# 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
sam.blazek@gmail.com

## 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 [6], 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 [8]. The benefits of active learning have also been demonstrated in other domains, and corroborated by other researchers in the information retrieval community as well [9]. 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 will be using an active learning approach.

We will also demonstrate the effects of bootstrapping an online system, and the susceptibility of the system to bootstrapping noise. By bootstrapping, we mean that we will 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. Other works have addressed learning from implicit feedback as well, but the paper by Joachims [5] seems to have been heavily cited, and highly influential in the area. We chose this work as central for this aspect of the research, therefore. 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. [7].

- *Reinforcement learning*: Since the work by Sutton and Barto [6], 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. In their work, Hofmann et al. 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). One key reinforcement learning algorithm is $\epsilon$-greedy RL. Intuitively, the parameter $0.0 \leq \epsilon \leq 1.0$ controls the exploration-exploitation tradeoff. 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. [9]. 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 [8]. 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 will be 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.

- An implementation of an *online* pairwise ranking algorithm, with user activity simulated using click models. The pairwise algorithm will rely on implicit feedback, similar to the approach by Joachims [5]. However, the pairwise ranking algorithm will be set in a reinforcement learning framework [6], shown to be an effective way of balancing exploration and exploitation in a recent work by Hofmann et al. [4]. Our original contribution will be 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 [8]. 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 will 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. [7]. Again, rather than using random measures for the exploration part of reinforcement learning, we will intelligently choose the space that we want to explore.
- We will 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 [8]. 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. If this effect is experimentally observed, a related question is how much bootstrapping is required and whether the system is robust to noise in that phase. One key contribution of this paper will be to answer some of these questions for the algorithms and settings above.

# 4. RISKS

For pairwise learning, what if the $\epsilon$-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, we will run additional experiments with controlled modifications to our original active learning algorithms in order to identify and 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. MILESTONES

**For October 31, 2013:**

(1) We will construct the stochastic gradient descent algorithm for pairwise feedback and pairwise learning used in the work by Hofmann et al. [4].
(2) We will construct the $\epsilon$-greedy algorithm for exploratory and exploitative document selection as used in the work by Hofmann et al. [4].
(3) We will construct the active learning procedure for exploratory and exploitative document selection so we can run controlled experiments.
(4) We will construct the bootstrapping procedure for exploratory and exploitative document selection so we can run controlled experiments.
(5) We will construct the NDCG code for model performance evaluation.
(6) We will perform the pairwise learning experiments and examine the NDCG scores for all approaches and using different parameter values on each dataset. We will compare scores using descriptive statistics.

**For November 14, 2013:**

(7) We will construct the dueling bandit gradient descent code for the listwise learning procedure used in [4].
(8) We will construct the balanced interleave method for list comparison used in the work by Hofmann et al. [4].
(9) We will construct the dueling bandit gradient descent algorithm for listwise selection with modified (active learning-inspired) exploratory list production.

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

(10) We will construct the probabilistic interleave method for list comparison used in the work by Hofmann et al. [4].
(11) We will perform the listwise learning experiments and examine the NDCG scores for all experiments and for different parameter values on each dataset. We will compare scores using descriptive statistics.

**For November 21, 2013:**

(12) Additional evaluation metrics and methods may or may not be explored, time permitting.
(13) We will plot our data to illustrate the differences between our findings and those of Hofmann et al. [4].
(14) We will analyze and make conclusions based on our findings.
(15) We will draft our final paper.

**For December 4, 2013:**

(16) We will complete our final paper.
(17) We will build a poster that includes:

- A Summary
- Description of Relevant Techniques and Methods (Stochastic Gradient Descent,$\epsilon$-greedy, active learning, bootstrapping).
- Hypotheses statements
- Snippets of code
- Graphics showing the NDCG and statistical significance of the active learning and bootstrapping (for pairwise learning) methods vs. the $\epsilon$-greedy methods.
- Conclusions

## 5.1 Division of Work

Both authors will jointly design algorithm implementations for:

- Stochastic Gradient Descent
- $\epsilon$-greedy Exploratory and Exploitative Document Selection
- Active Learning Exploratory and Exploitative Document Selection
- Bootstrapping Exploratory and Exploitative Document Selection
- Dueling Bandit Gradient Descent
- Dueling Bandit Gradient Descent with Modified Exploratory List Generation
- Balanced Interleave Method for List Comparison
- Probabilistic Interleave Method for List Comparison

Both authors will jointly perform the experiments, self-delegating as computation and time allow. They will modularly develop code to perform the NDCG calculations and descriptive statistical tests, and generate the relevant plots. Finally, they will write about the findings as a class paper.

**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*

# 6. CURRENT PROGRESS

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

## 6.1 Click Model

As in the work by Hofmann et al. [4] we have experimented with three different click models: *perfect*, *navigational* and *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 1. 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.

## 6.2 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.

## 6.3 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.

## 6.4 Pairwise Algorithms

There are three algorithms that were supposed to be implemented 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.

We have succeeded in setting up the code infrastructure for all three algorithms and initial tests show promising evidence that the baseline case is working, at least on the perfect click model. Our numbers look reasonable but more testing will be required to see if subtle bugs are present in the code. We are currently in the process of doing so.

### 6.4.1 Immediate Goals

The immediate goals on this front are to ensure that the pairwise algorithm above is absolutely correct on the baseline. The initial numbers do look promising but we will run more experiments to ensure everything is correct before moving forwards. Logically, the design of the other two algorithms (one is in the Hofmann paper) has already been decided, so it is a matter of implementing them and collecting data comparing all three versions of the algorithm.

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

[7] 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.

[8] 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.

[9] 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.