Inf Retrieval (2013) 16:63–90
DOI 10.1007/s10791-012-9197-9

Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval

Katja Hofmann · Shimon Whiteson · Maarten de Rijke

Received: 19 September 2011 / Accepted: 7 March 2012 / Published online: 7 April 2012

Abstract As retrieval systems become more complex, *learning to rank* approaches are being developed to automatically tune their parameters. Using *online* learning to rank, retrieval systems can learn directly from implicit feedback inferred from user interactions. In such an online setting, algorithms must obtain feedback for effective learning while simultaneously utilizing what has already been learned to produce high quality results. We formulate this challenge as an *exploration–exploitation dilemma* and propose two methods for addressing it. By adding mechanisms for balancing exploration and exploitation during learning, each method extends a state-of-the-art learning to rank method, one based on *listwise* learning and the other on *pairwise* learning. Using a recently developed simulation framework that allows assessment of online performance, we empirically evaluate both methods. Our results show that balancing exploration and exploitation can substantially and significantly improve the online retrieval performance of both listwise and pairwise approaches. In addition, the results demonstrate that such a balance affects the two approaches in different ways, especially when user feedback is noisy, yielding new insights relevant to making online learning to rank effective in practice.

Keywords Information retrieval · Learning to rank · Implicit feedback

An earlier version of this article appeared in Hofmann et al. (2011a). In this substantially revised and extended version we introduce a novel approach for balancing exploration and exploitation that works with pairwise online learning approaches, and carefully evaluate this new approach. Comparisons with the earlier described algorithm for listwise approaches yield new insights into the behavior of the two types of approach in online settings, especially how they compare in the face of noisy feedback and how they react to a balance of exploration and exploitation.

K. Hofmann (✉) · S. Whiteson · M. de Rijke
ISLA, University of Amsterdam, Amsterdam, The Netherlands
e-mail: k.hofmann@uva.nl

S. Whiteson
e-mail: s.a.whiteson@uva.nl

M. de Rijke
e-mail: derijke@uva.nl

1 Introduction

Information retrieval (IR) systems are becoming increasingly complex. For example, web search engines may combine hundreds of ranking features that each capture a particular aspect of a query, candidate documents, and the match between the two.[1] In heavily used search engines, these combinations are carefully tuned to fit users' needs. However, on smaller-scale systems, such careful manual tuning is often infeasible.

For automatically tuning the parameters of such a system, machine learning algorithms are invaluable (Liu 2009). Most methods employ supervised learning, i.e., algorithms are trained on examples of relevant and non-relevant documents for particular queries. While large amounts of data are available for training in some applications, such as web search, there are also many situations in which such data cannot be obtained. For example, when deploying a search engine for a company's intranet (enterprise search) or a personal computer (desktop search, personalization), collecting the large amounts of training data required for supervised learning is usually not feasible (Sanderson 2010). Even in environments where training data is available, it may not capture typical information needs and user preferences perfectly (Radlinski and Craswell 2010) and cannot anticipate future changes in user needs.

A promising direction for addressing a lack of resources for manual or supervised training are online approaches for learning to rank (Joachims 2002; Yue and Joachims 2009; Yue et al. 2009). These methods 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.[2]

In an online setting, it is crucial to consider the impact of such learning on the users. In contrast to offline approaches, where the goal is to learn as effectively as possible from the available training data, online learning affects, and is affected by, how user feedback is collected. Ideally, the learning algorithm should not interfere with the user experience, observing user behavior and learning in the background, so as to present search results that meet the user's information needs as well as possible at all times. This would imply passively observing, e.g., clicks on result documents. However, passively observed feedback can be biased towards the top results displayed to the user (Silverstein et al. 1999). Learning from this biased feedback may be suboptimal, thereby reducing the system's performance later on. Consequently, an online learning to rank approach should take into account both the quality of current search results, and the potential to improve that quality in the future, if feedback suitable for learning can be observed.

In this article, we frame this fundamental trade-off 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*

[1] http://www.google.com/corporate/tech.html

[2] This article focuses on learning solutions that generalize to unseen queries. Thus, learning from previous interactions with results for the *same* query is not possible, in contrast to settings assumed by most on-line relevance feedback and re-ranking approaches. These approaches are orthogonal to work in online learning to rank and could, e.g., be used to further improve learned rankings for frequent queries.

existing solutions. Making online learning to rank for IR work in realistic settings requires effective ways to balance exploration and exploitation.

We investigate mechanisms for achieving a balance between exploration and exploitation when using listwise and pairwise methods, the two most successful approaches for learning to rank in IR (Liu 2009). The pairwise approach takes as input pairs of documents with labels identifying which is preferred and learns a classifier that predicts these labels. In principle, pairwise approaches can be directly applied online, as preference relations can be inferred from clicks (Joachims 2002). However, as we demonstrate in this article, balancing exploration and exploitation is crucial to achieving good performance. In contrast, listwise approaches aim to directly optimize an evaluation measure, such as NDCG, that concerns the entire document list. Since such evaluation measures cannot be computed online, new approaches that work with implicit feedback have been developed (Yue and Joachims 2009). These online approaches rely on interleaving techniques, where preference relations between two ranking functions can be inferred from aggregated clicks (Joachims et al. 2007).

In this article, we present the first two algorithms that can balance exploration and exploitation in settings where only implicit feedback is available. First, we start from a recently developed listwise algorithm that is initially purely exploratory (Yue and Joachims 2009). Second, we develop a similar mechanism for a pairwise approach that is initially purely exploitative.

We assess the resulting algorithms using an evaluation framework that leverages standard learning to rank datasets and models of users' click behavior. Our main result is that finding a proper balance between exploration and exploitation can substantially and significantly improve the online retrieval performance of both listwise and pairwise approaches.

In addition, our results are the first to shed light on the strengths and weaknesses of using pairwise and listwise approaches online, as they have previously only been compared offline. We find that the pairwise approach can learn effectively when feedback is reliable. However, when feedback is noisy, a high amount of exploration is required to obtain reasonable performance. The listwise approach learns more slowly when provided with perfect feedback, but is much more robust to noise than the pairwise approach. We discuss in detail the effects on each approach of balancing exploration and exploitation, the amount of noise in user feedback, and characteristics of the datasets. Finally, we describe the implications of our results for making these approaches work effectively in practice.

The remainder of this paper is organized as follows. We present related work in Sect. 2 and our methods for balancing exploration and exploitation in Sect. 3. Experiments are described in Sect. 4, followed by results and analysis in Sect. 5. We conclude in Sect. 6.

2 Related work

While our methods are the first to balance exploration and exploitation in a setting where only implicit relative feedback is available, a large body of research addresses related problems. The question of how to explore is addressed by active learning approaches for supervised learning to rank, and in online learning to rank for IR approaches. A large body of work in reinforcement learning (RL), a branch of machine learning where an algorithm learns from interactions with an environment (Sutton and Barto 1998), addresses the problem of balancing exploration and exploitation when explicit feedback can be observed.

Below, we first summarize related work in learning to rank for IR and then turn to relevant results from RL.

As an alternative to manual tuning of IR systems, supervised learning to rank has become very popular in recent years (Liu 2009). Various approaches have been developed, the most successful of which are pairwise [e.g., RankSVM (Herbrich et al. 1999; Joachims 2002)] and listwise [e.g., SoftRank (Taylor et al. 2008)] learning to rank. As in other supervised learning settings, supervised learning to rank methods typically assume that a representative set of training data (including judgments) is available at training time, so that characteristics of the data can be estimated from this set.

In contrast to offline supervised learning, we address learning to rank in an online setting, where a system learns directly from interactions with the user. In this setting, labeled training data is not provided but needs to be collected through interaction with the user. A first evaluation of RankSVM in an online setting demonstrated that learning from implicit feedback is possible in principle (Joachims 2002). How to best collect feedback for effective learning from implicit feedback has so far not been examined further, but we hypothesize that online approaches need to explore to learn effectively.

Many researchers have considered how best to explore efficiently in information retrieval, without addressing the question of how to balance that exploratory behavior with exploitation. The most common approach is based on active learning, where the focus is on reducing manual labeling. Xu et al. (2007) present an algorithm that learns a linear combination of features based on relevance, document density, and diversity, which is then used to select documents for which to obtain relevance feedback. Similarly, Xu and Akella (2008) follow a probabilistic approach that selects documents expected to minimize model variance. Donmez and Carbonell (2009) apply active learning to two state-of-the-art learning to rank algorithms, RankSVM and RankBoost. Their approach selects the training instances expected to have the largest effect on the current model.

Two recently developed approaches have addressed the problem of exploration in online learning to rank. These stochastic methods infer feedback using interleaved comparison methods (Hofmann et al. 2011b; Radlinski and Craswell 2010). Using such techniques, the algorithms infer relative preferences between an exploratory and an exploitative ranking function (Yue and Joachims 2009; Yue et al. 2009). One algorithm compares a fixed set of ranking functions and selects the best one (Yue et al. 2009). The other algorithm, on which our approach is based, uses relative feedback about two ranking functions for stochastic gradient descent (Yue and Joachims 2009).

In RL, balancing exploration and exploitation is considered important for optimizing performance while learning, and we hypothesize that similar benefits can be achieved in information retrieval. The distinguishing characteristic of RL problems is that an agent interacts with an environment by trying out actions and receiving rewards (Kaelbling et al. 1996). In this setting, the agent can only observe the rewards for the actions it selected, meaning that it is never shown any examples of the optimal action for any situation, as is the case in e.g., supervised learning. For this reason, balancing exploration and exploration is crucial. The agent needs to try out new solutions to be able to learn from the observed feedback, and it needs to exploit what it has already learned to ensure high reward.[3]

[3] Balancing exploration and exploitation also plays an important role in other areas, such as sequential experimental design and in the multi-armed bandit work coming from the applied probability community. Early work includes (Robbins 1952), with an important breakthrough by Gittins (1979). A recent overview can be found in (Mahajan and Teneketzis 2008). Exploration and exploitation have also been extensively studied as fundamental principles of human and animal decision-making behavior (Cohen et al. 2007).

Particularly relevant to this article are methods for tackling contextual bandit problems,[4] a well-studied type of RL problem (Auer 2003; Barto et al. 1981; Langford and Zhang 2008; Strehl et al. 2006), as they have been successfully applied to problems similar to learning to rank for IR. For example, Langford et al. (2008) consider the ad placement application. Given a website, their algorithm learns the value of placing each of a set of candidate ads on the website. Similarly, Radlinski et al. (2008a) consider how to learn diverse document lists such that different information needs are satisfied; they present an algorithm for doing so that balances exploration and exploitation. Another widely-studied application area of related approaches is news recommendation, where news stories are selected for a user population or for individual users. Work in this area has focused on learning approaches (Agarwal et al. 2008; Li et al. 2010), and methods for offline evaluation (Li et al. 2011). Finally, an application to adaptive filtering is presented in (Zhang et al. 2003). However, like other RL algorithms, these methods all assume access to explicit feedback. For example, in ad placement, clicks provide explicit feedback because they are directly correlated with the value of the ad-website pair (assuming a pay-per-click model). Since explicit feedback is not typically available in online IR settings, these methods are not directly applicable. Note that, while in related areas implicit feedback can often be interpreted as absolute (possibly noisy) reward, this is not possible in our setting.

Within IR, using implicit feedback to improve search results has long been the goal of relevance feedback approaches. Exploration in the relevance feedback setting has, e.g., been formulated in terms of diversifying the list of initial documents that is shown to the user to elicit relevance feedback (Karimzadehgan and Zhai 2010). These methods typically assume explicit feedback (which can be noisy) and do not generalize over queries.

3 Approach

In this section, we first formalize the problem of online learning to rank for IR. Then we describe our approaches, starting with the baseline learning algorithms, and extend them to balance exploration and exploitation.

3.1 Problem formulation

Our formulation of learning to rank for IR differs from most other work in learning to rank in that we consider the interactions between users and the search engine as a continuous cycle. A natural fit for this problem are formalizations from RL, 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) (Sutton and Barto 1998). 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., Fig. 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,

[4] Contextual bandit problems are also known as *bandits with side information*, or *associative RL*.

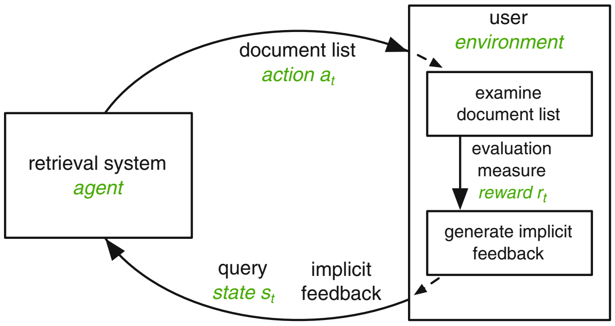

Fig. 1 The IR problem modeled as a contextual bandit problem, with IR terminology in *black* and corresponding RL terminology in *green* and *italics*

i.e., queries are submitted by different users and there are no sessions. This renders the problem a *contextual bandit problem* (Barto et al. 1981; Langford and Zhang 2008).

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 (Liu 2009), and, in the case of active learning, learning speed in terms of the number of required training samples (Xu et al. 2010).

As is common in RL, we measure *cumulative reward*, i.e., the sum of rewards over all queries addressed during learning (Sutton and Barto 1998). 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 $r_i : C = \sum_{i=1}^{\infty} \gamma^{i-1} r_i$.

To summarize, we model online learning to rank for IR as an interaction cycle between the user and the retrieval system. We assume an infinite horizon setting and use discounting to emphasize immediate reward. The resulting problem formulation differs from those traditionally used in IR because performance depends on cumulative reward during the entire learning process, rather than just the quality of the final retrieval system produced by learning. It also differs from typical contextual bandit problems, which assume that the agent has access to the true immediate reward resulting from its actions. Typical IR evaluation measures require explicit feedback, which is not available in most realistic use cases for online learning to rank. Thus, this contextual bandit problem is distinct in that it requires the learner to cope with implicit feedback such as click behavior.

3.2 Balancing exploration and exploitation in pairwise learning to rank

Our first approach builds off a pairwise formulation of learning to rank, and a stochastic gradient descent learner. Pairwise approaches model the pairwise relations between documents for a given query. Our formulation of the learning to rank problem from implicit feedback follows (Joachims 2002). The learning algorithm is a stochastic gradient descent algorithm, following (Sculley 2009; Zhang 2004).

3.2.1 Baseline learning approach

Pairwise learning to rank approaches for IR construct training data from observed queries and preference relations between pairs of documents. Given an appropriate method of obtaining such pairwise labeled training data, learning in this setting can be reduced to binary classification (Sculley 2009). Our baseline algorithm obtains pairwise labels by observing clicks on an exploitative result list and learns a weight vector w for ranking documents using the learning algorithm from (Sculley 2009; Zhang 2004).

Our pairwise approach infers labels from clicks, following the method developed by (Joachims 2002). This method is based on the observation that clicks on results are too unreliable to allow conclusions about absolute relevance of clicked documents, but that clicked documents that were displayed at lower ranks than a non-clicked document can be assumed to be preferred over the non-clicked document. For example, assume a query q, in response to which the system returns documents (d_1, d_2, d_3), in this order. If the user clicks on documents d_2 and d_3, but not on d_1, we can infer that $d_2 \succ d_1$ and $d_3 \succ d_1$. From these observations, labeled data could be extracted as $(d_1, d_2, -1)$ and $(d_1, d_3, -1)$.

Given a set of labeled document pairs, we apply the stochastic gradient descent (SGD) algorithm defined by Zhang (2004, Algorithm 2.1). This algorithm finds a weight vector $\hat{\mathbf{w}}$ that minimizes the empirical loss $L(\mathbf{w}, \mathbf{x}, y)$ given a set P of labeled training samples each consisting of a feature vector $\mathbf{x}$ and a label y:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \left[\frac{1}{P} \sum_{i=0}^{P} L(\mathbf{w}, \mathbf{x}_i, y_i) + \frac{\lambda}{2} ||\mathbf{w}||_2^2 \right], \tag{1}$$

where the last term is a regularization term. Using the hinge loss, i.e., $L(\mathbf{w}, \mathbf{x}, y) = \max(0, 1 - y\mathbf{w}^T\mathbf{x})$, the algorithm optimizes the same quantity as RankSVM (Joachims 2002). It was shown to perform competitively on standard learning to rank datasets in terms of ranking performance with only a fraction of the training time (Sculley 2009). Here, we follow the implementation provided in sofia-ml,[5] and apply it to pairwise learning by setting $x = (x_a - x_b)$, where x_a and x_b are the feature vectors of a document pair.

Combining the above method of inferring pairwise feedback and the pairwise learning method, we obtain our pairwise baseline algorithm (Algorithm 1). It receives as input a document set D, learning rate η, regularizer weight λ, and an initial weight vector $\mathbf{w}_0$. For each observed query q_t, a set of feature vectors $\phi(d_i|q)$ is extracted that characterize the relationship between the query and each candidate document $d_i \in D$. The document feature vectors are then scored using the weight vector learned so far ($\mathbf{w}_{t\text{-}1}$), and sorted by this score to obtain an exploitative result list (the best ranking given what has been learned so far).

The constructed exploitative result list is shown to the user, and clicks on any of the result documents are observed. From the observed clicks C, all possible labeled document pairs P are inferred using the pairwise labeling method described above (Joachims 2002).

The labeled samples in P are then used to update the weight vector $\mathbf{w}$. For each pair, the loss is obtained by comparing the current solution to the observed label (line 10, cf. the definition of the hinge loss above). If the labels do not match, or the prediction margin is too small, the weight vector is updated using the update rule $\mathbf{w}_t = \mathbf{w}_{t\text{-}1} + \eta y_i (\mathbf{x}_{a\_i} - \mathbf{x}_{b\_i}) - \eta\lambda\, \mathbf{w}_{t\text{-}1}$. Here, we use the unregularized version of this update rule (by setting $\lambda = 0$) and use a small constant η. This

[5] Provided online at http://code.google.com/p/sofia-ml.

Algorithm 1 Baseline algorithm for the pairwise setting, based on (Joachims 2002; Sculley 2009; Zhang 2004)

```
1: Input: $D$, $\eta$, $\lambda$, $\mathbf{w}_0$
2: for query $q_t$ ($t = 1..T$) do
3:     $\mathbf{X} = \phi(D|q_t)$          // extract features
       // construct exploitative result list
4:     $S = \mathbf{w}_{t-1}^T \mathbf{X}$
5:     $L$ = sortDescendingByScore($D$, $S$)
6:     $I = L$ [1:10]
7:     Display $I$ and observe clicked elements $C$.
8:     Construct all labeled pairs $P = (\mathbf{x}_a, \mathbf{x}_b, y)$ from $I$ and $C$.
9:     for $i$ in (1 .. $P$)do
10:        if $y_i \mathbf{w}_{t-1}^T (\mathbf{x}_{a\_i} - \mathbf{x}_{b\_i}) < 1.0$ and $y_i \neq 0.0$ then
11:            Update $w_t$ as: $\mathbf{w}_t = \mathbf{w}_{t-1} + \eta y_i (\mathbf{x}_{a\_i} - \mathbf{x}_{b\_i}) - \eta\lambda \mathbf{w}_{t-1}$
12: return $\mathbf{w}_t$
```

formulation was found to show good convergence properties (Zhang 2004) and resulted in good performance in preliminary experiments.

3.2.2 *Balancing exploration and exploitation*

In previous applications of pairwise learning to implicit feedback scenarios, learning was performed in a batch setting. First, implicit feedback was collected given an initial ranking function. Then, the algorithm was trained on all collected implicit feedback. Finally, this trained system was deployed and evaluated (Joachims 2002). In this setting, data collection is naturally exploitative. Users are shown results that are most likely to be relevant according to a current best ranking function. In the online setting, such an exploitative strategy is expected to result in the highest possible short-term performance. However, it is also expected to introduce bias, as some documents may never be shown to the user, which may result in sub-optimal learning and lower long-term performance. This is confirmed in our experiments, as we will see below.

In supervised applications of pairwise learning to rank methods, the learning algorithm is typically trained on the complete dataset. Sculley (2009) developed a sampling scheme that allows training of a stochastic gradient descent learner on a random subset of the data without noticeable loss in performance of the trained algorithm. In this setting, document pairs were sampled randomly such that at each learning step one relevant and one non-relevant document were selected to form a training pair. We expect this strategy, which is fully exploratory, to result in minimal training bias.

In the online setting where we learn from implicit feedback, we cannot directly determine for which document pairs we obtain feedback from the user. Any document list that is presented in response to a query may result in zero or more clicks on documents, such that zero or more pairwise constraints can be extracted. Due to position bias (Silverstein et al. 1999), the higher a document is ranked in the result list presented to the user, the more likely it is to be inspected and clicked.

Algorithm 2 Balancing exploration and exploitation in the pairwise setting

```
1: Input: $D, \eta, \lambda, \mathbf{w}_0, \epsilon$
2: for query $q_t$ ($t = 1..T$) do
3:     $\mathbf{X} = \phi(D|q_t)$        // extract features
       // construct exploitative result list
4:     $S = \mathbf{w}_{t-1}^T\mathbf{X}$
5:     $L$ = sortDescendingByScore($D$, $S$)
6:     $I[r] \leftarrow$ first element of $L \notin I$ with probability $\varepsilon$; element randomly sampled without replacement
       from $L \setminus I$ with probability $1 - \varepsilon$
7:     Display $I$ and observe clicked elements $C$.
8:     Construct all labeled pairs $P = (\mathbf{x}_a, \mathbf{x}_b, y)$ from $I$ and $C$.
9:     for $i$ in (1 .. $P$) do
10:        if $y_i (\mathbf{x}_{a\_i} - \mathbf{x}_{b\_i}) \mathbf{w}_{t-1}^T < 1.0$ and $y_i \neq 0.0$ then
11:            Update $w_t$ as: $\mathbf{w}_t = \mathbf{w}_{t-1} + \eta y_i (\mathbf{x}_{a\_i} - \mathbf{x}_{b\_i}) - \eta\lambda\, \mathbf{w}_{t-1}$
12: return $\mathbf{w}_t$
```

Here, we ignore explicit dependencies between displayed documents, and define two document lists, one exploratory and one exploitative, that are then combined to balance exploration and exploitation. The exploitative list is generated by applying the learned weights to compute document scores and then sorting by score, as in the baseline algorithm. The exploratory list is generated by uniform random sampling of the documents associated with a query.[6]

We employ a method for balancing exploration and exploitation that is inspired by ε-*greedy*, a commonly used exploration strategy in RL (Watkins 1989). In ε-greedy exploration, the agent selects an exploratory action with probability ε at each timestep. With probability 1 − ε, it selects the greedy action, i.e., the action with the highest currently estimated value.[7]

The difference between our approach and ε-greedy is that we do not pick a single action at each timestep, but that we can select a number of actions that are presented simultaneously. This results in Algorithm 2 that differs from our baseline algorithm in how the result list is constructed (line 6).

Therefore, we vary the relative number of documents from the exploratory and exploitative lists as determined by $\varepsilon \in [0, 1]$. For each rank, an exploitative action (a document from the exploitative list) is selected with probability 1 − ε. A document from

[6] In practice, candidate documents are typically collected based on some feature-based criteria, such as a minimum score. Here, we use the candidate documents provided with the learning to rank data sets used in our experiment, where candidate selection may have been biased (Minka and Robertson 2008). However, bias in terms of feature values can be neglected here, as the specifics of the learned ranker are not the subject of this study, and all learning methods are affected equally.

[7] More complex schemes of balancing exploration and exploitation are of course possible, but our focus here is on demonstrating the benefit of such a balance over purely exploratory and purely exploitative forms of soliciting feedback. A simple scheme is sufficient for this goal. We also experimented with a more complex softmax-like algorithm and obtained qualitatively similar results. However, such an algorithm is more difficult to tune than the ε-greedy-like algorithm used here (Sutton and Barto 1998; Whiteson and Stone 2006).

Algorithm 3 Baseline Algorithm for the listwise setting

```
1: Input: $f(l_1, l_2)$, $\alpha$, $\delta$, $\mathbf{w}_0$
2: for query $q_t$ ($t = 1..T$) do
3:     Sample unit vector $u_t$ uniformly.
4:     $\mathbf{w}'_t \leftarrow \mathbf{w}_t + \delta u_t$    // generate exploratory $\mathbf{w}$
5:     if $f(l(\mathbf{w}_t), l(\mathbf{w}'_t))$ then
6:         $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha u_t$    // update exploitative $\mathbf{w}$
7:     else
8:         $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$
9: return $\mathbf{w}_{t+1}$
```

the exploratory list is selected with probability ε. Thus, values of ε close to 0 mean that little exploration is taking place, making the algorithm collect feedback in an exploitative way ($\varepsilon = 0$ corresponds to the purely exploitative baseline setting). Values close to 1 mean more exploration.

3.3 Balancing exploration and exploitation in listwise learning to rank

Our second online learning to rank approach builds off a gradient-based policy search algorithm called Dueling Bandit Gradient Descent (DBGD) (Yue and Joachims 2009). This algorithm has been specifically developed for learning to rank in an online setting, and it requires only relative evaluations of the quality of two document lists, and infers such comparisons from implicit feedback (Radlinski et al. 2008b).

3.3.1 Baseline learning approach

Similar to the pairwise learning to rank approach discussed in the previous section, DBGD learns a ranking function consisting of a weight vector $\mathbf{w}$ for a linear weighted combination 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 $\mathbf{S} = \mathbf{w}\,\mathbf{X}$. Finally, documents are ranked by their scores to generate a ranked document list l.

Algorithm 3 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[8] and δ, and an initial weight vector $\mathbf{w}_0$. 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 $\mathbf{w}_t$, found to perform best up to the current timestep t. The exploratory list is produced from an exploratory weight vector $\mathbf{w}'_t$, which is generated by moving $\mathbf{w}_t$ in a random direction $\mathbf{u}_t$ 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 $\mathbf{w}'_t$ is judged to have produced the better document ranking, the current exploitative weight vector $\mathbf{w}_t$ is updated by moving it towards $\mathbf{w}'_t$ by a step size α.

[8] In (Yue and Joachims 2009), γ denotes the exploitation step size. We use α to avoid confusion with the discount factor γ.

For the comparison method $f(l_1, l_2)$, several implementations have been suggested (He et al. 2009; Radlinski et al. 2008b). 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 (He et al. 2009) 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.

3.3.2 *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.

Constructing result lists by probabilistic interleaving, allows us to apply a method similar to ε-greedy. Our probabilistic interleave algorithm, which supplies the comparison method that is required by DBGD, is shown in Algorithm 4. 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 list l_i ($i = 1, 2$) if the document is in the top N of l_i, where N is the lowest-ranked click.

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.[9] 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

[9] Note that the setting $k = 0.5$ corresponds to the fully exploratory baseline algorithm. Setting $k > 0.5$ would not increase the amount of information that can be gained from a comparison, but would hurt the expected reward, because fewer exploitative documents would be shown.

Algorithm 4 $f(l_1, l_2)$ – k-greedy comparison of document lists

```
1: Input: $l_1, l_2, k$
2: initialize empty result list $I$
   // construct result list
3: for rank $r$ in (1 .. 10) do
4:     $L \leftarrow l_1$ with probability $k$, $l_2$ with probability $1 - k$
5:     $I[r] \leftarrow$ first element of $L \notin I$
6: display $I$ and observe clicked elements $C$
7: $N = length(C)$; $c_1 = c_2 = 0$
8: for $i$ in (1 .. $N$) do
9:     if $C[i] \in l_1[1{:}N]$ then
10:        $c_1 = c_1 + 1$    // count clicks on $l_1$
11:    if $C[i] \in l_2[1{:}N]$ then
12:        $c_2 = c_2 + 1$    // count clicks on $l_2$
   // compensate for bias (Eq. 2)
13: $n_1 = | l_1[1{:}N] \cap I[1{:}N] |$
14: $n_2 = | l_2[1{:}N] \cap I[1{:}N] |$
15: $c_2 = \frac{n_1}{n_2} * c_2$
16: return $c_1 < c_2$
```

$$E[c_2] = \frac{n_1}{n_2} * E[c_1], \tag{2}$$

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 some bias remains because the observed clicks are converted to binary preference decisions before they are aggregated over queries. 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, robust bias correction.

4 Experiments

In this section, we describe the experiments that evaluate the algorithms presented in Sect. 3. We first give an overview of the general evaluation setup, before giving further details on specific aspects such as the datasets and runs.

4.1 Evaluation setup

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. Also, live experiments cannot evaluate and compare many different settings like simulated experiments can.

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 Fig. 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 used to simulate the user feedback and measure cumulative performance.

Using simulated evaluations naturally has limitations. Here, we can show how learning methods behave under different assumptions about user behavior, but to what degree these assumptions apply in specific practical settings needs to be studied in more detail.

4.2 Click model

Our click model is based on the Dependent Click Model (DCM) (Guo et al. 2009a, b), a generalization of the cascade model (Craswell et al. 2008). 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 (Guo et al. 2009b), we believe it is a good model for generating implicit feedback.

When a user examines a document in the result list, he or she does not know the true relevance label of the document. However, aspects of the document's representation in the result list (e.g., its 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 the 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 algorithms 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 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

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. We further implement two realistic models, the *navigational* and *informational* models. These two models are based on typical user behavior in web search (Broder 2002; Guo et al. 2009a), 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 big 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.

4.3 Data

We conducted our experiments using two standard collections for learning to rank: LETOR 3.0 and LETOR 4.0 (Liu et al. 2007). 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 and 150 queries each, with about 1,000 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, 1,700 and 800 respectively, but far 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.[10]

[10] 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.

Each dataset comes split up for machine learning experiments using fivefold 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.

4.4 Runs

4.4.1 *Pairwise approach*

In all pairwise experiments, we initialize the starting weight vector $\mathbf{w}_0$ to zero. In preliminary experiments we evaluated offline performance for $\eta \in \{0.0001, 0.001, 0.01, 0.1\}$, and selected the setting that performed best over all data sets ($\eta = 0.001$). Our *baseline* is the pairwise formulation of learning to rank with stochastic gradient descent as described in Sect. 3.2, in the fully exploitative setting ($r = 0$). Against this baseline we compare increasingly exploratory versions of the algorithm ($r \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$). All experiments are run for 1000 iterations.

4.4.2 *Listwise approach*

In all listwise experiments, we initialize the starting weight vector $\mathbf{w}_0$ to a random point on a unit sphere around the origin, and use the best performing parameter settings from (Yue and Joachims 2009): $\delta = 1$ and $\alpha = 0.01$ (these settings resulted in good performance over all datasets in our preliminary experiments). Our *baseline* is Algorithm 3, based on (Yue and Joachims 2009), which corresponds to a purely exploratory setting of $k = 0.5$ in our extended method.[11] 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 4. Again, we run all experiments for 1000 iterations.

4.5 Discounting

Because our problem formulation assumes an infinite horizon, cumulative performance is defined as an infinite sum of discounted rewards (cf. Sect. 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.

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 (Sutton and Barto 1998). 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.

4.6 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

[11] In the listwise approach, the highest level of exploration is reached when the two candidate lists are interleaved in equal parts, i.e., $k = 0.5$.

that the retrieval system accrues throughout the length of the experiment. Final performance is reported in terms of NDCG on the test set.

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 (purely exploitative for the pairwise approach ($k = 0.0$), purely exploratory for the listwise approach ($k = 0.5$)) 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 primary goal of this paper is to show that balancing exploration and exploitation in online learning to rank for IR can improve cumulative performance. As there is currently little known about the performance of pairwise and listwise learning to rank approaches in an online setting, we first report on the performance of each baseline algorithm, before turning to our main results. Finally, we compare the performance of the two approaches under the novel perspective of balancing exploration and exploitation.

5.1 Pairwise learning

As a first sanity check, we calculate standard evaluation measures that are typically used to evaluate supervised learning to rank methods for the baseline approach (purely exploitative, i.e., $r = 0.0$). Results in terms of NDCG at 10 (*NDCG@10*), Precision at 10 (*P@10*), and Mean Average Precision (MAP) (Liu 2009) for the *perfect* click model are shown in Table 2. These are computed after 1000 iterations (consisting of 1 query, result list, and learning step each), which means that learning may not have converged and higher results are possible. These results should therefore be interpreted as a rough indication of what performance can typically be achieved by this approach in an implicit feedback setting.

Based on the performance of the pairwise approach relative to state-of-the-art supervised approaches (Liu 2009), we can divide the datasets used in our experiments in two groups. For the first group, which includes the datasets *HP2003, HP2004, NP2003*, and *NP2004*, performance is quite high. In particular, for the datasets *HP2003* (*NDCG@10*=0.820) and *HP2004* (*NDCG@10*=0.803), the pairwise exploitative learner is among the top 3 approaches and even beats the supervised version of RankSVM (with an *NDCG@10* of 0.807 and 0.768, respectively). For the second group, which includes the

Table 2 NDCG@10, P@10, and MAP for the pairwise baseline algorithm (*perfect* click model)

	NDCG@10	P@10	MAP
HP2003	0.820	0.106	0.753
HP2004	0.803	0.099	0.705
NP2003	0.794	0.093	0.676
NP2004	0.796	0.095	0.672
TD2003	0.271	0.153	0.208
TD2004	0.279	0.217	0.183
OHSUMED	0.321	0.392	0.371
MQ2007	0.377	0.343	0.412
MQ2008	0.490	0.239	0.452

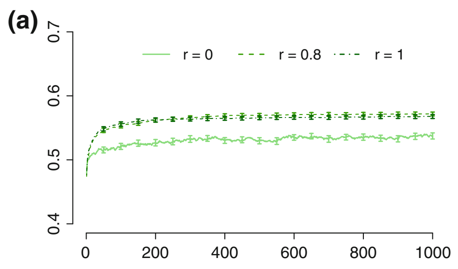

Fig. 2 Final performance for the pairwise approach (with 5 % confidence intervals) over time for the datasets (**a**) *MQ2007* and (**b**) *NP2003*, under the *perfect* click model and $r \in \{0.0, 0.8, 1.0\}$.

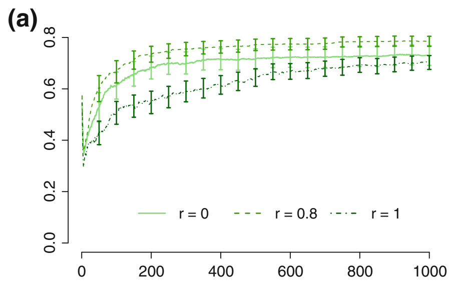

Fig. 3 Final performance for the pairwise approach (with 5 % confidence intervals) over time for the dataset *NP2003* for **a** *navigational*, and **b** *informational* click models and $r \in \{0.0, 0.8, 1.0\}$

remaining datasets, performance is lower both in absolute terms and relative to state-of-the-art supervised learning approaches. Here, performance is comparable to the mid-range of the approaches compared in (Liu 2009).

To analyze the performance differences between datasets, we study the learning curves of two representative datasets, *MQ2007* and *NP2003*, at different levels of exploration. Figure 2 shows the final performance in terms of NDCG (on the whole result list) plotted over time (up to 1,000 iterations). For the dataset *MQ2007* we see that best learning is achieved at high exploration rates (the dark, dashed lines; the difference between settings $r = 0.8$ and $r = 1.0$ is negligible). As expected, when implicit feedback is collected on fully exploratory result lists, it is unbiased and thus results in the best learning. However, we see the opposite for the dataset *NP2003*, where performance further improves when implicit feedback is collected on exploitative result lists ($r = 0$, light and solid lines). Most likely, this increase in performance results from an effect similar to that observed in active learning. Because the current top results are shown, feedback is focused on the part of the document space that is most informative for learning. All datasets for which this effect is observed have few relevant documents, so that focusing feedback on a promising region can have a substantial benefit. For the remaining datasets, there are more relevant documents per query. Thus, feedback that is focused on one part of the solution space (in the exploitative setting) does not improve performance.

For all datasets, the absolute difference in final performance at varying exploration rates is relatively small (similar to the two datasets analyzed above) under the *perfect* click model. Much higher variance is observed when we simulate noisy feedback. Figure 3 shows learning curves for the dataset *NP2003* at different settings of *r* for the *navigational* and *informational* click models. For the *navigational* click model (a) final performance improves over time for all r, although differences in performance for different r are higher than with perfect click feedback. Final performance is lowest for $r = 1.0$ (0.714), and increases for $r = 0.0$ (0.720) and $r = 0.8$ (0.778). For the *informational* click model, final performance degrades dramatically in the purely exploitative baseline settings ($r = 0$, 0.102). In this setting, performance decreases over time. The purely exploratory setting ($r = 1.0$) leads to reasonable final performance, while the best performance is achieved with high exploration but some exploitation ($r = 0.8$, 0.724).

Our analysis of final performance results in a number of observations. We hypothesized that the best learning would occur with perfect feedback and pure exploration because this setting minimizes variance and bias in user feedback. As expected, learning outcomes were best for perfect feedback and degraded with noisier feedback. However, the effect of the exploration rate changed with the amount of noise in user feedback and characteristics of the dataset. For perfect feedback, little to no exploration sometimes produced the best learning outcomes because exploitative result lists focused feedback on more informative parts of the solution space. Under noisy feedback, higher exploration rates generally improved learning, though the best performance occurred with moderate amounts of exploitation.

5.1.1 Balancing exploration and exploitation

We now turn to our main results for the pairwise approach: the influence of balancing exploration and exploitation on online performance. Above, we saw that the effect of this balance on offline performance is complex and that more exploration does not necessarily imply better learning. When optimizing online performance, a system should generally try to exploit as much as possible. However, if increasing exploration results in sufficiently high learning gains, the short-term cost of this exploration may be outweighed by long-term benefits, as it would increase the quality of exploitative result lists later on.

Table 3 shows our results for comparing runs with varying amounts of exploration ($r \in [0.2, 1.0]$) to the purely exploitative baseline ($r = 0.0$). The best runs per row are highlighted in bold and significant differences are marked as described above. For the *perfect* click model, the best performance is achieved in the baseline settings for 7 out of 9 datasets, ranging from 103.05 (*TD2003*) to 124.75 (*HP2003*). For these datasets, the best learning is achieved at low exploration rates, so that increasing exploration cannot lead to long-term benefits. The modest improvement in offline performance at slightly higher exploration rates (e.g., as for datasets *TD2003* and *TD2004* at $r = 0.2$) is not large enough to outweigh the cost of the increased exploration.

Exceptions are the datasets *OHSUMED* and *MQ2007*. For both datasets, online performance improves significantly when the exploration rate is increased to $r = 0.2$, with a relative increase of 12.9 % for *OHSUMED* and 2.8 % for *MQ2007*. As discussed above, these datasets have more relevant documents per query. Consequently, the expected utility is high, even when result lists are randomized (this can also be observed when comparing online performance at the purely exploratory setting $r = 1.0$). As a result, the cost of exploration is relatively low and easily outweighed by the long-term learning gains.

In the relatively reliable *navigational* click model, optimal online performance is achieved at the slightly higher exploration rate of $r = 0.2$ for all but one dataset. For 5 of these datasets, this improvement is significant when compared to the baseline setting. Compared to the *perfect* click model, online performance with the noisier *navigational* model is typically lower, as expected. The performance loss due to noise is between 3.5 %

Table 3 Results for the pairwise approach. Online performance (in terms of cumulative NDCG) over 1000 iterations

r	0.0	0.2	0.4	0.6	0.8	1.0
click model: perfect						
HP2003	**124.75**	122.32	109.82▼	89.50▼	61.12 ▼	1.35 ▼
HP2004	**115.08**	102.15 ▼	87.14 ▼	72.64 ▼	48.79 ▼	1.14 ▼
NP2003	**115.75**	110.72	96.21 ▼	87.43 ▼	54.94 ▼	1.84 ▼
NP2004	**111.26**	104.43	94.37 ▼	76.60 ▼	52.44 ▼	1.17 ▼
TD2003	**103.05**	98.88	85.37 ▼	69.65 ▼	43.63 ▼	8.23 ▼
TD2004	**106.77**	103.93	83.06 ▼	65.56 ▼	44.50 ▼	14.62 ▼
OHSUMED	119.13	**134.53** ▲	127.31 ▲	119.68	111.42 ▼	100.44 ▼
MQ2007	104.12	**107.06** ▲	105.50 ▲	101.53 ▼	95.96 ▼	90.13 ▼
MQ2008	**104.01**	103.38	98.96 ▼	93.70 ▼	87.58 ▼	80.60 ▼
click model: navigational						
HP2003	104.99	**111.01**	108.15	102.48	69.91 ▼	1.39 ▼
HP2004	91.69	**110.26** △	107.83 △	82.70	57.20 ▼	1.27 ▼
NP2003	**115.85**	110.40	106.50	91.94 ▼	61.19 ▼	1.88 ▼
NP2004	93.79	**116.11** ▲	106.39	89.67	64.20 ▼	1.07 ▼
TD2003	64.43	**85.20** ▲	77.85 ▲	66.59	42.80 ▼	8.08 ▼
TD2004	87.59	**97.90** ▲	82.91	66.31 ▼	45.46 ▼	14.30 ▼
OHSUMED	127.05	**129.73**	123.39 ▼	117.64 ▼	110.04 ▼	100.81 ▼
MQ2007	101.68	**102.88** △	101.99	99.13 ▼	95.08 ▼	90.05 ▼
MQ2008	100.16	**100.41**	97.18 ▼	93.06 ▼	87.31 ▼	81.23 ▼
click model: informational						
HP2003	7.21	40.39 ▲	**72.60** ▲	66.52 ▲	55.31 ▲	1.37 ▼
HP2004	6.39	29.81 ▲	51.81 ▲	**67.20** ▲	46.30 ▲	1.11 ▼
NP2003	5.75	24.39 ▲	55.16 ▲	**60.57** ▲	45.04 ▲	1.90 ▼
NP2004	5.64	23.95 ▲	**69.22** ▲	65.58 ▲	52.01 ▲	1.21 ▼
TD2003	7.47	23.64 ▲	43.92 ▲	**43.96** ▲	36.25 ▲	7.85
TD2004	17.31	50.10 ▲	**60.12** ▲	54.58 ▲	38.44 ▲	14.48 ▽
OHSUMED	102.60	121.48 ▲	**122.06** ▲	116.55 ▲	109.53 ▲	101.12
MQ2007	92.76	96.58 ▲	**98.19** ▲	96.66 ▲	95.43 ▲	90.00 ▼
MQ2008	90.00	91.14	**92.45** ▲	91.12	86.88 ▼	81.69 ▼

Best runs per row are highlighted in bold

(*MQ2008*) and 11.0 % (*HP2003*). One exception is the dataset *NP2003*, for which learning is not strongly affected by the small increase in noise and where performance is close to identical to the *perfect* click model. The second exception is the dataset *NP2004*, for which online performance for the *navigational* click model at the best settings ($r = 0.2$) is 4.4 % higher than for the best setting and the *perfect* click model. Interestingly, this result demonstrates that a small amount of noise and mild exploration can lead to a better exploration of the solution space and an improvement in online performance.

As in earlier results, we see a difference between datasets with high and low ratios of relevant documents. For datasets with a relatively low ratio of relevant documents (*HP**, *NP**, and *TD**) the relative improvement in online performance at this higher exploration rate is typically big (up to 32.2 % for the dataset *TD2003*). Although the short-term cost of exploration for these datasets is high, this high cost is outweighed by even higher long-term benefits (being able to rank the few relevant documents well). For datasets with a high ratio of relevant documents, both the cost of exploration, and its long-term benefits are small. This leads to relative improvements of between 0.3 % (*MQ2008*) and 2.1 % (*OHSUMED*). For dataset *NP2003*, increased exploration does not lead to better online performance.

In the noisier *informational* click model, the trends observed for the *navigational* click model continue. Performance in the purely exploitative setting is substantially lower than for the other click models, as the increase in noise results in lower learning gains and therefore lower online performance. In this setting, the cost of exploration decreases relative to its benefit, so optimal performance is seen at higher exploration rates. For 6 datasets, the best online performance is achieved at $r = 0.4$; for the remaining 3 datasets the best setting is $r = 0.6$. All improvements are statistically significant when compared to the purely exploitative baseline. For the datasets with a low ratio of relevant documents, online performance improves by as much as an order of magnitude. For datasets with a high ratio of relevant documents, improvements range from 2.7 % (*MQ2008*) to 19.0 % (*OHSUMED*). Compared to the *navigational* click model, online performance drops for all datasets, as expected. The increase in noise leads to a drop in online performance of between 4.6 % (*MQ2007*) and 48.4 % (*TD2003*).

Overall, for the pairwise approach the effect of balancing exploration and exploitation depends both on noise in user feedback and characteristics of the datasets. When feedback is reliable, online performance is best in the purely exploitative (baseline) setting. However, the pairwise baseline approach does not learn well when feedback is noisy. Increasing the exploration rate can partially compensate for this. As a result, the best online performance under noisy feedback is achieved at $r = 0.4$ and $r = 0.6$, for which result lists contain about one half exploratory and one half exploitative documents. These findings confirm our hypothesis that balancing exploration and exploitation in the pairwise approach improves online performance.

5.2 Listwise learning

As with the pairwise approach, we start by evaluating the final performance of the listwise baseline approach ($k = 0.5$) in terms of standard evaluation measures. Results for the *perfect* click model after 1000 iterations are listed in Table 4. In this setting, performance is slightly lower than for the pairwise approach for 6 of the 9 datasets used (except for *TD2003, TD2004*, and *OHSUMED*). Despite the limited information available to the

Table 4 NDCG@10, P@10, and MAP for the listwise baseline algorithm (*perfect* click model)

	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

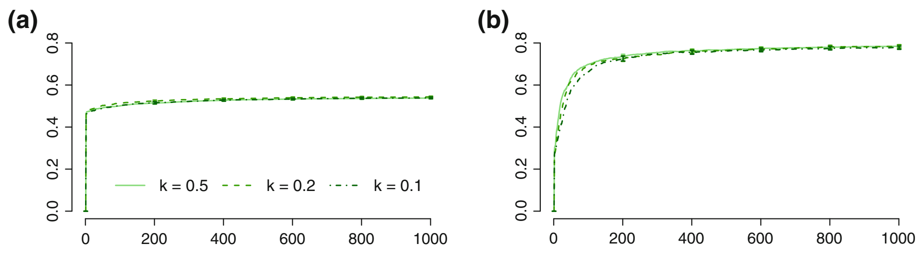

Fig. 4 Final performance (with 5 % confidence intervals) over time for the datasets **a** *MQ2007* and **b** *NP2003* for the *perfect* click model and $k \in \{0.1, 0.2, 0.5\}$

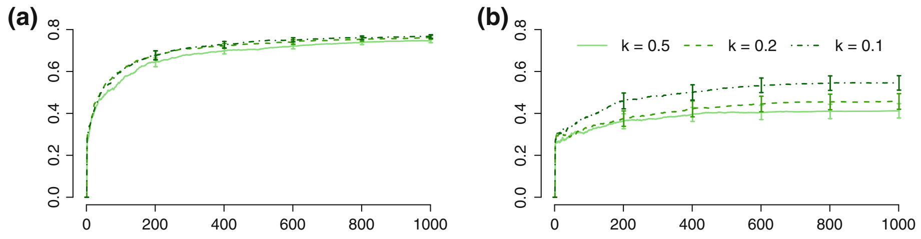

Fig. 5 Final performance (with 5 % confidence intervals) over time for the dataset *NP2003* for **a** *navigational*, and **b** *informational* click models and $k \in \{0.1, 0.2, 0.5\}$

algorithm (relative quality of the result list instead of explicit relevance judgment per document), performance is competitive with current supervised learning to rank algorithms (Liu 2009).

Figure 4 shows the learning curves for the datasets *MQ2007* and *NP2003* at different settings of *k* and the *perfect* click model. In contrast to the pairwise approach, there is no significant difference in performance after 1000 iterations for either dataset. For *NP2003*, learning in the fully exploratory setting ($k = 0.5$) is slightly faster than in other settings. This is expected, as the best feedback is available at maximal exploration. However, learning at lower exploration rates quickly catches up. Thus, for the listwise approach the exploration rate does not appear to have a significant effect on offline performance when feedback is perfect.

Learning curves for the *navigational* and *informational* click models are shown in Fig. 5. 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, final performance is between 0.412 and 0.546. Although final performance drops substantially as implicit feedback becomes extremely noisy, performance improves over time for all datasets as there is still a signal, i.e., relevant documents are more likely to be clicked than non-relevant ones.

Once again there is an interaction effect between click model and exploration rate, although it is different from that of the pairwise approach. Here, there is no significant difference between the final performance at different settings of *k* under the *perfect* click model. Under the *navigational* click model, the effect of noise is small, and results are similar to the *perfect* click model. 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 most reliable for the fully exploratory setting $k = 0.5$. Instead, it appears that, since bias is only partially compensated for (cf., Sect. 3), the bias that remains 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.

5.2.1 *Balancing exploration and exploitation for the listwise approach*

Our main results for the listwise approach are shown in Table 5. Here, we take the original, purely exploratory, form of the algorithm as the baseline ($k = 0.5$) against which we

Table 5 Results for the listwise approach. 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

Best runs per row are highlighted in bold compare *exploit* runs ($k \in [0.1, 0.4]$). With the *perfect* click model, all lower settings of k outperform the baseline in all datasets. For $k < 0.4$, all improvements over the baseline are statistically significant. The improvements range from 4.1 % (*OHSUMED*) to 12.35 % (*NP2004*).

Results for the *navigational* click model are similar. For all datasets, there are several lower settings of k which outperform 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, more exploitative settings outperform the purely exploratory baseline 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*.

Together, these results demonstrate 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 5 with final performance in Table 4, we find that cumulative performance does not depend only on final performance. For example, NDCG@10 and MAP for *HP2003* are higher than for *OHSUMED*, but cumulative performance is 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). The main factors affecting cumulative performance are the speed of learning and how effectively early learning gains are exploited. Thus, these results underscore the inadequacy of final performance as an evaluation measure for 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 online performance for all levels of noise in click feedback. This means that surprisingly little exploration is sufficient for good performance and that the original listwise algorithm explores too much. In contrast, the best setting for the pairwise approach depends on the amount of noise in the click model. When feedback is perfect, little to no exploration is needed. For the *informational* click model, the optimal setting of $r \in \{0.4, 0.6\}$ means that up to 60 % of the result list needs to be exploratory.

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 the baseline and best settings for *NP2003* is 21.9 %, while for *MQ2007* the difference is only 0.54 %. Like the differences observed under the pairwise approach, these are a consequence of the differences in the relative number of relevant documents per query. For datasets with few relevant documents, the differences between the exploitative and exploratory document lists are large, leading to a high cost of exploration but also high potential gains if this exploration leads to better learning. For datasets *MQ2007, MQ2008*, and *OHSUMED*, which contain a high ratio of relevant documents, the cost of exploration and the potential long-term gains from improved learning are relatively small. 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.

Cumulative performance is also affected by noise in click feedback, as observed in the results obtained for the different click models. Performance is highest 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.

5.3 Comparing the pairwise and listwise approach

For both the pairwise and the listwise approaches, our results confirm our hypothesis that a balance between exploration and exploitation is needed to optimize online performance. The mechanisms of how such a balance affects online performance, however, differ between the two learning approaches. Below, we first discuss how exploration impacts performance for both approaches. Then, we turn to an overall comparison of the online and offline performance of the two approaches, and conclude with implications for putting them in practice.

An unexpected outcome of our analysis is that increasing exploration does not always lead to better learning, i.e., better offline performance. As a result, exploitation can be "free," i.e., the current best solution is also the most useful for collecting feedback for learning. We observed this, e.g., for the pairwise approach under *perfect* click feedback. As click feedback on the exploitative list automatically focuses on an informative part of the document space, increasing exploration does not result in better learning. For the listwise approach, differences in learning for different exploration rates are small under the perfect click model. However, exploration does lead to faster convergence, so exploitation has a small cost.

When click feedback is noisy, the pairwise approach benefits from increased exploration, while the opposite is true for the listwise approach. Exploitation introduces bias in the pairwise approach, which focuses feedback on a relatively narrow part of the solution space that is not informative. Depending on the amount of noise, this effect can result in low final performance. Increasing exploration reduces this bias and improves final performance. As a result, the optimal balance between exploration and exploitation shifts towards increased exploration as feedback becomes noisier. For the listwise approach, an increase in noise also results in some bias. However, in this approach, the bias does not affect the inferred preference judgments (it only increases their variance) but acts as a safeguard against too frequent updates based on noisy data. This leads to less frequent but more reliable updates of the weight vector, thereby improving offline performance. Thus, as noise in click feedback increases, the long-term cost of exploitation increases for the pairwise approach, while it decreases for the listwise approach.

Final performance of the pairwise algorithm with the *perfect* click model is generally high, compared to state-of-the-art supervised learning to rank methods. Final performance also tends to be higher than for the listwise approach (winning on 6 out of the 9 datasets studied), although these results should be considered preliminary as we did not tune the approaches to the characteristics of the datasets and higher performance may be possible.

In terms of online performance, however, the two approaches perform similarly, with the listwise approach beating the pairwise approach on 4 datasets. An advantage of the listwise approach is that the cost of exploitation can be small if the exploratory document list is similar to the exploitative one, which is more likely as learning progresses. For the pairwise approach, the cost of exploration is generally high, so the approach has a disadvantage when a similar level of exploration is required for reasonable learning gains. Thus, at similar final performance and exploration rates, the listwise approach tends to achieve higher online performance than the pairwise approach.

Differences in how implicit feedback is processed by the two approaches lead to pronounced differences in their behavior when feedback is noisy. The performance of the pairwise approach can drop dramatically, especially in the original exploitative setting where often no learning is possible. However, balancing exploration and exploitation allows the algorithm to recover its performance. A relatively high amount of exploration, with about half the result list constructed from exploratory documents, is needed to achieve good learning outcomes and high online performance. The drop in performance due to noise is much less pronounced for the listwise method. Online performance of the algorithm in its original, fully exploratory, version is often an order of magnitude higher than for the original version of the pairwise approach. A possible reason is that, by aggregating feedback over document lists, the algorithm becomes inherently robust to noise. Increasing exploitation can further improve online performance. However, after balancing exploration and exploitation, the two algorithms perform similarly, with the pairwise approach winning on 4 datasets, and the listwise approach winning on 5.

Our analysis suggests that the pairwise and listwise approaches are appropriate for learning from implicit feedback in different settings. If the implicit feedback is known to be reliable, then the pairwise approach should be preferred as it results in good offline performance. Also, in this setting, the pairwise approach requires little to no exploration for substantial learning gains. Thus, it can exploit aggressively, leading to high online performance. However, if feedback is expected to be noisy, the listwise approach should be preferred. In contrast to the pairwise approach, it safeguards against dramatic loss in offline performance, as long as there is some signal in the feedback that prefers truly relevant documents. In addition, under noisy feedback, the listwise approach requires much less exploration than the pairwise approach, and the cost of exploration is lower.

6 Conclusion

We demonstrated that balancing exploration and exploitation can substantially and significantly improve online performance in online learning to rank for IR. We introduced two methods for balancing exploration and exploitation in this setting, based on one pairwise and one listwise learning approach. To the best of our knowledge, these are the first algorithms that can achieve such a balance in a setting where only relative feedback is available.

Our experimental results demonstrated that the effect of balancing exploration and exploitation is complex and that there is an interaction effect between the amount of exploitation and the amount of noise in user feedback. When feedback is reliable, both approaches learn well and a high amount of exploitation can be tolerated, which leads to high online performance. As feedback becomes noisier, learning under high exploitation becomes unreliable for the pairwise approach. A higher amount of exploration is required to maintain reasonable performance. For the listwise approach, however, a smoothing effect occurs under high exploitation, so that exploitation is essentially cost-free. This allows the listwise approach to maintain good performance under noisy feedback with a surprisingly small amount of exploration.

Our results also shed new light on the relative performance of online learning to rank methods. The pairwise approach makes very effective use of implicit feedback when there is little noise, leading to final performance comparable to supervised learning to rank approaches. However, it is strongly affected by noise in user feedback. Our results demonstrated that a balance of exploration and exploitation is crucial in such a setting. For the listwise approach, learning can be slower but, due to the aggregation of feedback over multiple result lists, this approach is more robust to noise.

After demonstrating the importance of balancing exploration and exploitation in online learning to rank, a crucial question is how best to explore. This is the direction we are planning to pursue in future work. For the pairwise approach, the cost of random exploration is high. Exploration methods based on active learning approaches (Donmez and Carbonell 2009; Tian and Lease 2011; Xu et al. 2007) are a promising alternative that may reduce its effect on short-term performance, while maintaining or even improving long-term learning. For the listwise approach, the current form of random exploration can lead to inefficient learning. Unlike for the pairwise setting, we are not aware of any active learning methods for the listwise setting that could be used for smarter exploration. Obvious starting points are methods for exploration in policy search reinforcement learning (Kalyanakrishnan and Stone 2010; Whiteson and Stone 2006).

A straightforward way to further improve online performance would be to use annealing schedules that reduce the exploration rate over time, as is common practice in RL. While this technique could improve absolute performance, it is not expected to affect the qualitative outcomes of our analysis. In realistic settings its applicability may be limited as such problems are likely non-stationary, meaning that the exploration rate should never become so small that the retrieval system cannot adapt to changes in user preferences. A more interesting question is whether and how the system could automatically adjust the amount of exploration to a specific setting.

In this paper we focused on evaluating our methods using simulations. This had the advantage that we could conduct experiments for various settings, such as varying the amount of noise in click feedback. Naturally, assumptions underlying these simulations need to be tested before moving these methods to a real setting. Based on our results, we believe that it is particularly important to study the level of noise in user feedback, and collection characteristics such as the typical ratio of relevant and non-relevant documents before applying the methods in a specific environment. Based on such statistics, an informed choice about the appropriate online learning approach and a good balance of exploration and exploitation can be made.

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, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreements nr 258191 (PROMISE Network of Excellence) and 288024 (LiMoSINe project), the Netherlands Organisation for Scientific Research (NWO) under project nrs 612.061.814, 612.061.815, 640.004.802, 380-70-011, 727.011.005, the Center for Creation, Content and Technology (CCCT), the Hyperlocal Service Platform project funded by the Service Innovation & ICT program, the WAHSP project funded by the CLARIN-nl program, the Dutch national program COMMIT, and by the ESF Research Network Program ELIAS.

References

Agarwal, D., Chen, B., Elango, P., Motgi, N., Park, S., Ramakrishnan, et al. (2008). Online models for content optimization. In: NIPS'08, pp 17–24.

Auer, P. (2003). Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research, 3*, 397–422.

Barto, A. G., Sutton, R. S., & Brouwer, P. S. (1981). Associative search network: A reinforcement learning associative memory. *IEEE Transaction on System, Man, and Cybernetics, 40*, 201–211.

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

Cohen, J. D., McClure, S. M., & Yu, A. J. (2007). Should I stay or should I go? How the human brain manages the trade-off between exploitation and exploration. *Philosophical Transactions of the Royal Society B: Biological Sciences, 362*(1481), 933–942.

Craswell, N., Zoeter, O., Taylor, M., & Ramsey, B. (2008). An experimental comparison of click position-bias models. In: WSDM '08, pp 87–94.

Donmez, P., & Carbonell, J. (2009). Active sampling for rank learning via optimizing the area under the ROC curve. In: ECIR'09, pp 78–89.

Gittins, J. C. (1979). Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society Series B (Methodological) 41*(2), 148–177.

Guo, F., Liu, C., & Wang, Y. M. (2009b). Efficient multiple-click models in web search. In: WSDM '09, pp 124–131.

Guo, F., Li, L., & Faloutsos, C. (2009a). Tailoring click models to user goals. In: WSCD '09, pp 88–92.

He, J., Zhai, C., & Li, X. (2009). Evaluation of methods for relative comparison of retrieval systems based on clickthroughs. In: CIKM '09, pp 2029–2032.

Herbrich, R., Graepel, T., & Obermayer, K. (1999). Support vector learning for ordinal regression. In: ICANN '99, vol 1, pp 97–102.

Hofmann, K., Whiteson, S., & de Rijke, M. (2011a). Balancing exploration and exploitation in learning to rank online. In: ECIR'11, pp 251–263.

Hofmann, K., Whiteson, S., & de Rijke, M. (2011b). A probabilistic method for inferring preferences from clicks. In: CIKM '11, pp 249–258.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In: KDD '02, pp 133–142.

Joachims, T., Granka, L., Pan, B., Hembrooke, H., Radlinski, F., & Gay, G. (2007). Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. ACM Transaction of Information and System 25(2):7+.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996) Reinforcement learning: A survey. *Journal Artifical Intelligence Research 4*(1), 237–285.

Kalyanakrishnan, S., & Stone, P. (2010). Efficient selection of multiple bandit arms: Theory and practice. In: ICML'10, pp 511–518.

Karimzadehgan, M., & Zhai, C. (2010). Exploration-exploitation tradeoff in interactive relevance feedback. In: CIKM '10, ACM, New York, NY, USA, pp 1397–1400.

Langford, J., & Zhang, T. (2008). The epoch-greedy algorithm for multi-armed bandits with side information. In: NIPS'08, pp 817–824.

Langford, J., Strehl, A., & Wortman, J. (2008). Exploration scavenging. In: ICML '08, pp 528–535.

Li, L., Chu, W., Langford, J., & Wang, X. (2011). Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In: WSDM '11, pp 297–306.

Li, L., Chu, W., Langford, J., & Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In: WWW '10, pp 661–670.

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

Liu, T. Y., Xu, J., Qin, T., Xiong, W., & Li, H. (2007). Letor: Benchmark dataset for research on learning to rank for information retrieval. In: LR4IR '07.

Mahajan, A., & Teneketzis, D. (2008). Multi-armed bandit problems. Foundations and Applications of Sensor Management pp 121–151.

Minka, T., & Robertson, S. (2008). Selection bias in the LETOR datasets. In: SIGIR Workshop on Learning to Rank for Information Retrieval, pp 48–51.

Radlinski, F., & Craswell, N. (2010). Comparing the sensitivity of information retrieval metrics. In: SIGIR '10, pp 667–674.

Radlinski, F., Kleinberg, R., & Joachims, T. (2008a). Learning diverse rankings with multi-armed bandits. In: ICML '08, ACM, pp 784–791.

Radlinski, F., Kurup, M., & Joachims, T. (2008b). How does clickthrough data reflect retrieval quality? In: CIKM '08, pp 43–52.

Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society 58*, 527–535.

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

Sculley, D. (2009). Large scale learning to rank. In: NIPS 2009 Workshop on Advances in Ranking.

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

Strehl, A., Mesterharm, C., Littman, M., & Hirsh, H. (2006). Experience-efficient learning in associative bandit problems. In: ICML '06, pp 889–896.

Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning: An introduction. MIT Press, Cambridge, MA, USA.

Taylor, M., Guiver, J., Robertson, S., & Minka, T. (2008). Softrank: optimizing non-smooth rank metrics. In: WSDM '08, ACM, pp 77–86.

Tian, A., & Lease, M. (2011). Active learning to maximize accuracy vs. effort in interactive information retrieval. In: SIGIR '11, pp 145–154.

Watkins, C. (1989). Learning from delayed rewards. PhD thesis, Cambridge University.

Whiteson, S., & Stone, P. (2006). On-line evolutionary computation for reinforcement learning in stochastic domains. In: GECCO 2006: Proceedings of the genetic and evolutionary computation conference, pp 1577–1584.

Xu, Z., & Akella, R. (2008). A Bayesian logistic regression model for active relevance feedback. In: SIGIR '08, pp 227–234.

Xu, Z., Akella, R., & Zhang, Y. (2007). Incorporating diversity and density in active learning for relevance feedback. In: ECIR'07, pp 246–257.

Xu, Z., Kersting, K., & Joachims, T. (2010). Fast active exploration for link-based preference learning using gaussian processes. In: ECML PKDD'10, pp 499–514.

Yue, Y., & Joachims, T. (2009). Interactively optimizing information retrieval systems as a dueling bandits problem. In: ICML'09, pp 1201–1208.

Yue, Y., Broder, J., Kleinberg, R., & Joachims, T. (2009). The k-armed dueling bandits problem. In: COLT'09.

Zhang, T. (2004). Solving large scale linear prediction problems using stochastic gradient descent algorithms. In: ICML '04, ACM, pp 116+.

Zhang, Y., Xu, W., & Callan, J. (2003). Exploration and exploitation in adaptive filtering based on bayesian active learning. In: ICML '03, pp 896–904.