval systems. *Foundations and Trends in Information Retrieval*, 4(4):247–375, 2010.

[20] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

[21] R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1998.

[22] C. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, 1989.

[23] Z. Xu, R. Akella, and Y. Zhang. Incorporating diversity and density in active learning for relevance feedback. In *ECIR'07*, pages 246–257, 2007.

[24] Z. Xu, K. Kersting, and T. Joachims. Fast active exploration for link-based preference learning using gaussian processes. In *ECML PKDD'10*, pages 499–514, 2010.

[25] Y. Yue and T. Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In *ICML'09*, pages 1201–1208, 2009.

[26] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The k-armed dueling bandits problem. In *COLT'09*, 2009.