Investigating the effects of bootstrapping and active learning in an online reinforcement learning setting to rank for information retrieval

Mayank Kejriwal
University of Texas at Austin
kejriwal@cs.utexas.edu

Sam Blazek
University of Texas at Austin
blazeks@utexas.edu

ABSTRACT

As the development of effective and efficient information retrieval systems relies increasingly on automated techniques, ranking procedures based on machine learning methods have drawn much attention. *Online learning to rank* algorithms permit retrieval systems to learn directly from live user activity, but offer several challenges. Most importantly, the system must learn from the user without interfering with his or her search activities and decreasing user satisfaction. This requires a balanced approach to both experimentation and tuning (*exploration*) and service provision (*exploitation*). We examine recent applications of *reinforcement learning* in *listwise* and *pairwise* approaches to learning to rank, and prototype and evaluate new *active learning* and *bootstrapping* strategies in which ranked results presented to the user are tailored to gather more valuable information with less interference with users' retrieval activities. Our hypothesis is that this reduces the amount of exploration required to produce results of equivalent quality to previous reinforcement learning implementations. We empirically investigate this hypothesis using a recently developed online simulation framework.

Keywords

Information Retrieval, Reinforcement learning, active learning

1. INTRODUCTION

As large quantities of information continue to be exposed on the web, and retrieval systems get more complicated to serve the demands of a global base of users, the need for effective automation of key parts of the retrieval process is pressing. In the information retrieval community, the cost and difficulty of obtaining relevance judgments is well documented. Relevance judgment sets for large collections of documents are often incomplete, in the sense that not every document in the collection has been judged as relevant or non-relevant. The dimensions of the problem only conflate as issues of what constitutes relevance, or even the granularity of relevance, come into play.

On the other hand, implicit feedback and preferences of users can often be ascertained using click log data. This data is usually noisy, that is, there is always a possibility that the click model assumed by the system is not the one that fits the user click model. Because of accidental clicks or variance of preference among users, deducing relevance judgments of documents based on a preference function is susceptible to flawed conclusions. Despite this observation, implicit feedback still provides useful data about the preferences of the user much of the time. The need of the day, then, is to develop ranking algorithms that can satisfy the needs of such users without requiring much manual fine-tuning or supervision, both of which mandate a high-confidence relevance judgment set.

In the framework of reinforcement learning [7], such systems that are constantly getting reinforced by feedback from the user face a natural dilemma. On the one hand, the system can choose to be *exploitative* i.e. it returns results or performs actions that have been found to be the best thus far. By being exploitative, however, the system is taking the risk of not discovering actions, or navigating a space of solutions, that *might* yield higher rewards had the system known about them. In other words, by being exploitative, the system chooses to be conservative and not *explore* unknown regions of the solution space that could potentially be gold mines. *Exploration* in reinforcement learning is, thus, the antithesis of *exploitation.* A system explores, not with the goal of maximizing present reward, but with an intent to seek out solutions that will pay off in the long run with higher rewards. On the other hand, exploitation presents the solution that has the highest guaranteed payoff in the present. Finding a balance between exploration and exploitation is an important problem in reinforcement learning. In other words, the present should not always be sacrificed for a future that may never arrive, but to cling to the present set of solutions may be to risk significant sub-optimality.

Reinforcement learning is a very general formulation that is well suited to many different tasks and domains. In information retrieval, exploring the solution space amounts to presenting documents to users for which the system has no score or judgement. Exploitation in information retrieval amounts to always retrieving and displaying those documents for which the system has a high score. If the system is perfect or an oracle, then pure exploitation is obviously the strategy of choice. In practice, this is not the case. The parameters of the system overfit on the limited data available to the system. Because of this, they are not useful for judging parts of the solution space about which they have no feedback. In a recent work, Hofmann et al. showed that striking a correct balance between exploration and exploitation is the strategy of choice for maximizing long term returns [4]. For the exploration part of the system, their model was *random*. That is, the system explored by sampling documents from the entire set randomly and presenting to the user for feedback.

In a separate, earlier work, Tian and Lease showed that active learning approaches can be useful for efficiently reducing uncertainty about the solution space [9]. The benefits of active learning have also been demonstrated in other domains, and corroborated by other researchers in the information retrieval community as well [10]. The idea is to explore the unknown solution space *intelligently* by presenting only those examples to users that are expected to maximize information gain.

In this work, we propose to combine the benefits of both reinforcement learning and of active learning *within* the exploratory aspect of reinforcement learning. In other words, rather than doing random exploration, we use an active learning approach.

We also demonstrate the effects of bootstrapping an online system, and the susceptibility of the system to bootstrapping noise. By bootstrapping, we mean that we train the initial parameters of the online system by exposing a small set of documents and relevance labels rather than starting with randomly generated initial parameters. In this paradigm, we are investigating the benefits of offline learning that precedes the online phase. We test the hypothesis that even very limited bootstrapping can drastically impact the convergence properties of online algorithms in the reinforcement learning framework.

2. RELATED WORK

Although our primary contributions (subsequently detailed) are original, there is some work that forms a precedent for ours. We list the main components of our research, and describe the focal work that biased us towards using or investigating those components:

Learning from implicit feedback: The work by Joachims [5] addressed the problem of using clickthrough data to train a system. The system used in that work was a Support Vector Machine (SVM). Building on this work, several others have showed that it is possible to learn reliably in an online framework i. e. from implicit feedback. Just like in the work by Hofmann et al. [4], we adopt the basic pairwise approach in this work. In a similar vein as Hofmann et al. [4], we also use a listwise algorithm. Our focal work for a listwise algorithmic approach is the SoftRank system by Taylor et al. [8].

Reinforcement learning: Since the work by Sutton and Barto [7], reinforcement learning has seen tremendous research in the machine learning community. The basic goal of reinforcement learning is to allow the *agent* to be *reinforced* by the *environment*. In other words, the goal of the agent is to maximize the rewards received from the environment. The naive way to do this is to always *exploit* what it knows already and receive a guaranteed reward at each time step. However, if large swathes of the solution space are unexplored, the performance of an exploitative agent would be highly sub-optimal. Hence, the agent needs to balance *exploration* with *exploitation* to maximize reward in the long run.

One way to frame this challenge in a reinforcement learning context, previously used in both information retrieval and other fields such as ad placement [6], is as a contextual bandit problem. Hofmann et al [4] used a method initially developed by Zhang [12] based on *stochastic gradient descent* in order to learn the weight vector for ranking documents. This method is particularly attractive because it requires few iterations, can attain optimal convergence with a constant learning rate, and is practical for both large datasets and online implementations [12]. In their pairwise learning experiments, Hofmann et al. [4] use a variant of this method that incorporates implicit feedback, updating the weight vector if the proposed labeled document pairs do not match to a certain threshold. They modeled the 'long run' as a discounted sequence of rewards over an infinite time horizon (in practice, the horizon ended after 1000 iterations in their experiments).

Hofmann et al. [4] also draw from *dueling bandit gradient descent* to perform reinforcement learning in a listwise setting. Dueling bandit gradient descent has been used previously in IR system optimization, notably in Yue and Joachims [11] who use it in the context of comparing two 'dueling' retrieval functions. DBGD compares the results of an exploitative point or vector w and an exploratory point or vector w' in a parameterized space and, depending on which performs better according to a given metric, updates the exploitative point w by projecting w' back into the space W, effectively updating w according to a given exploitative step size [11]. Although Yue and Joachims [11] use DBGD in a pairwise setting, it is effectively adaptable to a listwise learning to rank setting as in Hofmann et al. [4]

One key reinforcement learning algorithm is ϵ-greedy RL. Intuitively, the parameter $0.0 \leq \epsilon \leq 1.0$ controls the exploration-exploitation tradeoff. Hofmann et al. [4] use an ϵ-greedy approach in their pairwise learning to rank experiments and a variant of ϵ-greedy in their listwise experiments, and show that it performs above the baseline.

We are not aware of any works before Hofmann et al. [4] that use rigorous reinforcement learning algorithms in information retrieval. Hence, their work is our focal point for running experiments, formulating baseline and methodology, as well as guidance for future work and improvement. Hence, their work is arguably the focal reference for the overall research goals of this paper.

Active learning: Active learning is also a popular approach in the machine learning community; however, it is different from reinforcement learning in several ways. Unlike reinforcement learning, active learning has no formal provision for exploitation although practical systems use active learning in conjunction with some exploitation scheme. Instead, active learning dictates the rules of exploration. Its goal is to ensure that the *cost* of exploration is small. In other words, by intelligently exploring the unknown solution space, it is able to map it out with minimal user effort. One influential work in active learning in information retrieval was by Xu et al. [10]. They used an active learning method that used criteria of relevance, document diversity and document density to make decisions on what documents to present to the user. However, the focal reference for us in this area is the more recent work by Tian and Lease [9]. The reason why we find their work attractive is because they are able to abstract, successfully, the choice of features from the active learning itself and demonstrate the benefits of active learning as a standalone framework. They use a generic SVM to find vectors in the feature space that have maximum information gain by using a number of criteria, including the distance of the vector from the hyperplane surface of the SVM. Adapting their approach to enable active learning in the exploratory phase of reinforcement learning is a key contribution of our work.

3. CONTRIBUTIONS

In this paper, we propose the following contributions. Note that specific milestones and additional details are provided in a subsequent section. In this section, we merely aim to enumerate our high level goals for the final deliverables.

We construct an implementation of an *online* pairwise ranking algorithm, with user activity simulated using click models. The pairwise algorithm relies on implicit feedback, similar to the approach by Joachims [5]. However, the pairwise ranking algorithm is set in a reinforcement learning framework [7], shown to be an effective way of balancing exploration and exploitation in a recent work by Hofmann et al. [4]. Our original contribution is to place the *exploration* part of the setting in an active learning framework, instead of choosing documents randomly (as in [4]). In a series of experiments, Tian and Lease showed that active learning can be highly efficient with respect to minimizing user labeling effort in information retrieval [9]. We are currently not aware of any work (in information retrieval) that attempts to combine the benefits of both reinforcement learning and active learning.

We also repeat the above experiments for a *listwise* ranking algorithm (also online). An example of a listwise ranking algorithm, called SoftRank, is presented by Taylor et al. [8]. Again, rather than using random measures for the exploration part of reinforcement learning, we intelligently choose the space that we want to explore. Listwise ranking algorithms in a learning to rank setting still rely upon the testing of random ranking weight deviations to produce new weights. Our method decreases the amount of randomness and incorporates previously successful weight vectors into the calculation of exploratory weights.

We study the effects of limited supervised bootstrapping for the above approaches. A key observation upon surveying the current literature was that many authors tended to either adopt a supervised approach (i.e. an offline training and validation phase) or an approach that started with parameters that were random but that converged to acceptable values as more feedback was elicited from the user. For example, Hofmann et al. started with random, normalized weight vectors for their reinforcement learning algorithms and updated these vectors as implicit feedback was obtained [4]. Tian and Lease effectively do something similar, but in an active learning setting [9]. However, we would like to consider a compromise, based on practical needs. Specifically, we propose a *bootstrapping* phase before the online learning. Our hypothesis is that even if we generate our parameters (like the weight vector) based initially on a small set of training samples, and then use those parameters to start off the online session, we will witness considerable gains. An important related question is: how much bootstrapping is required, and is the system robust to noise in that phase? One key contribution of this paper is to begin to answer some of these questions for the algorithms and settings above.

4. RISKS

For pairwise learning, what if the ϵ-greedy random selection strategy of Hofmann et al. [4] actually works better than, or at least as well as, the *active learning* exploratory document selection criterion? Similarly, for listwise learning, what if the original dueling bandit gradient descent algorithm outperforms our modified approach? Either result would ostensibly argue against the assumed advantages of active learning and non-random exploratory document or list selection, and if we encounter either, it will be necessary to run additional experiments with controlled modifications to our original active learning algorithms in order to identify and subsequently report on the factors that counter our hypothesis.

Another issue could be that results are inconsistent or inconclusive across different experiments, which could prevent us from making firm conclusions regarding our findings. We use the cumulative NDCG metric to score our runs and perform basic descriptive statistical techniques such as student's t test to compare against baseline scores. If significance is not established in all experiments in a manner consistent with our hypothesis, then we must investigate the factors that may have made our hypothesis incorrect or difficult to test. In particular, finding unexpected differences between our listwise and pairwise algorithms' performance would require further investigation and, potentially, modification of one or both algorithms. It is a common adage that more data can always improve a learning model. However, several datasets are poorly formatted or otherwise unsuitable for our task. In addition, due to time constraints we limit our datasets to the TREC 2003 and 2004 tracks of LETOR 2.0 [1]. These datasets from Microsoft are designed to facilitate research in learning to rank. Therefore, our datasets limit the generalizability of our findings as they did in the work by Hofmann et al. [4].

5. EXPERIMENTAL DESIGN

In this section, we list our current progress per the milestones above.

5.1 Click Model

As in the work by Hofmann et al. [4] we have experimented with three different click models: *perfect*, *navigational* and

[1] http://research.microsoft.com/en-us/um/beijing/projects/letor/Letor2.0/dataset.aspx

Table 1: Overview of click models

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

Algorithm 1 SimulateProbability(p)

```
Input :
    • Probability $p$ (limited to two significant digits)
    • Integer Random number generator $g$
Output :
    • Boolean value True with probability $p$
Method :
    1. Initialize Set $S$ to be empty
    2. $range := [0, 100)$
    3. Randomly generate $100 - (p * 100)$ integers in $range$
       using $g$ and add to $S$
    4. Randomly generate an integer $q$ in $range$ using $g$
    5. if $q \in S$ then
           Terminate program and output False
    6. end if
    7. Terminate program and output True
```

informational. The click and stop probabilities, conditioned on whether the document was relevant or non-relevant, are outlined in Table 1. The perfect click model essentially assumes the user to be an oracle. Specifically, the user always clicks on a relevant document, and never clicks on a non-relevant one. The user never stops till the end of the list is encountered. Hence stop probabilities are zero. The informational and navigational models are based on empirically validated studies emulating typical user behavior in web search [1, 2, 3]. The navigational model is a slightly noisier version of the perfect model as far as clicking probabilities are concerned. The difference between probabilities is still large. However, the stopping probability, given a document is relevant is close to 1.0. This is because in a navigational task, users are looking for a very specific result and they stop once they find that result. In an informational task, the importance of recall is more emphasized and users are less certain about when to stop and when to click. All four probabilities show less deviation from each other than in the other two cases.

One of the first issues we encountered was how to simulate probability distributions. Specifically, given a probability we wanted to return a boolean value (for example, if we were simulating a user in a informational setting, and we knew a priori the document was relevant, we would return *True* with probability 0.9 per Table 1). We settled on a scheme that was simple and approximate but is shown to yield good results in practice. The pseudocode for our algorithm is shown in Algorithm 2. Essentially, the algorithm works by using a random number generator (ideally, this would be perfectly random, but is usually only pseudo-random in actual implementations) to independently generate $100-(p*100)$ integers in the range of $[0, 100)$ where p is the probability with which we want to return *True* and only has two significant digits (the data in Table 1 also assumes this). Next, we sample a single integer, again from the range $[0, 100)$. If this integer is equal to any of the integers generated in the previous step, we return *False* else we return *True*. To test the algorithm, we called it 1000, 10000 and 100000 times with $p = 0.9$. For the first two cases, the algorithm returned true around 90.2 percent of the time (90.1 percent for the last case). Hence, there is less than 0.2 percent error which further decreases as we sample more often. We ran a similar experiment with other probability values and the results were validated.

5.2 Metrics

We implemented more than we intended on this front. Originally, we were intending to only implement NDCG@10 by now and implement the other metrics later; currently, however, we have NDCG@10, MAP and Precision@10 implemented. We have conducted experiments on them and have determined them to be functioning as expected.

5.3 Data

We have succeeded in locating and parsing the datasets we intended to work on (Letor 2.0 TREC 2003 and 2004) into appropriate data structures. This was one of the first things we got set up in the code infrastructure. Note that we are implementing everything in Java.

5.4 Pairwise Algorithms

We implemented three algorithms for the pairwise case of implicit feedback. These, respectively, were the baseline case where the weight vector is learned offline using stochastic gradient descent, a click model and feedback [5], the baseline case supplemented to be online in a reinforcement learning framework as proposed originally in [4] and our proposed approach where we supplement the exploration phase of the reinforcement learning in the previous case with active learning.

MORE NEEDED HERE FROM MAYANK RE: PAIRWISE

5.5 Listwise Algorithms

There are four algorithms that were supposed to be implemented for the listwise case of implicit feedback. The first is the baseline dueling bandit gradient descent algorithm for the listwise learning procedure used in [4], which generates an exploitative list based on a current weight vector and an exploratory list based on a random deviation from the current weight vector with step size δ. These lists are combined using an *interleave* algorithm, and then compared based on

```
Algorithm 2 generateExploratoryWeight($w$)
  Input :
    • Matrix Previous exploratory weight vectors $W_p$
    • Vector current weight $w_t$
    • Double Step size $\sigma$
  Output :
    • Vector Exploratory Weight $w_e$
  Method :
    1. Initialize Vector $w_e$ to be empty
    2. Calculate mean vector $m_p$ = mean(row) for row in $W_p$
    3. Uniformly sample random vector and normalize to create unit vector $u_t$
    4. $w_e \leftarrow m_p + \sigma u_t$
    5. return $w_e$
```

click data. The combination and comparison steps are described as a function of the two original lists, $f(l1, l2)$.

Two methods are built for the purpose of list interleave. The first is the *balancedinterleave* method from Joachims [5]. This method minimizes presentation bias between two comparison lists by randomly selecting a starting list and then alternatingly sampling from each list such that an approximately equal (differing by at most 1) number of top results from each list (excluding duplicates) is included in the interleaved list [**?**]. Evaluation is achieved by observing user clicks on the interleaved list as follows: the lowest clicked document N is identified, then for each list the number of clicked documents within the top N is counted and the list with more clicks is selected.

The second is the probabilistic interleave method from Hofmann et al. [4]. This algorithm avoids potential biases originally mentioned by Joachims [5] by probabilistically selecting the list from which to draw a document at each rank of the interleaved list with probability $p \in [0, 0.5]$[4].

To the best of our knowledge, there are no existent *non-random exploration* methods in RL that have been adapted for the listwise learning to rank setting, and so we are cautiously developing the first. The dueling bandit gradient descent algorithm with intelligent exploratory list generation relies on a modified approach to producing the exploratory list wherein the list is generated not by a random weight vector, but rather by a weight vector that learns from prior exploratory list performance to deviate in an intelligently pseudorandom manner. Our current experimentation updates weights by biasing the exploratory vectorâĂŹs value on a number F of previous successful exploratory weight vectors (exploratory vectors that generated lists which received more clicks than the current exploitative list) and deviating from this value by the product of a uniformly random unit vector and a predetermined, small step size σ.

6. EXPERIMENTS

The experimental results reveal many notable trends. We discuss findings along different metrics and click models separately.

6.1 Pairwise Algorithms

6.1.1 Perfect Click Model

For the perfect click model, the baseline model is outperformed by both the reinforcement learning model and the active learning model for NDCG@10, Precision@10, and Mean Average Precision (MAP) for both the TD2003 and the TD2004 datasets. In terms of MAP measurements, the baseline was outperformed not only by both of the other models, but also at every level of r for the TD2004 dataset, and by most values of r for the TD2003 dataset.

In the TD2004 dataset, the highest overall values for every metric came from the reinforcement learning model with $r = 1$ (purely exploratory). This is consistent with the NDCG@10 and Precision@10 measurements for TD2003 as well, but not with the MAP values, where the active learning model outperformed the reinforcement model for values of $r = 0.6$ and higher. The single highest MAP value for the TD2003 experiments comes from the active learning model at $r = 1$.

Looking at both learning models, 9 of the 12 perfect click model metrics have maxima at $r = 1$, where exploration is maximized. This is inconsistent with the findings of Hofmann et al [cite], whose best performance on TD2003 and TD2004 is achieved when minimizing exploration.

6.1.2 Navigational Click Model

The navigational click model provided slightly worse results than those of the perfect click model. This is consistent with the findings of previous Learning to Rank literature [cite Hofmann et al].

Interestingly, when using the navigational click model, the reinforcement learning model performed best at a balanced value of exploration (r=0.4), rather than at an extreme, for all metrics of both TD2003 and TD2004 except for Precision@10 in TD2003, where the best performance was found at r=1. The active learning model, by comparison, had more scattered maxima: for TD2004, performance in terms of NDCG@10 and Precision@10 was best at r=1, MAP was best at r=0.2; for TD2003, all models performed best at r=0.6. The active learning outperformed the reinforcement learning model in terms of both NDCG@10 and MAP when tested using TD2003, and also bested the MAP and Precision@10 scores of the reinforcement learning model when tested using TD2004.

Overall, the active learning model appeared to show greater improvement over the reinforcement learning model when applying the navigational click model than when applying the perfect click model.

6.1.3 Informational Click Model

The informational click modelâĂŹs experimental results are roughly comparable to those of the navigational click model, and slightly worse than the perfect click model. However, the informational click modelâĂŹs best results have the greatest variance in values of r between the three metrics and two datasets.

For the TD2003 dataset, the reinforcement learning model maximizes NDCG@10 and Precision@10 values at r=0.4 and maximized MAP at r=0.2. The active learning model maximizes performance for NDCG@10 and MAP at r=0.4 and Precision@10 for r=0.2. These findings suggest a balanced exploration/exploitation approach is best for TD2003, which is generally consistent with Hofmann et al [cite]. The active learning model outperforms the reinforcement learning model in terms of both NDCG@10 and Precision@10, but not MAP.

For the TD2004 dataset, the reinforcement learning model outperforms the active learning model for all metrics. The reinforcement learning model performs best when totally exploratory for NDCG@10 and Precision@10, and totally exploitative for MAP. The active learning model performs best when totally exploitative for NDCG@10 and Precision@10, and highly exploratory ($r = 0.8$) for MAP.

6.2 Listwise Algorithm

The Listwise baseline experiment provided measurements slightly higher than those of the Pairwise experiments. This is consistent with Hofmann et al [cite]. However, the moving average experiment was not able to improve on this baseline for any test, metric, dataset, or fold.

There are several possible explanations for this. One is that the datasets are not large enough to evaluate the differences between the listwise algorithms, which do not gather information directly from particular documents and, as a result, may struggle to effectively improve the ranking weight. In addition, as previously mentioned the $TD*$ datasets are sparse in terms of relevant documents for each query: in general, fewer than five percent of documents were found to be relevant. These two factors may have made the updating process ineffective for a majority of queries; indeed, in our baseline experiment the exploratory list scored more highly than the exploitative list three times in a thousand iterations.

In addition, the moving average approach carries several operational risks. Chief among them, it may overemphasize improvements to the ranking weight vector by both updating it according to a successful exploratory weight and by factoring previously successful exploratory weights into the generation of new exploratory weights. This effectively counts such weights *twice* when generating a new exploratory weight, and may lead the weight vector to overcompensate after successful explorations.

Although the experiment did not improve on the current listwise learning to rank algorithm, due to its structure and the difficulty of updating the ranking weight in an intelligent and automated manner, there are very few known approaches to the task. As such, the fact that our approach was unsuccessful may be helpful to scholars seeking to avoid dead ends in future research endeavors in this area.

7. ANALYSIS

In general, our findings confirm that both learning methods are able to perform higher than the baseline, and support the emphasis of exploration over exploitation. This second finding is in contrast to the findings of Hofmann et al [cite], whose results were higher for reinforcement learning models that favored exploitation over exploration, or a near-even balance of the two, for all click models, and especially for the perfect click model.

The perfect click model is the least noisy of the click models, yet exploration is most reliably high here, which seems contradictory. However, it bears noting that both of the $TD*$ datasets are low-relevance datasets; they have very few relevant documents for each query. This may overemphasize the benefits of exploration as well as the superiority of both models to the baseline, as the benefits of exploration are more apparent when the target relevant documents are sparsely distributed (an exploitative algorithm is likely to have more trouble finding them). Results may have differed for datasets with a greater proportion of relevant documents to irrelevant documents for each query.

In addition, the reinforcement learning model appears to dominate the active learning model when using a perfect click model, but the reverse is true of the navigational click model, and the models are competitive when using the informational click model. This may be due to the increased noise of the informational and navigational models over the perfect click model, which, coupled with the sparseness of relevant documents for each query, allows active learning to gain information more effectively by targeting noisy documents to a greater extent than in the reinforcement learning approach, which randomly targets exploratory documents.

8. REFERENCES

[1] A. Broder. A taxonomy of web search. In *ACM Sigir forum*, volume 36, pages 3–10. ACM, 2002.

[2] F. Guo, C. Liu, A. Kannan, T. Minka, M. Taylor, Y.-M. Wang, and C. Faloutsos. Click chain model in web search. In *Proceedings of the 18th international conference on World wide web*, pages 11–20. ACM, 2009.

[3] F. Guo, C. Liu, and Y. M. Wang. Efficient multiple-click models in web search. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 124–131. ACM, 2009.

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

[5] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002.

[6] S. A. . W. J. Langford, J. Exploration scavenging.

[7] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. Cambridge Univ Press, 1998.

[8] M. Taylor, J. Guiver, S. Robertson, and T. Minka. Softrank: optimizing non-smooth rank metrics. In *Proceedings of the international conference on Web search and web data mining*, pages 77–86. ACM, 2008.

[9] A. Tian and M. Lease. Active learning to maximize accuracy vs. effort in interactive information retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 145–154. ACM, 2011.

[10] Z. Xu, R. Akella, and Y. Zhang. Incorporating diversity and density in active learning for relevance feedback. In *Advances in Information Retrieval*, pages 246–257. Springer, 2007.

[11] Y. Yue and T. Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proceedings of the international conference on Machine learning*, pages 1201–1208. ACM, 2009.

[12] T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the international conference on Machine learning*, pages 116–. ACM, 2004.