Fast and reliable online learning to rank for information retrieval

Hofmann, K.

Publication date
2013

6

Balancing Exploration and Exploitation

In the previous two chapters we developed and investigated interleaved comparison methods as a promising solution for inferring information about rankers from implicit user feedback. In this and the next chapter, we focus on the question of how to learn reliably and efficiently from the inferred feedback.[1]

Methods for online learning to rank for IR need to address a number of challenges. First, the most robust methods for inferring feedback provide only relative information, e.g., about the relative quality of documents (§2.3) or rankers (see the interleaved comparison approaches discussed in previous chapters). Algorithms for learning from such relative feedback have been proposed (§2.5), and these form our baseline algorithms. Second, even relative feedback can be noisy and biased. Our empirical results in this chapter provide first insights into how algorithms for learning to rank from pairwise and listwise relative feedback perform under noise.

A challenge in online learning to rank for IR that has not been addressed previously is that algorithms for this setting need to take into account the effect of learning on 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 chapter, we frame this fundamental trade-off as an *exploration–exploitation*

[1] This chapter is based on work presented in Hofmann et al. (2011a, 2013b). However, the empirical results presented here differ from our previous work as follows. First, the *navigational* click model is instantiated differently (as shown in §3.3) to match the model used in Chapter 4. Second, we previously applied our algorithms to the documents provided by the LETOR data sets in their original order. Because this order is non-randomized, learners started from high-quality lists, which could result in higher absolute performance. In the experiments presented here, we address this problem by breaking ties randomly (so that a result list generated from a zero weight vector is completely randomized). Despite these changes, the results are qualitatively identical to those presented earlier and support the same conclusions.

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 focuses too much on document lists from which it can gain a lot of new information, it risks presenting bad results to the user during learning. Therefore, to perform optimally, the system must *explore* new solutions, while also maintaining satisfactory performance by *exploiting* existing solutions. 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 pairwise and listwise methods, the two most successful approaches for learning to rank in IR (§2.2). 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. However, as we demonstrate in this chapter, balancing exploration and exploitation is crucial to achieving good performance.

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). The existing algorithm learns directly from the relative feedback that can be obtained from interleaved comparison methods, but we show that it over-explores without a suitable balance of exploration and exploitation.

We present the first two algorithms that can balance exploration and exploitation in settings where only relative feedback is available. First, we start from a pairwise approach that is initially purely exploitative (§2.5.1). Second, we start from a recently developed listwise algorithm that is initially purely exploratory (Yue and Joachims, 2009) (§2.5.2). We assess the resulting algorithms using the evaluation framework described in Chapter 3 to answer the following questions:

RQ 11 Can balancing exploration and exploitation improve online performance in online learning to rank for IR?

RQ 12 How are exploration and exploitation affected by noise in user feedback?

RQ 13 How does the online performance of different types (pairwise and listwise) of online learning to rank for IR approaches relate to balancing exploration and exploitation?

Our main result is that finding a proper balance between exploration and exploitation can substantially and significantly improve the online retrieval performance in pairwise and listwise online learning to rank for IR. In addition, our results are the first to shed light on the strengths and weaknesses of pairwise and listwise learning in an online setting, as these types of approaches have previously only been compared offline. We find that learning from document-pairwise feedback can be effective when this feedback is reliable. However, when feedback is noisy, a high amount of exploration is required to obtain reasonable performance. When clicks are interpreted as listwise feedback, learning is similarly effective as under the pairwise interpretation, but it is much more robust to noise. However, online performance under the original listwise learning approach is suboptimal, as it over-explores. Dramatically reducing exploration allows learning rankers equally well, but at much lower cost. Consequently, balancing exploration and

exploitation in the listwise setting results in significantly improved online performance under all levels of noise. 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 data sets. Finally, we describe the implications of our results for making these approaches work effectively in practice.

The remainder of this chapter is organized as follows. We present our methods for balancing exploration and exploitation in the pairwise and listwise setting in §6.1. Our experiments are described in §6.2, followed by results and analysis in §6.3. We conclude in §6.4.

6.1 Approaches

In this section, we describe our approaches for balancing exploration and exploitation for learning to rank in IR. These build on the pairwise and listwise baseline learning algorithms shown in §2.5. Our approaches are based on the problem formulation of online learning to rank for IR as a contextual bandit problem, as shown in §3.1.

6.1.1 Balancing Exploration and Exploitation in Pairwise Learning to Rank

Our first approach builds off a pairwise formulation of learning to rank (Herbrich et al., 1999; Joachims, 2002), in particular the stochastic gradient descent algorithm presented in Sculley (2009). As detailed in §2.5.1, this algorithm optimizes a weight vector $\mathbf{w}$ for linear combinations of ranking features $\mathbf{x} = \phi(d, q)$ to minimize a loss function formulated in terms of the pairwise ranking loss (see Algorithm 4 on page 29). As shown by Joachims (2002), the required pairwise feedback can be inferred from implicit feedback, such as click data.

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 data set. Sculley (2009) developed a sampling scheme that allows the training of a stochastic gradient descent learner on a random subset of the data without a noticeable loss in performance. Document pairs are sampled randomly such that at each learning step one relevant and one non-relevant document were selected to form a training pair. In the online setting, we expect such a fully exploratory strategy to result in minimal training bias and best long-term learning.

Algorithm 10 Balancing exploration and exploitation in the pairwise setting.

1: **Input**: $\mathcal{D}$, η, λ, $\mathbf{w}_0$, ϵ
2: **for** query q_t ($t = 1..\infty$) **do**
3: $\quad\mathbf{X} = \phi(\mathcal{D}|q_t)$ *// extract features*
$\quad$*// generate exploitative result list*
4: $\quad S = \mathbf{w}_{t-1}^T\mathbf{X}$
5: $\quad\mathbf{l}_1 = sort\_descending\_by\_score(\mathcal{D}, \mathbf{s})[1:10]$
6: $\quad\mathbf{l}[r] \leftarrow$ first element of $\mathbf{l}_1 \notin \mathbf{l}$ with probability ϵ; element randomly sampled without replacement from $\mathcal{D} \setminus \mathbf{l}$ with probability $1-\epsilon$
7: $\quad$Display $\mathbf{l}$ and observe clicked elements $\mathbf{c}$.
8: $\quad$Construct all labeled pairs $\mathcal{P} = (\mathbf{x}_1, \mathbf{x}_2, y)$ for q_t from $\mathbf{l}$ and $\mathbf{c}$.
9: $\quad$**for** $(\mathbf{x}_1, \mathbf{x}_2, y)$ in $\mathcal{P}$ **do**
10: $\quad\quad$**if** $y\mathbf{w}_{t-1}^T(\mathbf{x}_1 - \mathbf{x}_2) < 1.0$ and $y \neq 0.0$ **then**
11: $\quad\quad\quad\mathbf{w}_t = \mathbf{w}_{t-1} + \eta y(\mathbf{x}_1 - \mathbf{x}_2) - \eta\lambda\mathbf{w}_{t-1}$ *// update* $\mathbf{w}_t$

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.

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.[2]

We employ a method for balancing exploration and exploitation that is inspired by ϵ-*greedy*, a commonly used exploration strategy in RL (§2.4.2).[3] The difference between our approach and ϵ-greedy is that we do not pick a single action at each timestep, but rather select a number of actions that are presented simultaneously. This results in Algorithm 10, which differs from our baseline algorithm in how the result list is constructed (line 6).

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

[2]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.

[3]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, 2006a).

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 ($\epsilon = 0$ corresponds to the purely exploitative baseline setting). Values close to 1 mean more exploration.

6.1.2 Balancing Exploration and Exploitation in Listwise Learning to Rank

Our second online learning to rank approach builds off 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). An overview of DBGD is given in §2.5.2.

Given an appropriate function for comparing document lists, DBGD 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 $f(\mathbf{l}_1, \mathbf{l}_2)$ that does allow balancing exploration and exploitation.

We base our comparison method $f(\mathbf{l}_1, \mathbf{l}_2)$ on BI (Joachims, 2003; Radlinski et al., 2008b), as detailed in §2.3.1 (in particular, Algorithm 1 on page 20). Extending this algorithm to balance exploration and exploitation is easiest compared to other interleaved comparison methods, and this is sufficient to test our hypothesis that balancing exploration and exploitation in online learning to rank for IR can improve online performance. Algorithms for balancing exploration and exploitation based on other interleaved comparison methods are possible and will be investigated in the future. A related approach, which is based on PI-MA and PI-MA-IS (Chapter 4) and improves online performance by reusing previously collected interaction data for more effective exploration, is presented in Chapter 7.

In contrast to previous work, we alter BI to randomize not only the starting list and then interleaving documents deterministically, but instead 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. We call this altered version of BI *stochastic BI*.

Constructing result lists using stochastic BI allows us to apply a method similar to ϵ-greedy. The resulting algorithm, which supplies the comparison method that is required by DBGD, is shown in Algorithm 11. The algorithm takes as input two document lists $\mathbf{l}_1$ and $\mathbf{l}_2$, 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 $\mathbf{c}$ are observed. Then, for each clicked document, a click is attributed to list $\mathbf{l}_i$ ($i \in \{1, 2\}$) if the document is in the top v of $\mathbf{l}_i$, where v is the lowest-ranked click (as in Algorithm 1).

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

Algorithm 11 $f(\mathbf{l}_1, \mathbf{l}_2)$ – k-greedy comparison of document lists using stochastic BI.

Input: $\mathbf{l}_1$, $\mathbf{l}_2$, k
$\mathbf{l} = []$, $n_1 = 0$; $n_2 = 0$
while $(len(\mathbf{l}) < len(\mathbf{l}_1)) \wedge (len(\mathbf{l}) < len(\mathbf{l}_2))$ **do**
  $a \leftarrow 1$ with probability k **else** 2
  $j = min\{i : \mathbf{l}_a[i] \notin \mathbf{l}\}$
  $append(\mathbf{l}, \mathbf{l}_a[j])$
  $n_a = n_a + 1$
// present $\mathbf{l}$ *to user and observe clicks* $\mathbf{c}$*, then infer outcome (if at least one click was observed)*
$d_{max} =$ lowest-ranked clicked document in $\mathbf{l}$
$v = min\{j : (d_{max} = \mathbf{l}_1[j]) \vee (d_{max} = \mathbf{l}_2[j])\}$
$c_1 = len\{i : c[i] = true \wedge \mathbf{l}[i] \in \mathbf{l}_1[1..v]\}$
$c_2 = len\{i : c[i] = true \wedge \mathbf{l}[i] \in \mathbf{l}_2[1..v]\}$
// compensate for bias (Eq. 6.1)
$c_2 = \frac{n_1}{n_2} * c_2$
return -1 **if** $c_1 > c_2$ **else** 1 **if** $c_1 < c_2$ **else** 0

equal number of documents are presented to the user in expectation.[4] As k decreases, more documents are contributed by the exploitative list, which is expected to improve the quality of the result list but produce noisier feedback.

As k decreases, more documents from the exploitative list are presented, which introduces bias for inferring feedback. The bias linearly increases the expected number of clicks on the exploitative list and reduces the expected number of clicks on the exploratory list. We can partially compensate for this bias since

$$E[c_2] = \frac{n_1}{n_2} * E[c_1], \tag{6.1}$$

where $E[c_i]$ is the expected number of clicks within the top v of list $\mathbf{l}_i$, and n_i is the number of documents that $\mathbf{l}_i$ contributed to 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. More complex, unbiased sampling schemes can be developed using PI-MA and PI-MA-IS (Chapter 4), but this is beyond the scope of this thesis.

[4] Note that the setting $k = 0.5$ corresponds to the fully exploratory baseline algorithm. Setting $k > 0.5$ would typically 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.

6.2 Experiments

In this section, we describe the experiments that evaluate the algorithms presented in §6.1. All experiments use the experimental setup detailed in Chapter 3. Here, we provide further details and give an overview of the experimental runs we evaluate in the pairwise and listwise setting.

For the experiments in this chapter we use the 9 LETOR 3.0 and 4.0 data sets (§3.4). Click data is generated using the *perfect*, *navigational*, *informational*, and *almost random* click models as shown in Table 3.1.[5] As detailed in §3.5, we use the training folds of each data set for training during the learning cycle and for calculating online performance (in terms of discounted cumulative NDCG, with $\gamma = 0.995$). We use the test sets for measuring final performance (in terms of NDCG).

For each data set 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 ($\epsilon = 0.0$), purely exploratory for the listwise approach ($k = 0.5$)) using a two-sided student's t-test (§3.5).

6.2.1 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 §6.1.1, in the fully exploitative setting ($\epsilon = 0$; equivalent to Algorithm 4). Against this baseline we compare increasingly exploratory versions of the algorithm ($\epsilon \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$). All experiments are run for 1,000 iterations.

6.2.2 Listwise Approach

In all listwise experiments, we initialize the starting weight vector $\mathbf{w}_0$ to zero. We 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 data sets in our preliminary experiments). Our *baseline* is Algorithm 5, based on (Yue and Joachims, 2009), which corresponds to a purely exploratory setting of $k = 0.5$ in our extended method.[6] Against this baseline we compare *exploit* runs that balance exploration and exploitation by varying the exploration rate k between 0.4 and 0.1 as shown in Algorithm 11. Again, we run all experiments for 1,000 iterations.

6.3 Results and Discussion

In this section, we present the results of our experiments, designed to test our main hypothesis — that balancing exploration and exploitation can improve the online perfor-

[5] Results for the *almost random* click model are omitted, as they did not result in any new insights beyond those obtained from the other three click models.

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

mance of online learning to rank for IR systems. We first address this hypothesis for the pairwise learning algorithm (§6.3.1), and then for the listwise learning algorithm (§6.3.2). For both approaches we further analyze online and offline performance to identify factors that affect online performance. Finally, we compare the two approaches under the novel perspective of balancing exploration and exploitation (§6.3.3).

6.3.1 Pairwise Learning

We present our results for the experiments on the pairwise approach, described in §6.2.1, in Table 6.1. It shows the online performance of the baseline approach (*exploitative*, $\epsilon = 0$) and increasingly more exploratory runs ($\epsilon > 0.0$) for the 9 LETOR 3.0 and 4.0 data sets and the *perfect*, *navigational*, and *informational* click models.

We expect good online performance for the exploitative baseline if the algorithm can learn well despite any bias introduced due to the high level of exploitation. Generally, an online learning to rank approach should exploit as much as possible, as it ensures that users see the best possible result lists given what has been learned. However, if increased exploration results in sufficiently high gains in offline performance, its short-term cost may be outweighed by its long-term benefits, as it increases the quality of result lists later on.

For the *perfect* click model, the best online performance is achieved in the baseline setting for four out of nine data sets, ranging from 87.38 (*NP2004*, row 4) to 108.06 (*HP2003*, row 1). For these data sets, the exploitative baseline algorithm appears to learn well enough, so that additional exploration does not lead to high gains in offline performance that would outweigh its cost. For the three data sets *TD2003* (row 5), *TD-2004* (row 6), and *MQ2008* (row 9), online performance is higher at $\epsilon = 0.2$ than in the baseline setting, but the difference is not statistically significant. For the remaining data sets, we see statistically significant improvements over the baseline at $\epsilon = 0.4$. Online performance improves by 58% for the data set *OHSUMED*, and by 6% for the data set *MQ2007* (rows 7–8).

In the *navigational* click model, optimal online performance is achieved at higher exploration rates than for the *perfect* click model. For five data sets, the best setting is $\epsilon = 0.2$, and for four data sets it is $\epsilon = 0.4$. For all but one data set (*MQ2008*, row 18) the improvements in online performance over the baseline are statistically significant. Under noisier feedback, learning becomes more difficult, meaning that the quality of the learned weight vectors that can be exploited is lower than under perfect feedback. This reduces the benefit of exploitation, and lowers the cost of exploration, increasing the relative benefit of exploration. The biggest performance gains under increased exploration are observed for *NP2004* (row 13) and *TD2003* (row 14), where the online performance obtained in the best exploratory setting is 2.5 and 1.2 times that of the baseline setting. High performance gains are also observed for *HP2003* (81% improvement over the baseline, row 10), *HP2004* (98% improvement, row 11), *NP2003* (72.8%, row 12), and *TD2004* (81.97%, row 15). The improvement for *OHSUMED* is 51% (row 16). Small improvements are observed for *MQ2007* (6%, row 17) and *MQ2008* (1%, not statistically significant, row 18).

Compared to the *perfect* click model, online performance with the *navigational* model is much lower, as expected. The performance loss due to noise is between 2.7% (*NP-*

	ϵ	**0.0**	**0.2**	**0.4**	**0.6**	**0.8**	**1.0**
				perfect click model			
1	*HP2003*	**108.06**	99.96$^{\triangledown}$	87.58$^{\blacktriangledown}$	76.04$^{\blacktriangledown}$	50.75$^{\blacktriangledown}$	1.00$^{\blacktriangledown}$
2	*HP2004*	**101.75**	84.38$^{\blacktriangledown}$	73.88$^{\blacktriangledown}$	60.47$^{\blacktriangledown}$	42.22$^{\blacktriangledown}$	0.89$^{\blacktriangledown}$
3	*NP2003*	**104.51**	98.67	89.72$^{\blacktriangledown}$	72.00$^{\blacktriangledown}$	46.31$^{\blacktriangledown}$	1.57$^{\blacktriangledown}$
4	*NP2004*	**87.38**	84.33	75.76$^{\blacktriangledown}$	65.25$^{\blacktriangledown}$	44.62$^{\blacktriangledown}$	0.98$^{\blacktriangledown}$
5	*TD2003*	50.18	**50.54**	39.69$^{\blacktriangledown}$	28.55$^{\blacktriangledown}$	16.15$^{\blacktriangledown}$	1.94$^{\blacktriangledown}$
6	*TD2004*	47.49	**48.82**	34.00$^{\blacktriangledown}$	23.77$^{\blacktriangledown}$	13.57$^{\blacktriangledown}$	3.29$^{\blacktriangledown}$
7	*OHSUMED*	49.31	**78.07**$^{\blacktriangle}$	69.94$^{\blacktriangle}$	60.51$^{\blacktriangle}$	49.94	37.76$^{\blacktriangledown}$
8	*MQ2007*	64.59	67.56$^{\blacktriangle}$	**68.35**$^{\blacktriangle}$	63.99	57.88$^{\blacktriangledown}$	51.14$^{\blacktriangledown}$
9	*MQ2008*	89.20	**89.67**	85.74$^{\blacktriangledown}$	80.58$^{\blacktriangledown}$	73.78$^{\blacktriangledown}$	66.70$^{\blacktriangledown}$
				navigational click model			
10	*HP2003*	50.01	**90.34**$^{\blacktriangle}$	88.31$^{\blacktriangle}$	80.38$^{\blacktriangle}$	51.38	0.99$^{\blacktriangledown}$
11	*HP2004*	41.80	**82.76**$^{\blacktriangle}$	76.73$^{\blacktriangle}$	65.72$^{\blacktriangle}$	46.23	0.92$^{\blacktriangledown}$
12	*NP2003*	49.21	78.67$^{\blacktriangle}$	**85.03**$^{\blacktriangle}$	74.73$^{\blacktriangle}$	49.90	1.68$^{\blacktriangledown}$
13	*NP2004*	24.75	73.31$^{\blacktriangle}$	**86.53**$^{\blacktriangle}$	75.96$^{\blacktriangle}$	53.16$^{\blacktriangle}$	0.89$^{\blacktriangledown}$
14	*TD2003*	16.65	**36.53**$^{\blacktriangle}$	34.26$^{\blacktriangle}$	25.99$^{\blacktriangle}$	15.37	2.01$^{\blacktriangledown}$
15	*TD2004*	22.07	**40.16**$^{\blacktriangle}$	32.05$^{\blacktriangle}$	22.85	13.31$^{\blacktriangledown}$	3.30$^{\blacktriangledown}$
16	*OHSUMED*	46.16	**69.63**$^{\blacktriangle}$	66.28$^{\blacktriangle}$	58.58$^{\blacktriangle}$	48.77$^{\blacktriangle}$	37.83$^{\blacktriangledown}$
17	*MQ2007*	58.66	60.74$^{\blacktriangle}$	**62.08**$^{\blacktriangle}$	60.39$^{\blacktriangle}$	56.68$^{\blacktriangledown}$	51.21$^{\blacktriangledown}$
18	*MQ2008*	79.53	79.60	**80.38**	77.70$^{\triangledown}$	72.85$^{\blacktriangledown}$	66.23$^{\blacktriangledown}$
				informational click model			
19	*HP2003*	4.26	12.47$^{\blacktriangle}$	38.36$^{\blacktriangle}$	**46.37**$^{\blacktriangle}$	39.11$^{\blacktriangle}$	0.97$^{\blacktriangledown}$
20	*HP2004*	2.54	16.01$^{\blacktriangle}$	30.98$^{\blacktriangle}$	**39.85**$^{\blacktriangle}$	28.09$^{\blacktriangle}$	0.93$^{\blacktriangledown}$
21	*NP2003*	3.87	9.44$^{\blacktriangle}$	25.48$^{\blacktriangle}$	**41.97**$^{\blacktriangle}$	38.29$^{\blacktriangle}$	1.60$^{\blacktriangledown}$
22	*NP2004*	2.28	10.97$^{\blacktriangle}$	31.76$^{\blacktriangle}$	**49.12**$^{\blacktriangle}$	37.71$^{\blacktriangle}$	0.95$^{\blacktriangledown}$
23	*TD2003*	1.66	7.28$^{\blacktriangle}$	14.17$^{\blacktriangle}$	**16.03**$^{\blacktriangle}$	10.62$^{\blacktriangle}$	1.96$^{\triangle}$
24	*TD2004*	4.71	14.09$^{\blacktriangle}$	**20.03**$^{\blacktriangle}$	17.45$^{\blacktriangle}$	10.85$^{\blacktriangle}$	3.25$^{\blacktriangledown}$
25	*OHSUMED*	36.77	49.75$^{\blacktriangle}$	**59.85**$^{\blacktriangle}$	55.79$^{\blacktriangle}$	48.00$^{\blacktriangle}$	37.81
26	*MQ2007*	55.02	56.33$^{\triangle}$	56.42$^{\blacktriangle}$	**56.87**$^{\blacktriangle}$	55.06	51.14$^{\blacktriangledown}$
27	*MQ2008*	**72.68**	72.22	72.36	72.15	70.85$^{\blacktriangledown}$	66.33$^{\blacktriangledown}$

Table 6.1: Results for the pairwise approach. Online performance (in terms of cumulative NDCG) over 1,000 iterations for the exploitative baseline $\epsilon = 0$ and increasingly exploratory runs ($\epsilon > 0$).

2004) and 27.7% (*TD2003*), when comparing the best settings for each click model and data set.

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 makes learning more difficult. Compared to the *navigational* click model, online performance drops by another 8% (*MQ2007*) to 51% (*HP2004*).

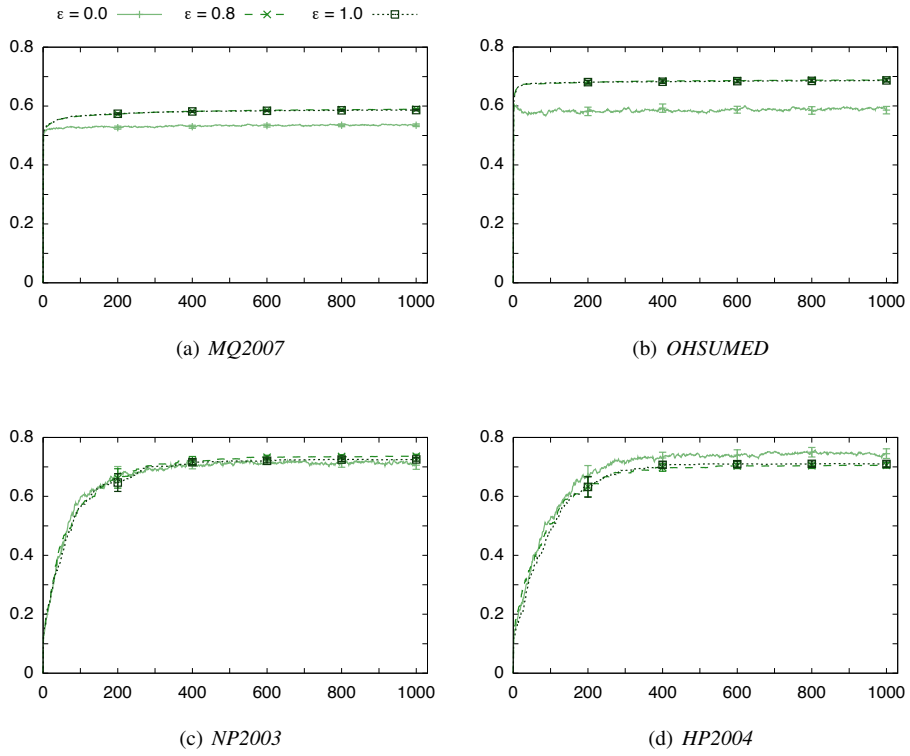
Under this click model, the cost of exploration further decreases relative to its benefit, so optimal performance is again seen at higher exploration rates. For six data sets, the best online performance is achieved at $\epsilon = 0.6$; for two data sets the best setting is $\epsilon = 0.4$. All improvements are statistically significant when compared to the purely exploitative baseline. For the *HP*, *NP*, and *TD* data sets, online performance improves by as much as an order of magnitude (rows 19–24). For the remaining data sets, improvements are lower, at 63% (*OHSUMED*, row 25) and 3% (*MQ2007*, row 26). An exception is *MQ2008*, for which there are no significant differences in online performance for runs with $\epsilon \in [0.0, 0.6]$ (row 27).

Overall, we conclude that balancing exploration and exploitation for the pairwise approach can lead to significant and substantial improvements in online performance. This balance appears to be strongly affected by noise, with highest relative improvements observed under the noisiest *informational* click model. Also, as click noise increases, the amount of exploration required for good online performance increases. Best values are $\epsilon \in [0.0, 0.2]$ for the *perfect* click model, $\epsilon \in [0.2, 0.4]$ for the *navigational* click model, and $\epsilon = [0.4, 0.6]$ for the *informational* click model. These findings confirm our hypothesis that balancing exploration and exploitation in the pairwise approach improves online performance.

Besides overall trends in online performance under different exploration rates, we find performance differences between data sets. One such difference is that for the *HP* and *NP* data sets online performance tends to be higher than the remaining data sets, especially under the *perfect* and *navigational* click models. This suggests that these data sets are easier, i.e., that click feedback can be used effectively to learn linear weight vectors that generalize well. We can confirm this analysis by comparing the offline performance that the pairwise approach achieves on these data sets. For the *perfect* click model, an overview is included in Table 6.3 (on page 109). Indeed, offline NDCG@10 ranges from 0.704 (*HP2004*) to 0.760 (*HP2003*) for the "easy" data sets, and is substantially lower for the more difficult data sets (from 0.272 for *TD2003* to 486 for *MQ2008*). While our NDCG scores are not directly comparable with those reported by Liu (2009) (only NDCG@1 scores are equivalent, cf., §2.1), they show the same trend in terms of relative difficulty.

Under the *perfect* click model we found differences between most data sets and *OHSUMED* and *MQ2007*. For these two data sets, online performance increased significantly at $\epsilon = 0.2$, while for the remaining data sets, no significant improvements over the purely exploitative baseline were observed. The significant improvements at an increased exploration rate suggest that either big learning gains were realized for these data sets with increased exploration (outweighing the cost of exploraiton), or that exploration for these data sets is relatively low. As we detail below, both effects play a role here.

To analyze performance differences between the data sets, we study the learning curves of these data sets at different levels of exploration. Figure 6.1 shows the offline performance in terms of NDCG (on the whole result list) plotted over time (up to $1{,}000$ iterations) for the data sets *MQ2007*, *OHSUMED*, *NP2003*, and *HP2004*. For the data sets *MQ2007* (Figure 6.1(a)) and *OHSUMED* (Figure 6.1(b)) we see that the best offline performance is achieved at high exploration rates (the dark, dashed and dotted lines; the difference between settings $\epsilon = 0.8$ and $\epsilon = 1.0$ is negligible). For *MQ2007*, offline performance at $\epsilon = 1.0$ is 0.533, 3% higher than in the baseline setting. The biggest



(a) *MQ2007* (b) *OHSUMED*

(c) *NP2003* (d) *HP2004*

Figure 6.1: Final performance for the pairwise approach over time for the data sets *MQ2007*, *OHSUMED*, *NP2003*, and *HP2004*, under the *perfect* click model and $\epsilon \in \{0.0, 0.8, 1.0\}$.

difference between the final performance of the exploitative baseline and higher levels of exploration under the *perfect* click model is observed for the data set *OHSUMED* (Figure 6.1(b), offline performance is 0.657 for $\epsilon = 1.0$, 9% higher than in the baseline setting). For this data set, the pairwise algorithm learns very poorly without at least some exploration. Not shown is the learning curve for *MQ2008*. It follows the same trend, with a final difference in offline performance of 4% (offline performance is 0.497 when $\epsilon = 1.0$).

Different behavior is observed for the remaining data sets. For the data sets *NP2003* (Figure 6.1(c)), *HP2003*, *TD2003*, and *TD2004* (not shown) there is no significant difference in offline performance between less and more exploratory settings under perfect feedback. This is contrary to the expected behavior that the highest level of exploration should result in best learning, as pure exploration corresponds to randomly sampling document pairs for preference detection. Most likely, this unexpected behavior under the *perfect* click model 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. The data set for which this effect

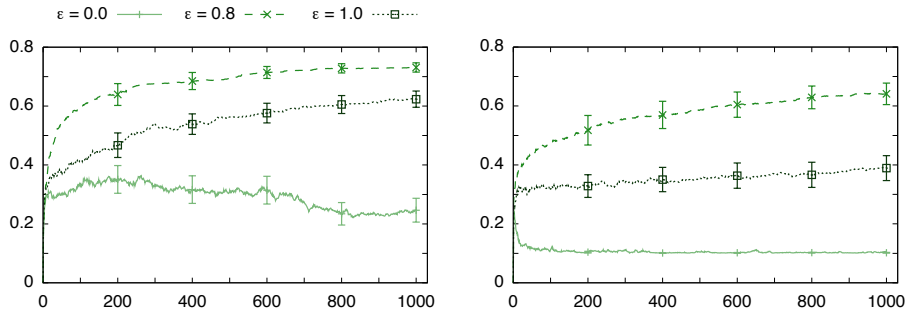
is observed has only few relevant documents, so that focusing feedback on a promising region can have a substantial benefit. The strongest effect is seen for data set *HP2004* (Figure 6.1(d)), where offline performance improves when implicit feedback is collected on exploitative result lists ($\epsilon = 0$, light and solid line) as opposed to more exploratory settings.

Besides the gains in offline performance realized under the *perfect* click model for *OHSUMED* and *MQ2007* under increased exploration rates, we can also confirm the relatively low risk of exploration for these data sets. In Table 6.1 we see that under pure exploration, the drop in online performance for these two data sets and *MQ2008* is much smaller than for the remaining data sets. For example, online performance for *MQ2008* at $\epsilon = 1.0$ is 66.7 (row 9), which corresponds to an NDCG of 0.3–0.4 for the average result list presented to the user during learning. In contrast, online performance at this level of exploration is 0.89 for data set *HP2004* (row 2), which corresponds to an average NDCG of less than 0.005. These differences are a result of the number of candidate documents per query, and the relative ratio of relevant to non-relevant documents provided per query. As described in §3.4, *OHSUMED* and the *MQ* data sets have much fewer candidate documents per query (approximately a factor of 10, compared to the other data sets), and a much higher ratio of relevant to non-relevant documents. Under these conditions, randomizing candidate documents has a much smaller negative effect on online performance than for data sets with many (non-relevant) candidate documents. This results in the low cost of exploration observed for these data sets. For the *HP*, *NP*, and *TD* data sets, the low ratio of relevant to non-relevant documents results in a much higher cost of exploration.

While the low number of candidate documents for *OHSUMED* and the *MQ* data sets results in a low cost of exploration, they also reduce its benefit. Comparing the learning curves in Figure 6.1(a)–6.1(b) to those in Figure 6.1(c)–6.1(d), we see that a much smaller gain in offline performance is realized (the increase in offline performance over, e.g., the first 100 iterations is much smaller). Thus, for data sets with a high ratio of relevant documents, exploration is cheap, but its benefit is limited. An extreme case is *MQ2008*, where the benefit of improving offline performance through increased exploration is so small that it does not lead to significant improvements in online performance (rows 9, 18, and 27 in Table 6.1). More generally, we find that the balance of exploration and exploitation is affected by the magnitude of the learning gains (in terms of offline performance) that can be realized under increased exploration, and the cost of the exploration.

For all data sets, the absolute difference in final performance at varying exploration rates is relatively small under the *perfect* click model. Much higher variance is observed when we simulate noisy feedback. Figure 6.2 shows learning curves for the data set *NP2003* at different settings of ϵ for the *navigational* and *informational* click models. For the *navigational* click model (Figure 6.2(a)) final performance improves over time for all $\epsilon > 0.0$.

For the *informational* click model, final performance degrades dramatically in the purely exploitative baseline settings ($\epsilon = 0$, 0.102). In this setting, performance decreases over time. The purely exploratory setting ($\epsilon = 1.0$) leads to reasonable final performance, while the best performance is achieved with high exploration and some exploitation ($\epsilon = 0.8$, 0.724). This finding also confirms our earlier observation that



(a) *navigational* (b) *informational*

Figure 6.2: Final performance for the pairwise approach over time for the data set *NP-2003* for *navigational*, and *informational* click models and $\epsilon \in \{0.0, 0.8, 1.0\}$.

best offline performance is not always achieved in the most exploratory setting (possibly because feedback under increased exploitation focuses on promising documents).

Our analysis of offline 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 data set. 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. For data sets with a low ratio of relevant to non-relevant documents, the low cost of exploration resulted in significant gains in online performance under reliable feedback. Under noisy feedback, higher exploration rates generally improved learning, though the best performance occurred with moderate amounts of exploitation. Overall, our results confirmed that balancing exploration and exploitation can significantly and substantially improve online performance in pairwise online learning to rank for IR.

6.3.2 Listwise Learning

Our main results for the listwise approach are shown in Table 6.2. The experiments described in §6.2.2 measure online performance of the exploratory baseline approach ($k = 0.5$) and increasingly exploitative ($k < 0.5$) experimental runs on the 9 LETOR 3.0 and 4.0 data sets on the *perfect*, *navigational*, and *informational* click models.

In the listwise setting, we expect best learning (in terms of offline performance) for the exploratory baseline approach. However, the online performance of the baseline approach is expected to be low, as it does not sufficiently exploit what has been learned. We hypothesize that increasing exploitation can improve online performance as long as its benefits outweigh the resulting loss in offline performance.

For the *perfect* click model, all data sets except *MQ2008* (row 9) improve over the

k		0.5	0.4	0.3	0.2.	0.1
			perfect click model			
1	*HP2003*	102.89	113.60▲	116.82▲	**122.38**▲	122.36▲
2	*HP2004*	95.81	103.38▲	108.87▲	**112.76**▲	109.71▲
3	*NP2003*	95.41	101.24▲	107.35▲	**110.24**▲	108.66▲
4	*NP2004*	99.67	108.41▲	114.83▲	**118.01**▲	117.87▲
5	*TD2003*	38.97	41.19△	43.86▲	**44.59**▲	42.72▲
6	*TD2004*	35.32	37.99▲	39.75▲	**42.01**▲	40.49▲
7	*OHSUMED*	69.03	71.78▲	74.47▲	**75.08**▲	75.02▲
8	*MQ2007*	59.66	61.50▲	61.81▲	**61.86**▲	61.86▲
9	*MQ2008*	77.90	78.05	**79.17**	78.98	77.86
			navigational click model			
10	*HP2003*	84.07	98.98▲	103.77▲	**108.43**▲	106.28▲
11	*HP2004*	73.83	85.14▲	88.74▲	91.22▲	**95.74**▲
12	*NP2003*	76.23	87.38▲	92.50▲	**96.94**▲	93.71▲
13	*NP2004*	83.75	95.93▲	97.89▲	106.28▲	**107.36**▲
14	*TD2003*	31.41	34.04△	35.39▲	37.26▲	**37.61**▲
15	*TD2004*	30.72	33.17▲	**34.62**▲	33.29▲	33.18△
16	*OHSUMED*	67.06	69.13▲	70.45▲	**71.72**▲	70.47▲
17	*MQ2007*	56.46	57.20	58.30△	**58.63**▲	57.73
18	*MQ2008*	74.84	74.70	**76.79**△	76.04	76.01
			informational click model			
19	*HP2003*	49.82	60.39▲	65.60▲	71.91▲	**75.68**▲
20	*HP2004*	44.76	48.39	55.69▲	**61.14**▲	60.41▲
21	*NP2003*	47.72	58.31▲	64.14▲	66.42▲	**77.17**▲
22	*NP2004*	48.64	63.44▲	66.43▲	**79.94**▲	78.74▲
23	*TD2003*	21.81	22.67	24.73△	**26.53**▲	25.83▲
24	*TD2004*	22.02	22.68	**24.50**▲	21.36	21.99
25	*OHSUMED*	62.83	63.47	**65.17**	63.81	61.02
26	*MQ2007*	54.89	54.79	**55.45**	54.66	55.12
27	*MQ2008*	71.38	72.43	72.77	71.93	**73.17**

Table 6.2: Results for the listwise approach. Online performance over 1,000 iterations for *baseline* ($k = 0.5$) and *exploit* ($k \in [0.1, 0.4]$) runs.

purely exploratory baseline for all settings of $k < 0.5$. For all these data sets, the best online performance is obtained at a relatively low setting of $k = 0.2$. Increases in online performance over the baseline range from 4% (*MQ2007*, row 8) to 19% (*HP2003*, row 1, and *TD2004*, row 6). The data set *MQ2008* is an exception. Although online performance is highest for $k = 0.3$, none of the exploitative settings perform significantly differently from the exploratory baseline. As discussed in the previous chapter, this data set has fewer candidate documents than other data sets, leading to a relatively low benefit of

increased exploitation.

Results for the *navigational* click model are similar. For all data sets, online performance is significantly higher under higher exploitation than in the baseline setting. Best performance is achieved for $k \in [0.1, 0.3]$. For two data sets, $k = 0.3$ performs best. For four of the remaining data sets, best performance is achieved at $k = 0.2$, and for three data sets, best online performance is achieved at $k = 0.1$. Improvements of the best setting of k over the baseline range from 3% (*MQ2008*, row 18) to 30% (*HP2004*, row 11). As expected, performance under the *navigational* click model is lower than under *perfect* feedback.

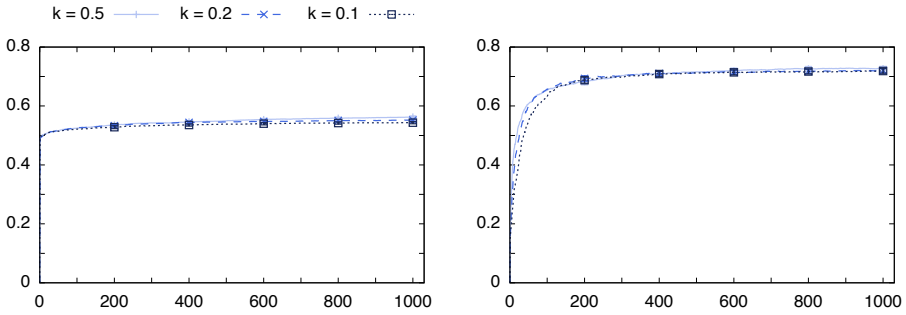
The trend continues for the *informational* click model. Again, more exploitative settings of k outperform the purely exploratory baseline in all cases. For six out of nine cases, the improvements are statistically significant. These improvements range from 11% (*TD2004*, row 24) to 64% for the data set *NP2004* (row 22). For the remaining three data sets, no statistically significant differences between baseline and exploitative runs are observed, but small increases over the exploratory baseline are observed at smaller k.

Together, these results demonstrate that, for all click models and all data sets, balancing exploration and exploitation in listwise learning to rank for IR can significantly improve online performance over the purely exploratory baseline, which confirms our hypothesis. The best overall setting for the exploration rate is $k = 0.2$. This 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. We conclude that the original listwise algorithm explores too much and surprisingly little exploration is sufficient for good performance.

Online performance is 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 data sets is more strongly affected by noisy feedback. For the *HP*, *NP*, and *TD* data sets, performance for the *informational* model drops substantially. This may again be related to the large number of non-relevant documents in these data sets. Because 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 for all levels of noise.

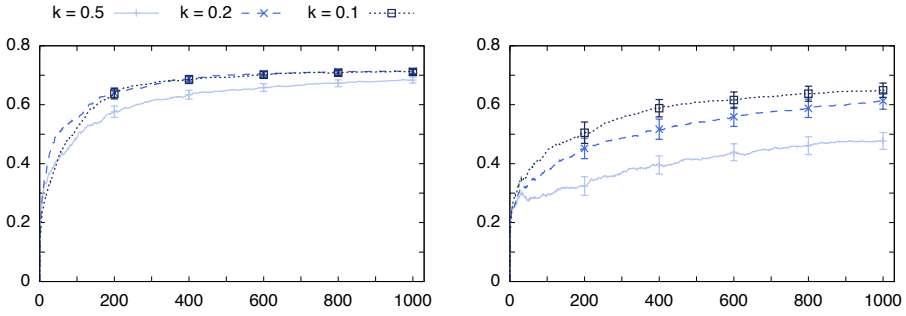
As for the pairwise approach, we analyze the relationship between online and offline performance by examining the learning curves for different levels of exploration. Figure 6.3 shows the learning curves for the data sets *MQ2007* and *NP2003* at different settings of k and the *perfect* click model. In contrast to the pairwise approach, there is no significant difference in performance after $1{,}000$ iterations. We find the same behavior for all data sets. 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 for the data set *NP2003* are shown in Figure 6.4. As expected, learning is faster when feedback is more reliable. For the idealized *perfect* click model, offline performance after $1{,}000$ iterations ranges between 0.719 ($k = 0.1$) and 0.727 ($k = 0.5$) for different settings of k. For



(a) *MQ2007* (b) *NP2003*

Figure 6.3: Offline performance (computed on the test set after each learning step) over time for the data sets *MQ2007* and *NP2003* for the *perfect* click model and $k \in \{0.1, 0.2, 0.5\}$.



(a) *navigational* (b) *informational*

Figure 6.4: Final performance (with 5% confidence intervals) over time for the data set *NP2003* for *navigational*, and *informational* click models and $k \in \{0.1, 0.2, 0.5\}$.

the noisy *informational* click model, final performance is between 0.477 ($k = 0.5$) and 0.649 ($k = 0.5$). Although final performance drops substantially as implicit feedback becomes extremely noisy, performance improves over time for all data sets as there is still a signal to learn from, 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 observed under 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 offline performance is similar to the *perfect* click model. However, in the *informational* click model, variance increases and there is a large difference between offline 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., §6.1), 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.

6.3.3 Comparing the Pairwise and Listwise Approach

For both the pairwise and the listwise approaches, our results show 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 compare the online and offline performance of both approaches. Then, we discuss how exploration impacts the performance of both approaches, and conclude with implications for putting them in practice.

Table 6.3 gives an overview of the offline performance of the pairwise and listwise approaches in their best-performing setting under *perfect* click feedback. Like the online performance, these are computed after 1,000 iterations (consisting of one 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 online learning setting with relative feedback.

	pairwise			**listwise**		
	N@1	**N@3**	**N@10**	**N@1**	**N@3**	**N@10**
HP2003	**0.687**	0.727	0.760	0.684	**0.730**	**0.761**
HP2004	0.559	0.650	0.704	**0.582**$^{\triangle}$	**0.673**$^{\triangle}$	**0.725**$^{\triangle}$
NP2003	**0.531**	0.649	**0.705**	**0.531**	**0.650**	0.704
NP2004	**0.529**	**0.658**	**0.714**	0.521	0.656	0.710
TD2003	0.247	0.271	0.272	**0.318**$^{\triangle}$	**0.300**$^{\triangle}$	**0.295**$^{\triangle}$
TD2004	0.315	0.307	0.275	**0.385**$^{\triangle}$	**0.343**$^{\triangle}$	**0.300**$^{\triangle}$
OHSUMED	**0.515**	**0.474**	**0.444**	0.510	0.470	0.441
MQ2007	**0.352**	**0.359**	**0.400**	0.329$^{\triangledown}$	0.338$^{\triangledown}$	0.381$^{\triangledown}$
MQ2008	**0.347**	**0.390**	**0.486**	0.333$^{\triangledown}$	0.376$^{\triangledown}$	0.475$^{\triangledown}$

Table 6.3: Offline performance (in terms of NDCG@N) for the pairwise ($\epsilon = 1.0$) and listwise ($k = 0.5$) online learning to rank algorithms under the *perfect* click model.

In terms of offline performance, the pairwise and listwise approaches perform similarly. The pairwise approach outperforms the listwise approach on five out of nine data sets (in terms of NDCG@10), but the performance differences are significant for only two data sets (*MQ2007* and *MQ2008*). The listwise approach outperforms the pairwise approach on four data sets, in three cases significantly (*HP2004*, *TD2003*, and *TD2004*).

We note that the performance of the listwise approach is competitive, despite the limited information available to the algorithm (relative feedback per ranker instead of per document), and the weak information about the gradient that is inferred from this feedback (based on random exploration of the gradient instead of computing a gradient, as for the pairwise approach).

We compare the online performance of the pairwise and listwise approaches by comparing Tables 6.1 (page 101) and 6.2 (page 106). Under the *perfect* click model, and in the purely exploratory baseline setting, the listwise approach performs worse than the purely exploitative pairwise approach, as expected. However, at their optimal settings, the two approaches perform similarly, with the listwise approach beating the pairwise approach on four out of the nine data sets. We conclude that, under reliable feedback, the pairwise and listwise approaches perform similarly well when used with an appropriate balance of exploration and exploitation.

When click feedback is noisy, the listwise approach performs better than the pairwise approach. Under the *navigational* click model, the listwise approach outperforms the pairwise approach in terms of online performance on six data sets. Under the *informational* click model, this number increases to seven out of the nine data sets (at the optimal levels of exploration). The reason is that the approaches react to noise differently. For the pairwise approach in its exploitative baseline setting, increases in noise lead to dramatically reduced offline performance. However, balancing exploration and exploitation allows the algorithm to recover its performance. As a result, the optimal balance between exploration and exploitation shifts towards increased exploration as feedback becomes noisier. A relatively high amount of exploration, with about half the result list constructed from exploratory documents, is needed to achieve good learning outcomes. This relatively high amount of exploration, in turn, has a negative effect on 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 when feedback is noisy. 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. While increases in exploitation introduce some amount of bias, this bias does not result in lower offline performance. Instead, it 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, a moderate level of exploitation can improve learning under the listwise approach.

Another advantage of the listwise approach is that the cost of exploration 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.

Our analysis suggests that the pairwise and listwise approaches are appropriate for learning from relative feedback in different settings. When user feedback is reliable, 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 good offline performance. It can exploit aggressively, leading to high online performance. However, when feedback is 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.

6.4 Conclusion

In this chapter, we studied the effect of balancing exploration and exploitation on 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.

Regarding the main research question addressed in this chapter, we found that balancing exploration and exploitation can substantially and significantly improve online performance in pairwise and listwise online learning to rank for IR. The effect of balancing exploration and exploitation is complex and there is an interaction between the amount of exploitation and the amount of noise in user feedback. When feedback is reliable, both pairwise and listwise 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 learn well despite a high level of exploitation. 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 effective use of implicit feedback when there is little noise, leading to high offline performance. 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, with more exploration needed as feedback becomes noisier. The offline performance of the listwise approach is similar to that of the pairwise approach under perfect feedback, but it is much more robust to noise, due to the aggregation of feedback over result lists. The listwise approach shows lower online performance than the pairwise approach in its purely exploratory baseline setting, but it performs well when exploration and exploitation are properly balanced. This first comparison of pairwise and listwise learning to rank in an online setting suggests that listwise approaches are a promising avenue of future development, because performance is competitive, robustness to noise is high, and only few approaches have been developed for the online setting (for learning with relative feedback).

The results of this chapter show that it is important to consider the effects of online learning to rank approaches on online performance. It is not sufficient to learn effectively, but by explicitly addressing online performance users can be provided with significantly better results throughout learning. We showed that balancing exploration and exploitation

is one way in which online performance can be improved.

Our approach to balancing exploration and exploitation for listwise online learning to rank were based on a simple stochastic extension of BI. Bias introduced by increased exploitation could only be compensated for approximately, and had a complex effect of online and offline performance. A similar solution can be devised for TD. Using the probabilistic interleaving methods developed in Chapter 4, comparison outcomes can be inferred from a much larger family of distributions over result lists. This opens up a range of possibilities for constructing exploratory and exploitative result lists without introducing bias. The basic mechanisms of balancing exploration and exploitation, e.g., that increased exploitation at the same offline performance increases online performance, are expected to hold under all alternative approaches. However, more complex solutions, e.g., enabled by PI-MA-IS, are expected to lead to further gains in online performance in online learning to rank for IR.

In Chapter 4, we focused on methods for inferring accurate feedback through interleaved comparisons, but did not consider the effects of the developed evaluation approaches on online performance. In the next chapter (Chapter 7), we investigate how online performance is affected by existing interleaved comparison methods and the probabilistic approach developed in the earlier chapter. In particular, we investigate how to learn quickly and reliably from noisy user feedback in an online learning to rank setting.