# Multi-Armed Bandits for Adjudicating Documents in Pooling-Based Evaluation of Information Retrieval Systems

David E. Losada[a], Javier Parapar[b], Alvaro Barreiro[b]

[a]*Centro Singular de Investigación en Tecnoloxías da Información (CiTIUS)*
*Universidade de Santiago de Compostela, Spain*
david.losada@usc.es
[b]*Information Retrieval Lab*
*Department of Computer Science*
*University of A Coruña, Spain*
{barreiro,javierparapar}@udc.es

**Abstract**

Evaluating Information Retrieval systems is crucial to making progress in search technologies. Evaluation is often based on assembling reference collections consisting of documents, queries and relevance judgments done by humans. In large-scale environments, exhaustively judging relevance becomes infeasible. Instead, only a pool of documents is judged for relevance. By selectively choosing documents from the pool we can optimize the number of judgments required to identify a given number of relevant documents. We argue that this iterative selection process can be naturally modeled as a reinforcement learning problem and propose innovative and formal adjudication methods based on multi-armed bandits. Casting document judging as a multi-armed bandit problem is not only theoretically appealing, but also leads to highly effective adjudication methods. Under this bandit allocation framework, we consider stationary and non-stationary models and propose seven new document adjudication methods (five stationary methods and two non-stationary variants). Our paper also reports a series of experiments performed to thoroughly compare our new methods against current adjudication methods. This comparative study includes existing methods designed for pooling-based evaluation and existing methods designed for metasearch. Our experiments show that our theoretically grounded adjudication methods can substantially minimize the assessment effort.

*Keywords:* Information Retrieval, Evaluation, Pooling, Reinforcement Learning, Multi-armed Bandits

## 1. Introduction

To measure the effectiveness of Information Retrieval (IR) systems, it is customary to build benchmarks consisting of a document collection, a series of information needs, and a set of relevance judgments [37, 13]. Standard collections, like those developed under the Text Retrieval Conference (TREC) [49], are so large that exhaustively judging every query-document pair becomes infeasible. Furthermore, with complete relevance judgments, human assessors' time would be mostly dedicated to analyzing non-relevant documents. Such an exhaustive approach would be a waste of time and effort. The most productive use of human assessors occurs when they judge documents deemed to be relevant [39]. This is why most assessment processes in IR experimentation are supported by a sampling method called *pooling*.

Pooling is a traditional method [43] that has been extensively used in campaigns like TREC, CLEF (Conference Labs of the Evaluation Forum), NTCIR (NII Testbeds and Community for Information Access Research) and INEX (Initiative for the Evaluation of XML Research). In pooled test collections, relevance judgments are only done for the documents that were among the top retrieved for some systems that participated in the evaluation campaign. The top retrieved documents have the greatest impact on effectiveness and, therefore, estimates made in this way are accurate.

The pooling methodology works as follows: i) given a document collection, the campaign's organizers define a search task that tests the ability of systems to retrieve relevant documents in response to a set of queries (often known as topics), ii) different groups participating in the task submit their system's results, iii) for each query, the top $k$ documents from each participating system are pooled into a single set and presented to assessors for judging. Under

this setting, the rankings of documents produced by the participating systems are commonly known as *runs*; $k$ is the *pool depth*; and the resulting set of judgments are known as *qrels*. Having runs from a sufficient variety of systems and a reasonable setting for $k$ (typically set to 100), the assessments can be done at an affordable cost and the resulting benchmark is solid and reusable [49].

If we can afford the cost of judging the whole pool then we can just pass the pooled documents to the assessors in random or arbitrary order. Instead, if our budget is limited, then we might want to judge a subset of the pool. This has motivated the emergence of a stream of proposals on how to adjudicate pooled documents for judgment [15, 29, 14]. An effective adjudication method selects documents from the pool following a given criterion. These adjudication methods, when compared with random or arbitrary alternatives, can substantially reduce the assessment effort required to produce a qrel file with a sufficient number of relevant documents [29].

Selection strategies for labeling items from a pool of unlabeled items is of interest well beyond Information Retrieval. In many data mining applications, unlabeled data is abundant and manually labeling is expensive. Supervised learning –e.g. classification– requires a set of labeled examples and it is crucial to reduce the costs associated with creating the training data. This has motivated the emergence of a large number of studies on pool-based selection for learning [35]. Here, we are only concerned with the specifics of IR pooling but the lessons learned from our research are potentially applicable to other areas.

The assessment process needed to create an IR test collection can be seen as a "learning from interaction" process. The more assessed documents we have, the more we learn about the relative quality of the runs. Here we propose an innovative and formal adjudication approach based on multi-armed bandits. Research on document adjudication for IR evaluation has been mostly adhoc and has largely ignored the lessons learned in reinforcement learning.

The multi-armed bandit problem [36], also known as K-armed bandit problem, is a long-established problem in reinforcement learning. Reinforcement learning is concerned with how an autonomous system interacts with an uncertain environment so as to maximize a numerical reward over some time period. The system is not explicitly told which actions to take but, instead, must learn which actions yield the most reward by testing them out. At any time point, each possible action has an estimated value and the system can opt to exploit its current knowledge (i.e. try the action whose estimated value is greatest). This exploitative choice is often called the greedy action. Alternatively, the system can choose to explore non-greedy actions. Exploring the environment in this way enables the system to improve its estimates for non-greedy actions. Exploitation is the right thing to do to maximize the short-term expected reward, but exploration may produce better results in the long run. Multi-armed bandits offer a theoretical framework for analyzing the trade-off between exploration and exploitation. Due to its generality, the exploration vs exploitation dilemma has been studied in many disciplines [44], including medicine, economics, and ecology. For instance, balancing between exploration and exploitation has been employed to investigate the effects of different experimental treatments while minimizing patient losses [33]. We show here that balancing exploration and exploitation has also a practical application in IR evaluation and, in particular, in how to build qrels from a set of runs from different systems. Within this process, concentrating only on systems that currently look effective is risky. We can miss relevant documents that are only supplied by other apparently inferior runs. Multi-armed bandit algorithms are a natural solution to address this balance formally.

In summary, this paper contributes in the following interrelated aspects:

- We adapt multi-armed bandit models to address the problem of document adjudication in pooling-based evaluation of search algorithms. This innovative use of reinforcement learning leads to seven new effective adjudication methods that early identify relevant documents in the pools. Furthermore, this is a theoretically-grounded framework where we can analyze the exploration/exploitation dilemma. In doing so, we show how different document adjudication methods behave with respect to this dilemma.
- We conduct a thorough comparison of existing adjudication methods and confront their merits against effective models of metasearch. While a number of isolated studies have analyzed and proposed different adjudication methods, the literature is lacking a complete picture of their effectiveness. There is little experimental evidence on the relative merits of existing adjudication methods when compared with effective metasearch models. In this paper we try to fill this gap by performing a thorough evaluation of existing adjudication methods and a comparison of these methods against our seven bandit-based solutions. Our study comprises reference methods specifically designed for pooling-based evaluation and reference methods designed for metasearch. To the best of our knowledge, this is the first study that evaluates such a highly diversified portfolio of adjudication methods.

- We compare the most effective document adjudication methods with respect to their ability to identify relevant documents and in terms of the induced bias. By judging only a subset of the pooled documents we are inducing a bias with respect to judging all pooled documents. This bias is reduced as we judge more documents. We study the biases induced by different methods and their evolution. This complements the evaluation based on counts of relevant documents found, which is merely focused on finding relevant items at any cost.
- We show that some non-stationary instances of the bandit-based models are regularly superior to all previous adjudication methods. As a matter of fact, one simple Bayesian model requires fewer human assessments than any other competing approach and leads to a ranking of retrieval algorithms that highly correlates with the official system rankings.
- We are strongly committed to making our experiments reproducible. We provide the R code of all our implementations and we encourage other teams to experiment with them and build new solutions based on our code and instructions.

This paper is an extended version of work published in [27]. We have extended our previous work in a number of ways. First, by including metasearch models into the study. In particular, we show here that one of the metasearch models tested is extremely effective in terms of the early finding of relevant documents. Second, by not only studying the patterns of relevant documents found but also the evolution of the bias of the different strategies while increasing the number of judgments. Third, by doing a rigorous statistical analysis of the differences found in the counts of relevant documents. Fourth, by extending the discussion on related work, as well as the set of algorithms and examples.

The remainder of the paper is organized as follows: section 2 reviews the related work. In section 3, we provide some definitions to formally characterize the document adjudication problem. Section 4 presents our novel framework to cast document adjudication as a multi-armed bandit problem. Section 5 briefly discusses existing methods to adjudicate pooled documents for judgment and section 6 sketches two metasearch algorithms that can be used for adjudicating judgments. Experiments are reported in section 7 and a bias analysis is provided in section 8. A discussion of the findings is presented in section 9. The paper concludes a summary of the contributions and hints at possible lines of future work.

## 2. Related Work

Pooling is a fundamental technique in IR evaluation and has attracted attention for decades [48, 50, 15, 14, 8, 21]. Pooling strategies are regularly employed in campaigns like TREC [49] or CLEF [17]. Over the years, pooling has been supporting not only standard search tasks but also other Information Access specialized tasks. For instance, pooling was recently employed for building an evaluation framework for link discovery [45].

Many research teams have explored ways to efficiently scan pools, with the objective of identifying a sufficient number of relevant documents as quickly as possible. Subset pooling methods can save substantial costs by reducing the assessment effort and, thus, they have been adopted in evaluation campaigns like INEX [23]. But cost is not the only reason to embrace subset pooling methods. Even if you have a large budget for doing judgments, that does not mean that judging deeply the pools is the best use of your budget. As argued by Sanderson [37], building test collections with more topics and fewer judgments per topic is an increasing priority. Several studies have shown that the resulting benchmarks are more powerful and reusable when compared to benchmarks constructed with fewer queries and more judgments per query [9, 39, 7]. The quality and reusability of collections constructed using subset pooling strategies has been thoroughly demonstrated in the past. For instance, Moffat et al [29] concluded that a few hundred judgments per query is sufficient to produce good bounds on the effectiveness of the systems. Similar conclusions were drawn by Aslam and his colleagues [5], Carterette and his co-authors [10], and Cormack and Lynam [14]. A recent study on rank fusion based on score distributions also suggested that shallow judgments lead to reliable test collections [28].

There is therefore substantive evidence to support subset pooling strategies and to employ them to assess a reduced set of documents per query. In well-known competitions and campaigns, there is an increasing tendency to do fewer judgments per query by focusing the judgment effort on selected samples from the pool. This strategy has been

adopted in the TREC filtering track [42], in Speech Retrieval [32], in the Million Query TREC Track [11] and in building collections for specific domains [31].

Well-known subset pooling methods, such as Move To Front [15] or the methods proposed by Moffat et al. [29], have been considered in our study and compared against our bandit-based solutions. Our multi-armed bandit models address this assessment task in a more formal way and lead to competitive solutions.

Aslam et al. [4] have employed online learning to simultaneously solve the problems of metasearch, pooling and system evaluation. Their most succesful proposal was the Hedge algorithm –also included in our study– that produces metasearch lists whose quality equals or exceeds that of benchmark methods such as CombMNZ or Condorcet.

Other authors have opted to build test collections with no system pooling. For instance, Sanderson and Joho [38] experimented with interactive relevance feedback with no system pooling and also tested a strategy based on the use of a single manual run to form the qrels. Soboroff et al. [41] proposed an experimental methodology that replaces human relevance judgments with *pseudo-relevance* judgments, which are obtained from random sampling from the pool. Recent work by Jayasinghe et al. [21] proposed a novel methodology to generate diverse judgments. A valuable source of diversity comes from manual runs, which contribute many unique relevant documents to the pool. Manual runs are, however, expensive to produce. Jayasinghe and his colleagues combined a voting model and supervised learning technology –fed by documents supplied by automatic runs– and were able to identify relevant documents that would normally only be found through manual runs. Instead of building a static collection from a finite set of systems, Tonon and his colleagues [47] argued recently for a new IR evaluation paradigm where retrieval approaches are evaluated iteratively on the same collection. This proposal was based on the continuous use of either crowdsourcing or professional editors to obtain relevance judgments. All of these strategies are complementary to our own.

Multi-armed bandit models have been applied in multiple application domains. For example, bandit-based solutions have been employed for assigning patients to clinical trials [33]. In IR, multi-armed bandit solutions have recently been employed in several ways. Hofmann et al. [20] proposed a bandit-based solution to capture interactions between search engines and users for improving online learning to rank. In this problem, an exploitation versus exploration tradeoff arises naturally: the search engine wants to exploit what is already known to be a good ranker, but it also wants to explore by trying out variations of the current ranker. Therefore, two lists of documents are maintained: an exploitative list (based on the current best ranker) and an exploratory list (based on variations to explore potential improvements). The user is presented with an interleaved list, and preferences are inferred from his clicks. Similarly, Yue and Joachims [53] proposed an online learning framework based on bandits for comparing retrieval algorithms. The method gathered implicit feedback from users in the form of ordinal judgments and also learned by observing interleaved results. Sloan and Wang [40] coupled multi-armed bandits with the Portfolio Theory and defined a dynamic model that tries to maximize the user's satisfaction by combining relevance and diversity. Kleinberg and colleagues [34] were also interested in exploring relevance and diversity and presented a multi-armed bandit algorithm that learns a diversified ranking of results. Their approach analyses clickthrough data and attempts to minimize user abandonment in interactive experiments. Recently, Li et al. [24] demonstrated the advantage of bandit-based models for complex information needs composed of multiple aspects. Their method was oriented to rate limited search services and their approach was based on multiple queries, which are kept active simultaneously, and a bandit algorithm that chooses among them.

The exploration versus exploitation dilemma can also be handled with learning models not based on bandits. Karimzadehgan and Zhai [22] followed a machine learning approach for optimizing the utility of relevance feedback in a session of user interaction. The tradeoff was between presenting search results with the highest immediate utility and presenting search results with the best potential for collecting useful feedback. For example, judging documents that all have similar contents is not particularly useful in terms of feedback when compared to judging more diversified documents.

## 3. The Document Adjudication Problem

First, let us formally define the document adjudication problem in pooling-based evaluation. For each query, we have multiple rankings of documents in decreasing order of estimated relevance (multiple runs supplied by different search systems):

$$
\begin{aligned}
(run_1) &: d_{1,1}, d_{1,2}, \ldots, d_{1,l_1} \\
(run_2) &: d_{2,1}, d_{2,2}, \ldots, d_{2,l_2} \\
&\vdots \\
(run_m) &: d_{m,1}, d_{m,2}, \ldots, d_{2,l_m}
\end{aligned} \tag{1}
$$

where $d_{s,j}$ is the document retrieved at the $jth$ position by $run_s$. Every system run implements a different retrieval strategy and, therefore, the length of the runs ($l_i$s) can have some variance. Only documents ranked above a given cutoff (pool depth, $k \in \{1, .., max(l_1, ..., l_m)\}$) are candidates for judgment. Given the pool depth $k$, the documents contributed to the pool by $run_s$ are:

$$
candidates\_run_s \quad = \left\{ d_{s,p} \right\}_{1 \leq p \leq k} \tag{2}
$$

And the pool is formed by collecting all candidate documents from all runs:

$$
pool \quad = \bigcup_{1 \leq s \leq m} candidates\_run_s \tag{3}
$$

A document adjudication strategy is an algorithm that takes the set of runs $\{run_1, \ldots, run_m\}$ and the pool depth ($k$) as an input and produces a sequence of documents to be judged for relevance ($PoolOrder$). The output, $PoolOrder$, is a permutation of the pool: $PoolOrder \in Perm(pool)$.

Some adjudication methods are *static*: the ordering of the pooled documents is only based on the input runs, and a full ordering of the pooled documents is done before seeing any relevance assessment. *Dynamic* methods, instead, iteratively select documents from the pool, evaluate the relevance of the chosen document and the outcome of this relevance assessment affects the decision on the next pick. Dynamic methods can quickly adapt to the outcome of the assessments performed so far. For instance, we can dynamically avoid poor search systems or re-rank the remaining unjudged documents. This can improve our chances of early finding relevant documents. However, dynamic methods put an additional burden on building the benchmark [26]. The whole process needs to be coordinated such that the next relevance assessment cannot start until the previous assessment has finished. However, this does not mean that all assessments must be done by a single person in a serial approach. Every time we need an assessment we can send the document for judgment to several human assessors and, given the individual assessments, we can obtain the relevance outcome by, e.g., majority. This complicates the evaluation exercise (the time for doing an assessment varies with factors such as the abilities of the assessors, or their workload) but guarantees that assessments are done by aggregating judgments from multiple humans. Furthermore, other speed-up strategies, based on parallel distribution across multiple assessors can be implemented. For example, judgments for different queries can be interleaved so that we make the most of the assessors' time.

The challenge is to identify relevant documents as early as possible and, therefore, we will prefer adjudication methods that output sequences of documents with many relevant documents at the beginning. A standard way to compare adjudication methods is to plot recall against the number of judgments done. An optimal adjudication strategy would choose all relevant documents first, and all non-relevant documents would be at the end of the sequence. In practice, an effective adjudication method is convenient because we could stop the assessment process after extracting a sufficient number of relevant documents.

## 4. A Bandit-Based Approach for Adjudicating Judgments

The multi-armed bandit problem was defined by Robbins [36] as follows. Imagine a gambler at a row of $K$ slot machines (or *bandits*). Each machine has an unknown probability of distributing a prize and, when played, provides a numerical reward. Formally speaking, we have a set of $K$ distributions $\{R_1, \ldots, R_K\}$, each distribution being associated with the rewards delivered by one of the $K$ machines; and $\mu_1, \ldots, \mu_K$ are the mean values associated with

these reward distributions. The gambler has to choose one machine per round and his objective is to maximize the sum of rewards earned over some time period. The regret $\rho$ after $T$ plays is defined as the expected difference between the sum associated with an optimal strategy and the sum of the obtained rewards:

$$\rho \quad = \quad \mu_{opt} \cdot T - \sum_{n \in \{1..T\}} r_n \tag{4}$$

where $r_n$ is the reward obtained at round $n$, and $\mu_{opt} = max_l\{\mu_l\}$.

Through repeated plays the gambler tries to maximize his winnings by picking the best machines. The problem is complicated, however, by its stochastic nature. A sub-optimal machine can return many winnings, purely by chance; and an optimal machine can return many losses. So, if the gambler finds a machine that returns reasonably good results should he keep playing this machine to maintain his current good score or, should he try other machines in hopes of finding an even-better machine? This is the classical exploration versus exploitation tradeoff. Depending on factors such as the uncertainty of the current estimates and the number of remaining plays, it might be more profitable to exploit or to explore. The optimal solution to this problem is difficult to obtain. However, there are many balancing methods that implement approximately-optimal solutions and scale very well.

### *4.1. Employing bandit allocation methods in pooling-based evaluation*

Linking pooling-based evaluation and multi-armed bandit algorithms is a novel contribution of our research and permits the application of the lessons learned over many years in this active subarea of reinforcement learning. Our proposal consists of adapting existing multi-armed bandit solutions for sequentially selecting retrieval systems that contributed to the pool. This is a natural application of reinforcement learning for pooling-based evaluation of search algorithms. The resulting formal models iteratively learn about the quality of the runs and react to changes in the observed relevance of the assessed documents. Under this framework, we can therefore define new dynamic adjudication methods whose decisions are driven by formal models.

Initially, we have no knowledge on the quality of the runs. As we extract and judge documents for relevance, we learn about the quality of the runs and the assessment process can be directed at the most effective runs. At any given moment, we can opt for inspecting sub-optimal runs. These currently inferior runs can actually become suppliers of relevant documents. Playing a machine here means picking a run, extracting the top unjudged document and judging it for relevance. The outcome of the play –reward– is the relevance degree of the assessed document. We obtain the judgments from the official TREC qrel file, which contains relevance judgments for all pooled documents. In this paper, we work with binary relevance and, therefore, constrain our discussion to bandit models with binary rewards.

### *4.2. Allocation methods*

An allocation method, also known as policy or selection strategy, is the core component of any multi-armed bandit algorithm: it decides which machine to play next. The mechanism to make this selection depends on past plays and rewards. Each allocation method captures distinct ideas on how to handle the exploration versus exploitation dilemma. The following sections describe the main features of well-known allocation methods and how to apply them in pooling-based evaluation.

#### *4.2.1. Random*

This naïve allocation method chooses the next machine to play in a random way. It is a baseline commonly used in experiments with bandits. In our pooling environment, this simply consists of randomly selecting a run and assessing the next document supplied by the run.

#### *4.2.2. $\epsilon_n$-greedy*

The greedy choice is to pick the machine with the highest average reward (based on previous plays). In our case, where each bandit is a run and the associated reward is the binary relevance of the document supplied by the run,

the average reward is the mean relevance of the documents assessed from the run[1]. This greedy approach maximizes immediate rewards and spends no time at all inspecting apparently inferior actions. However a purely greedy method often gets stuck performing sub-optimal actions and it offers poor performance in the long run. A natural alternative is to randomly oscillate between the greedy choice and a random one. The $\epsilon$-greedy algorithm [44] trades off exploitation and exploitation in this way.

At each round, the $\epsilon$-greedy algorithm picks a random machine with probability $\epsilon$, and picks the machine with the highest average reward otherwise (probability $1 - \epsilon$). In such a simple way, the algorithm alternates between a purely randomized method and the exploitative instinct of maximizing profits. This alternation improves the chances of the algorithm recognizing optimal actions and makes the $\epsilon$-greedy superior to a purely greedy approach.

As we see more results, our estimates about the machines become more accurate and we might want to gradually reduce exploration. The $\epsilon_n$-greedy algorithm implements this idea by decreasing the exploration parameter as follows:

$$\epsilon_n \quad = \quad min(1, \frac{c \cdot K}{d^2 \cdot n}), n = 1, 2, \ldots \tag{5}$$

where $n$ is the round number, $K$ is the number of machines, and $c > 0$ and $0 < d < 1$ are parameters of the algorithm. The value of $d$ is usually set to the difference (in expected reward) between the best action and the second best[2]. Under certain conditions, this setting allows a logarithmic bound on the regret to be proven [6].

In noisy environments it takes more exploration to find the optimal action and $\epsilon$-greedy methods fare even better relative to the greedy method. In non-stationary situations, where bandits change over time, exploration is needed to make sure that a non-greedy action has not changed to become better than the greedy one. Non-stationary is the case most commonly encountered in reinforcement learning [44]. This is also true in our pooling environment where the relative quality of the runs varies as we go down in the ranked list of documents.

#### *4.2.3. Upper Confidence Bound (UCB)*

UCB methods compute upper confidence bounds for the estimates associated to each machine. After $n$ plays, the leading machine is the one with the largest empirical mean of obtained rewards. While we could be tempted to pick this leading machine, we first need to be sure that the other machines have been inspected enough times. Otherwise, we cannot be sufficiently confident that they are indeed inferior. The UCB allocation methods consider the uncertainty associated with each estimate, compute upper confidence bounds for all machines and make the selection based on these bounds. For instance, the *UCB1* policy [6] defines an index that is the sum of the current average reward plus a term related to the size of the one-sided confidence interval for the average reward. This index derives from Agrawal's policy [1], which achieves asymptotic logarithmic regret behavior. According to [1], the probability that the true expected rewards falls within this interval is very high.

*UCB1-Tuned* [6] is an effective enhancement of UCB1 that takes into account the variances of the average rewards. We implemented and experimented with both UCB1 and UCB1-Tuned and found that UCB1-Tuned was consistently better than UCB1. We therefore constrain our discussion to UCB1-Tuned, whose adaptation for pooling is shown in Algorithm 1. In the algorithm, $\mu_s$ and $\sigma_s^2$ refer to the sample mean and variance of the rewards obtained from run $s$ so far, $n_s$ is the number of times run $s$ has been visited, and $n$ is the overall number of assessments. The quantity added to the sample average is steadily reduced as the run is visited, and uncertainty about the reward probability is reduced. As a result, by always selecting the run with the highest bound, UCB1-Tuned gradually moves from exploration to exploitation.

#### *4.2.4. Bayesian Bandits*

The allocation methods outlined in the previous sections are frequentist in nature. The average rewards are regarded as unknown deterministic values and the learning algorithm aims to attain the best parameter-dependent performance. Bayesian methods, on the other hand, deal with multiple alternative hypotheses and quantitatively weight

[1] Initially, all averages are set to 0.5.

[2] We set $d$ to 0.1 and $c$ to 0.01. In our initial tests we found that performance was insensitive to $d$ and moderately sensitive to $c$. We tested $c \in (0, 0.15]$ (steps of 0.01); all tested $c \in (0, 0.1]$ were optimal and all other $c$ tested were sub-optimal.

**Algorithm 1:** UCB1-Tuned algorithm adapted for pooling

Assess one document from each run;
**Loop**
  Assess one doc from run $s$ that maximizes...
  $\mu_s + \sqrt{\frac{\ln n}{n_s} \cdot min(1/4, \sigma_s^2 + \sqrt{\frac{2 \ln n}{n_s}})}$

the evidence supporting these. In this section we describe some Bayesian allocation methods and how we have adapted them for our relevance assessment task.

A Bayesian perspective allows the uncertainty associated to the probabilities of winning to be dealt with formally. Each machine is characterized by its probability of winning (probability of supplying a relevant document in our case). From a Bayesian view, this probability can be regarded as a parameter endowed with a prior distribution. The process begins by assuming complete ignorance of the probabilities and, thus, assigning a uniform prior, $\mathcal{U}(0, 1)$, to each machine. This is equivalent to starting with $Beta(1, 1)$ for all machines (the Uniform distribution is a particular case of $Beta(\alpha, \beta)$ when both $\alpha$ and $\beta$ are set to 1). Every time that we judge a document we obtain evidence that is used to revise our beliefs about the probability distribution. In our case, the result of playing the machine is the binary relevance of the judged document ($O \in \{0, 1\}$), which can be seen as a Bernoulli random variable or, equivalently, Binomial with a single trial. This is an algebraic convenience because Beta is the conjugate prior distribution for Binomial. This gives a closed-form expression for the posterior distribution: $Beta(\alpha + O, \beta + 1 - O)$. Bayesian inference therefore supplies a natural and formal framework where the Beta distributions of the machines are updated as we get judgments from the assessors.

Every time we have a win (Bernoulli trial equals 1) $\alpha$ is increased and $\beta$ remains unchanged. In practice, this moves some probability mass to the right end of the interval $[0, 1]$ (i.e. we tend to skew the distribution to the left). Conversely, a loss (Bernoulli trial equals 0) leaves $\alpha$ unchanged and increases $\beta$, moving some probability mass to the left end of the interval $[0, 1]$ (i.e. we tend to skew the distribution to the right). At any given point, the Beta distribution is asymmetric when $\alpha$ and $\beta$ are different. If $\alpha > \beta$ the distribution is skewed to the left (more probability mass concentrated on the right). Otherwise, it is skewed to the right (more probability mass concentrated on the left). At the early stages of the process, both $\alpha$ and $\beta$ are low and the Beta distribution tends to be flat. As we see more evidence, $\alpha$ and $\beta$ tend to grow, making the Beta distribution more peaky. This models how we increasingly reduce uncertainty about the quality of the machine.

Bayesian Learning Automaton (BLA) [19] is an algorithm that follows this Bayesian approach and does random sampling from the posterior distributions to choose the next machine. A sample is drawn from each machine's distribution and the machine supplying the largest sample is the one selected for the next play. Those machines that have supplied many wins will have their Beta distribution skewed to the left and, at sampling time, they have more chance of supplying a large value. The idea of sampling from the posterior distribution dates back to Thompson [46] and is an effective and efficient heuristic for addressing the exploration/exploitation dilemma [12]. Furthermore, Granmo [19] has shown that BLA is self-correcting –the more the estimate of a machine's mean falls below the true value, the higher the probability of selecting machine $m$– and converges to pulling the optimal machine. A formal analysis and an empirical evaluation of Thompson Sampling for the multi-armed bandit problem can be found in [12, 2].

The adaptation of BLA for pooling is shown in Algorithm 2. BLA has no parameters and is known to be more effective than UCB or $\epsilon_n$-greedy [19]. We therefore implemented BLA and included it into our comparative study. We also implemented a further Bayesian method where the next machine is chosen by taking the maximum expectation of the posterior distributions. This approach will be referred to as $MM$ (MaxMean) and is sketched in Algorithm 3. Observe that the expectation of a distribution $Beta(\alpha, \beta)$ is $\alpha/(\alpha + \beta)$.

In our assessment task, we can take further advantage from the observed evidence. Given a binary judgment, we can update not only the Beta distribution of the run from which the document was extracted, but also the Beta distributions of all runs that retrieved the same document. In this way, evidence about relevance quickly spreads across all rankings[3]. Figure 1 shows an example of the process for a case with three runs. The documents marked

[3]With $MM$, this often leads to several run having the maximum mean. Ties are resolved by staying at the run from which the document was

**Algorithm 2:** Bayesian Learning Automaton for pooling

**foreach** $s \in$ *runs* **do**
  $\alpha_s \leftarrow 1, \beta_s \leftarrow 1$;

**Loop**
  **foreach** $s \in$ *runs* **do**
    Draw a sample $x_s$ from $Beta(\alpha_s, \beta_s)$;
  $next\_run \leftarrow \arg\max_s x_s$ ;
  Assess one document from $next\_run$ ($O_{next\_run}$: binary relevance of the asssesed document) ;
  $\alpha_{next\_run} \leftarrow \alpha_{next\_run} + O_{next\_run}$ ;
  $\beta_{next\_run} \leftarrow \beta_{next\_run} + 1 - O_{next\_run}$ ;

**Algorithm 3:** Maximum Mean for pooling

**foreach** $s \in$ *runs* **do**
  $\alpha_s \leftarrow 1, \beta_s \leftarrow 1$;

**Loop**
  **foreach** $s \in$ *runs* **do**
    $mean_s \leftarrow \alpha_s/(\alpha_s + \beta_s)$;
  $next\_run \leftarrow \arg\max_s mean_s$ ;
  Assess one document from $next\_run$ ($O_{next\_run}$: binary relevance of the asssesed document) ;
  $\alpha_{next\_run} \leftarrow \alpha_{next\_run} + O_{next\_run}$ ;
  $\beta_{next\_run} \leftarrow \beta_{next\_run} + 1 - O_{next\_run}$ ;

with an asterisk are those deemed as relevant.

## 5. Existing Pooling-based Methods for Adjudicating Judgments

In the literature of pooling-based evaluation, several methods have been employed to order the documents in the pool:

**DocID**. This is the standard sequence of assessments followed by TREC [48]. The set of unique documents in the pool is sorted by document identifier. This simple approach ignores any existing evidence about the estimated relevance of the pooled documents.

**Rank**. Pooled documents are chosen in decreasing order of rank. Top 1 documents (union of top retrieved documents from all runs) go first, top 2 documents go next, and so forth.

**MoveToFront (MTF)** [15]. The MTF method maintains a priority score for each run. Initially, all priorities are uniform. A maximum priority run is randomly selected and its top-ranked document is judged (documents already judged are skipped). If the judged document is relevant then documents from the same run continue to be judged until a non-relevant document is found. After seeing a non-relevant document, MTF reduces the priority of the current run and randomly jumps to another maximum priority run.

Moffat and colleagues [29] proposed several methods for ordering assessments. All their methods depend on rank-biased precision (RBP) scores. RBP [30] is an effectiveness metric that has been used for comparing search systems. It approximates user behavior by weighting the utility of a document based on the likelihood that a user reaches the rank position of the document. The underlying assumption is that users examine documents in order and there is a likelihood of 50% or so of getting to the 4th-ranked document. Under the standard RBP setting, the top

extracted.

| $run_1$ | $run_2$ | $run_3$ |
|---|---|---|
| $d_{47}*$ | $d_{53}*$ | $d_{80}$ |
| $d_{53}*$ | $d_{69}$ | $d_{44}$ |
| $d_{14}*$ | $d_{48}$ | $d_{56}$ |

Sequence of steps:

1. $\alpha_1, \beta_1, \alpha_2, \beta_2, \alpha_3, \beta_3 \leftarrow 1$
2. $mean_1, mean_2, mean_3 \leftarrow 1/2$
3. $next\_machine \leftarrow 2$ (random choice, all means are the same; let us assume that $run_2$ is selected)
4. $O_{next\_machine} \leftarrow 1$ ($d_{53}$ is assessed. It is relevant)
5. $\alpha_2 \leftarrow 2, \beta_2 \leftarrow 1$; $\alpha_1 \leftarrow 2, \beta_1 \leftarrow 1$ (updates all runs that retrieved $d_{53}$)
6. $mean_1, mean_2 \leftarrow 2/3$ ; $mean_3 \leftarrow 1/2$
7. $next\_machine \leftarrow 2$ (tie beween $run_1$ and $run_2$. It is resolved by staying at $run_2$)
8. $O_{next\_machine} \leftarrow 0$ ($d_{69}$ is assessed. It is non-relevant)
9. $\alpha_2 \leftarrow 2, \beta_2 \leftarrow 2$ (only $run_2$ retrieved $d_{69}$)
10. $mean_1 \leftarrow 2/3$ ; $mean_2, mean_3 \leftarrow 1/2$
11. $next\_machine \leftarrow 1$
12. $O_{next\_machine} \leftarrow 1$ ($d_{47}$ is assessed. It is relevant)
13. $\alpha_1 \leftarrow 3, \beta_1 \leftarrow 1$ (only $run_1$ retrieved $d_{47}$)
14. $mean_1 \leftarrow 3/4$ ; $mean_2, mean_3 \leftarrow 1/2$
15. ...

Figure 1: MaxMean (MM) algorithm adapted for pooling. Example of the sequence of steps taken for a case with three runs and pool depth equal to 3.

retrieved document is always examined, and the user proceeds from each document to the next with probability $p$, or terminates his search with probability $1 - p$. The overall utility of a ranking is defined as:

$$RBP = (1 - p) \cdot \sum_{i=1} u_i \cdot p^{i-1} \tag{6}$$

where $u_i \in [0, 1]$ is the relevance degree of document at rank $i$. It can be shown that $1/(1 - p)$ is the average number of documents examined. It is standard practice to set $p$ to $0.8$. This leads to $5$ documents examined on average, which is a reasonable model of actual user behavior.

With binary relevance, a relevant document at rank $1$ adds 0.2 to the overall ranking score, a relevant document at rank $2$ adds 0.16, and so forth. In [29], these rank-based contributions were used for adjudicating documents for judgment. The following variants were proposed:

**Moffat et al.'s Method (A)** [29], "*Summing contributions*". Given a document $d$ and a set of runs, the document is assigned a score ($w_d$) as follows:

$$w_d = \sum_{s \in runs} c_{s,d} \tag{7}$$

$$c_{s,d} = (1 - p) \cdot p^{r_{d,s}-1} \tag{8}$$

where $r_{d,s}$ is the document's rank in the run and $c_{s,d}$ is the potential contribution of the document to the RBP score of the run (if $d$ is relevant it would add $c_{s,d}$ to the RBP of $s$)[4].

The value of $w_d$ is computed for all documents in the pool and, next, documents are judged in decreasing order of $w_d$. This method favors documents that are highly ranked by many runs. This is a *static* method because the outcome

[4]If $d$ is not retrieved by $s$ then $r_{d,s} = \infty$ and, thus, $c_{s,d} = 0$.

of the judgments does not alter the ordering. The sequence of judgments only depends on the ranks of the documents in the runs.

**Moffat et al.'s Method (B)** [29], "*Weighting by residual*". This is an evolution over the previous method that gives more importance to documents coming from runs with many unjudged documents. The uncertainty about the quality of a run with many unjudged documents is high and method B tries to reduce this uncertainty by sampling more documents from this type of run. This is encoded by promoting runs whose *residual* RBP is large. At any moment, the *base* RBP of a run is the RBP that the run has achieved so far (computed from the documents that have been judged). The *residual* RBP of the run is the maximum increment in RBP that the run could achieve (if all remaining unjudged documents are relevant). As we judge documents from the run, its residual is reduced. Given a document $d$ and a set of runs $S$, the document weight is now defined as:

$$w_d = \sum_{s \in runs} c_{s,d} \cdot res_s \tag{9}$$

where $res_s$ is the run's residual.

**Moffat et al.'s Method (C)** [29], "*Weighting by predicted score*". By concentrating on runs with large residuals, method B runs the risk of giving too much weight to ineffective runs. Method C attempts to remedy this by promoting documents retrieved by effective runs. The document weight is defined as:

$$w_d = \sum_{s \in runs} c_{s,d} \cdot res_s \cdot (base_s + res_s/2)^3 \tag{10}$$

where $base_s$ is the RBP obtained by the run so far (computed from the documents that have been judged). The final RBP will be in the range $[base_s, base_s + res_s]$ and Method C takes this range's midpoint, $base_s + res_s/2$, as an estimation of the effectiveness of the run. In [29], this estimation was raised to the power of 3 to boost the strength of this factor.

## 6. Metasearch Methods for Adjudicating Judgments

This section discusses two methods that were originally designed for metasearch. However, they can be also employed for adjudicating judgments in the context of a pooling-based evaluation. Given a user query, a metasearch engine sends it to various search engines, collects their results, and combines them with the aim of improving the effectiveness of the ranking. These *distributed search* tools must carry out three main tasks: i) resource representation, which consists of generating a description of each information resource (search engine), ii) resource selection, which consists of selecting one or more search engines based on the query and the descriptions of the search engines, and iii) result merging, which consists of merging the ranked lists of documents from the searches carried out at every search engine. This last component, result merging, is directly related to the challenge of fusing the ranked lists (runs) in pooling-based evaluation. We therefore experimented with the following result merging methods, which were originally designed for metasearch:

**Borda Fuse**: Borda Fuse [3] is a metasearch model based on Borda Count. Borda Count is a traditional voting procedure that has been repeatedly applied for data fusion and metasearch problems. Each voter ranks a given set of $c$ candidates in order of preference. The top ranked candidate gets $c$ points, the 2nd ranked candidate gets $c-1$ points, and so forth[5]. Each candidate is finally assigned an overall voting score, which is the sum of points obtained from each voter, and the candidates are ranked in decreasing order of this score.

This voting-based model can be naturally applied for metasearch: the voters are the search engines and the retrieved documents are the candidates. The resulting Borda Fuse method is a simple and efficient combination algorithm that requires no relevance scores and no training data [3]. Furthermore, it performs quite well at combining the ranked lists returned by multiple retrieval algorithms [3]. Other effective combination methods require either retrieval scores –e.g. CombMNZ– or training data –e.g. Weighted Borda Fuse, which needs training queries to weight the candidates– making them unsuitable for adjudicating judgments in a pooling environment.

[5] Those candidates left unranked by the voter –if any– get an uniform portion of the remaining points.

**Hedge**: Hedge is an online learning algorithm [18] for combining expert advice. It was employed by Aslam and his colleagues [4] for simultaneously solving the problems of metasearch and pooling. The algorithm works as follows: Given a set of rankings produced by multiple retrieval algorithms, we start by selecting some document that is highly ranked by many systems. If the document is relevant we begin to trust systems that returned the document at a high position in the ranking and to lose faith in systems that did not return the document at a high position in the ranking. Conversely, if the document is not relevant we begin to lose faith in systems that returned the document at a high position and start to place trust in systems that did not return the document at a high position. In subsequent rounds, we would likely choose documents that are ranked at a high position by trusted systems. The specifics of how to apply Hedge for adjudicating judgments are reported in Algorithm 4. The weight $w_s$ encodes our faith in the performance of run $s$. In the absence of any prior knowledge, the algorithm begins with uniform weights and probabilities for each run. Essentially, the algorithm selects likely relevant documents based on the run's weights and the documents' ranks. In each judgment round, the relevance outcome ($rel_{d_{max}}$) and the position of the assessed document in each ranking ($r_{d_{max},s}$) determines the loss assigned to each run ($loss_s$). After each iteration, the weights and probabilities are adjusted. The algorithm is parameterized by a learning rate parameter ($\beta \in [0, 1]$)[6].

## 7. Experiments

We planned and performed a thorough series of experiments to evaluate the document adjudication methods discussed above. The experiments are fully reproducible. The code was developed in R and all instructions and functions can be accessed through our institutional website[7].

### *7.1. Collections*

Table 1 presents the main statistics of the four test collections that we used for experimentation. All of these are TREC collections associated with the ad-hoc retrieval task. This task has been at the heart of TREC since the early nineties. It is a standard search task where participants are given a set of queries (created by members of the TREC document assessment team) and a collection of documents and they are asked to rank documents from the collection for every query using their systems.

TREC defined two main classes of run: automatic and manual. Automatic runs are those runs produced by the participating systems where no manual intervention took place between the submission of the topics to the systems and the outputting of the run. Manual runs are those runs produced by the participating systems where any amount of human intervention in the generation of the results was allowed. For instance, a manual run can result from concatenating the best results from multiple (reformulated) queries.

In TREC, it is common to include manual runs in the pool. Such runs, where humans can reformulate queries and merge results, generally contribute many unique relevant documents to the pool. Combining automatic runs and manual runs leads to more diversified qrels and the resulting test collection is more robust and reusable.

We obtained from TREC all runs, manual and automatic, which contributed to the pool and ran the assessment process on a query-by-query basis. A global approach, where ordering is applied across documents from all queries, has been applied in the past [29]. In a multiple-query evaluation, a global approach consists of building a single ranked list to prioritize all the required assessments from all queries. Essentially, this means that a ranking of query-document pairs is built and, next, the top ranked pairs are those that are judged for relevance. Globally managing all judgments can lead to further improvements in early finding relevant documents. The assessment process can quickly extract relevant documents by focusing on the most promising query-document pairs. However, different queries would receive different number of judgments. This approach biases the qrels towards queries that have many relevant documents. We have opted to apply the document adjudication process individually for each query and, therefore, all queries receive the same treatment. Applying bandit models for adjudicating judgments in a global way is out of the scope of this paper and will be the subject of future research.

[6] Following [4], $\beta$ was fixed to 0.1 in all our experiments.

[7] `http://tec.citius.usc.es/ir/code/pooling_bandits_ms.html`

**Algorithm 4:** Hedge Algorithm for Pooling

**foreach** $s \in$ *runs* **do**
  // set uniform weights & probabilities
  $w_s = 1, p_s = 1/|runs|$;
$JudgedDocs = \emptyset$;
$D =$ union of all docs retrieved by all runs;
// $D$ contains pooled and unpooled docs
// In TREC, it is typically the union of the top 1000 docs retrieved
$r_{max} = |D|$;
**foreach** $s \in$ *runs*, $d \in D$ **do**
  // Initially, we set all losses assuming that all docs are non-relevant
  **if** $d$ *was retrieved by* $s$ **then**
    $loss_{s,d} = \frac{1}{2} \cdot \log \frac{r_{max}}{r_{d,s}}$;
    // $r_{d,s}$ is the rank position of doc $d$ in run $s$
  **else**
    $t_s =$ # docs retrieved by $s$ ;
    $loss_{s,d} = avg\{\frac{1}{2} \cdot \ln \frac{r_{max}}{j}\}_{j:t_{s+1},...,r_{max}}$;
    // the doc gets the avg loss it would get if retrieved in positions $t_{s+1}, ..., r_{max}$
// Now, the judging process starts
**Loop**
  **foreach** $d \in D \setminus JudgedDocs$ **do**
    $WeightedAvgMixLoss_d = \sum_s p_s \cdot loss_{s,d}$
  $d_{max} = \underset{d \in D \setminus JudgedDocs}{\arg\max} \; WeightedAvgMixLoss_d$ ;
  // we judge $d_{max}$
  $rel_{d_{max}} =$ (binary) relevance of $d_{max}$ ;
  $JudgedDocs = JudgedDocs \cup \{d_{max}\}$;
  // next, we compute the loss associated to this doc
  **foreach** $s \in$ *runs* **do**
    **if** $d_{max}$ *was retrieved by* $s$ **then**
      $loss_s = \frac{1}{2} \cdot (-1)^{rel_{d_{max}}} \cdot \ln \frac{r_{max}}{r_{d_{max},s}}$;
      // $r_{d_{max},s}$ is the rank position of doc $d_{max}$ in run $s$
    **else**
      $t_s =$ # docs retrieved by $s$ ;
      $loss_s = avg\{\frac{1}{2} \cdot (-1)^{rel_{d_{max}}} \cdot \ln \frac{r_{max}}{j}\}_{j:t_{s+1},...,r_{max}}$;
      // the doc gets the avg loss it would get if retrieved in positions $t_{s+1}, ..., r_{max}$
  // update run weights & probabilities
  **foreach** $s \in$ *runs* **do**
    $w_s = w_s \cdot \beta^{loss_s}$ ;
    $p_s = w_s / \sum_j w_j$ ;

| | *TREC5* | *TREC6* | *TREC7* | *TREC8* |
|---|---|---|---|---|
| # queries | 50 | 50 | 50 | 50 |
| # pooled runs (automatic+manual) | 77+24 | 31+15 | 77+7 | 62+9 |
| # assessed docs | 133681 | 72270 | 80345 | 86830 |
| avg. # docs judged per query | 2673.6 | 1445.4 | 1606.9 | 1736.6 |
| % of relevant docs in the pool | 4.1% | 6.4% | 5.8% | 5.4% |
| avg. # rels per query | 110.48 | 92.22 | 93.48 | 94.56 |

Table 1: Main statistics of the test collections

### *7.2. Performance metric*

We analyze here the adjudication methods described in the previous sections in terms of their ability to early identify relevant documents. We computed the trends of relevant documents found as follows. For each query, each adjudication method induces a sequence of judgments of the pooled documents. Recall@n can be computed at any point ($n$) in this sequence. The official TREC qrels contain relevance judgments for all documents in the pool and, therefore, every time we need an assessment we can just go to the qrel file and access the required judgment. In this way, we can evaluate any subset pooling strategy and analyze the evolution of Recall@n. The main evaluation metric is therefore Recall@n averaged over the set of queries in the test collection.

First, we experimented with the six adjudication methods discussed in section 5, and the two metasearch methods discussed in section 6. These experiments, reported next, concluded with the selection of two prominent methods as our reference baseline methods. The subsequent experiments, reported in section 7.4, evaluated all the bandit-based methods and compared them against the two reference baselines.

### *7.3. Pooling Baselines*

Figure 2 depicts the evolution –at varying judgment levels– of Recall@n achieved by the six adjudication methods presented in section 5. MTF is the best performing method. It is remarkably superior to all other baseline methods. DocID is the worst performing method. This is as expected, since DocID is not concerned about early identifying relevant documents. Our comparison shows also that method C is superior to both method A and method B. This confirms the results obtained by Moffat et al. [29]. To the best of our knowledge, MTF was never compared to the methods A, B and C defined by Moffat et al, and our experimental study demonstrates that MTF is superior to these three methods. It is important to observe, though, that Moffat et al. aimed at optimizing the uncertainty in the RBP-based ranking of retrieval algorithms and, thus, their methods are not necessarily optimal at identifying relevant documents under a fixed budget.

Figure 3 plots the recall achieved by the two metasearch methods. For the sake of comparison, we have also included the best performing pooling-based method (MTF) into this plot. These experiments show a consistent ordering among these three strategies. Hedge is the best performing method; followed by MTF, which is superior to Borda. Overall, Hedge and MTF are capable of early finding more relevant documents than those found by the six other methods evaluated. We therefore adopted both Hedge and MTF as our reference baseline methods for our further experiments and analyses.

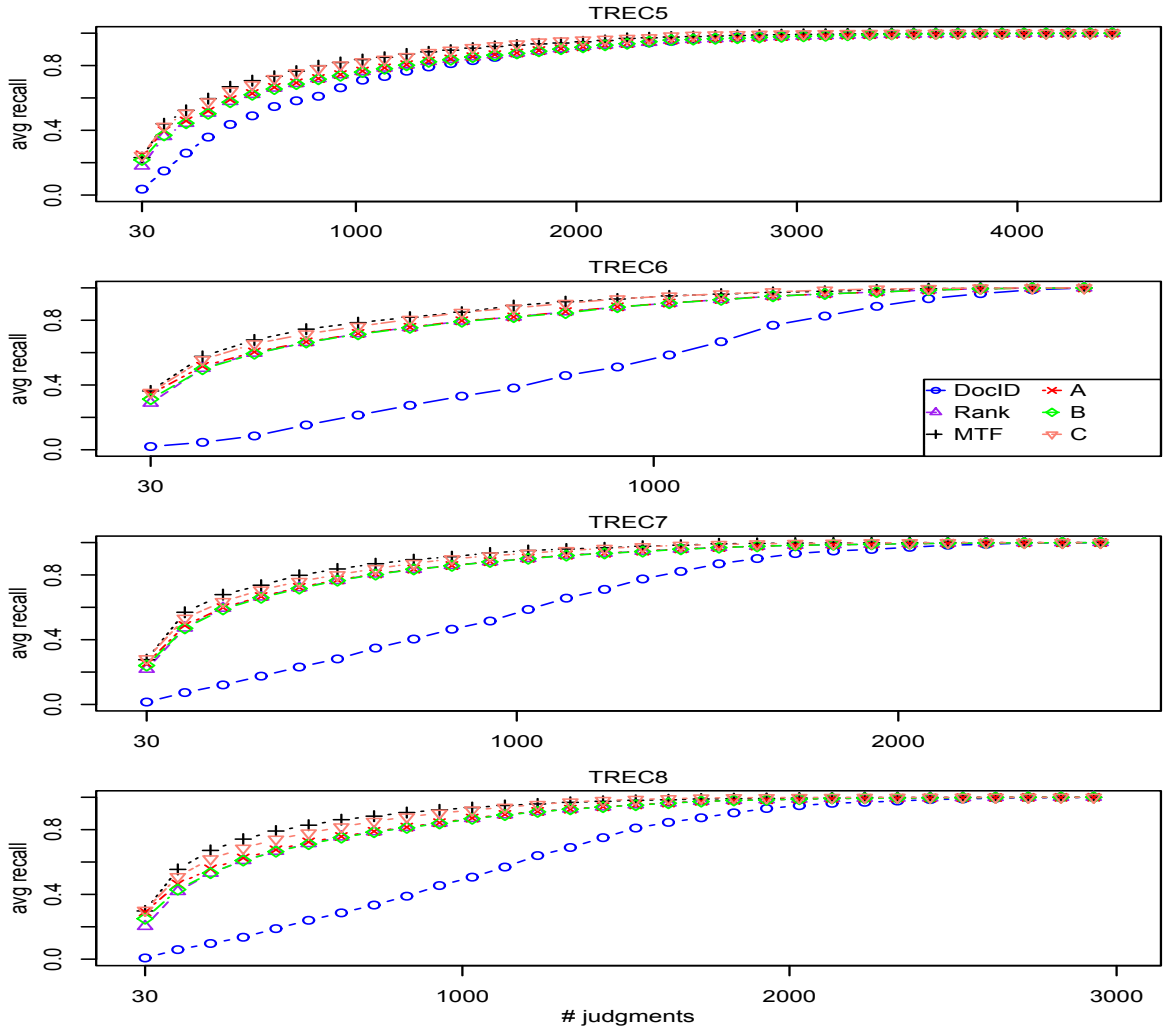


Figure 2: Comparison of methods for adjudicating judgments: DocID, Rank, MoveToFront (MTF), and Moffat et al.'s methods A, B and C. The graph plots average recall against the number of judgments.

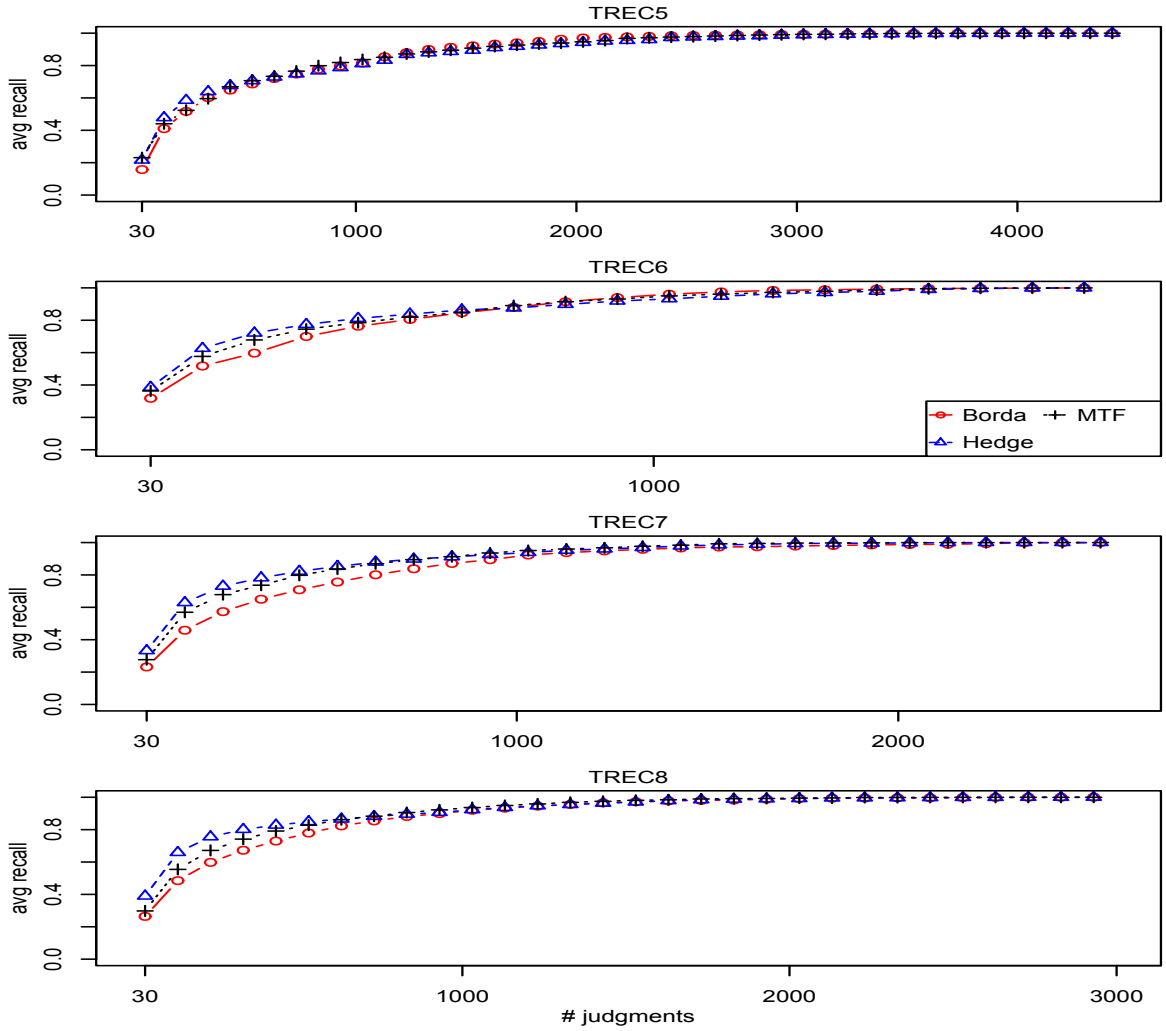


Figure 3: Comparison of methods for adjudicating judgments: Borda, Hedge and MoveToFront (MTF). The graph plots average recall against the number of judgments.

### 7.4. Bandit-based models

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | ***TREC5*** | | | | |
| | | | | *Number of judgments* | | | | |
| | 30 | 100 | 300 | 500 | 700 | 900 | 1100 | 2000 |
| MTF | .2310 | .4033$^{\downarrow}$ | .5761$^{\downarrow}$ | .6945 | .7555 | .8118$^{\downarrow}$ | .8465$^{\downarrow}$ | .9436$^{\downarrow}$ |
| HEDGE | .2158 | **.4383** | **.6228** | .6977 | .7425$^{\downarrow}$ | .7819$^{\downarrow}$ | .8251 | .9411$^{\downarrow}$ |
| BLA | .2312 | .3861$^{\downarrow}$ | .5809$^{\downarrow}$ | .6768$^{\downarrow}$ | .7567$^{\downarrow}$ | .8101 | .8463 | .9502$^{\downarrow}$ |
| MM | .2157 | .4116$^{\downarrow}$ | .6170 | **.7191** | **.7890** | .8326 | .8609 | **.9640** |
| RANDOM | .1947$^{\downarrow}$ | .3613$^{\downarrow}$ | .5817 | .6803 | .7570 | .8164$^{\downarrow}$ | .8516$^{\downarrow}$ | .9469$^{\downarrow}$ |
| UCB | **.2380** | .3674$^{\downarrow}$ | .6067 | .6956 | .7785 | .8292$^{\downarrow}$ | .8616 | .9494$^{\downarrow}$ |
| $\epsilon_n$-GREEDY | .2136$^{\downarrow}$ | .3616$^{\downarrow}$ | .6084 | .7044 | .7839 | **.8389** | **.8666** | .9510$^{\downarrow}$ |
| | | | | ***TREC6*** | | | | |
| | | | | *Number of judgments* | | | | |
| | 30 | 100 | 300 | 500 | 700 | 900 | 1100 | |
| MTF | .3633$^{\downarrow}$ | .5333$^{\downarrow}$ | .7302$^{\downarrow}$ | .8102$^{\downarrow}$ | .8783$^{\downarrow}$ | .9250$^{\downarrow}$ | .9585$^{\downarrow}$ | |
| HEDGE | **.3852** | **.5850** | **.7629** | .8343 | .8721$^{\downarrow}$ | .9118$^{\downarrow}$ | .9426$^{\downarrow}$ | |
| BLA | .2933$^{\downarrow}$ | .4793$^{\downarrow}$ | .6870$^{\downarrow}$ | .7975$^{\downarrow}$ | .8632$^{\downarrow}$ | .9143$^{\downarrow}$ | .9505$^{\downarrow}$ | |
| MM | .3555$^{\downarrow}$ | .5410$^{\downarrow}$ | .7354 | **.8401** | **.8960** | **.9423** | **.9649** | |
| RANDOM | .2979$^{\downarrow}$ | .4894$^{\downarrow}$ | .6971$^{\downarrow}$ | .8003$^{\downarrow}$ | .8623$^{\downarrow}$ | .9137$^{\downarrow}$ | .9523$^{\downarrow}$ | |
| UCB | .2960$^{\downarrow}$ | .5108$^{\downarrow}$ | .7039$^{\downarrow}$ | .8168$^{\downarrow}$ | .8687$^{\downarrow}$ | .9174$^{\downarrow}$ | .9563$^{\downarrow}$ | |
| $\epsilon_n$-GREEDY | .3092$^{\downarrow}$ | .5067$^{\downarrow}$ | .7296 | .8191$^{\downarrow}$ | .8740$^{\downarrow}$ | .9230$^{\downarrow}$ | .9541$^{\downarrow}$ | |
| | | | | ***TREC7*** | | | | |
| | | | | *Number of judgments* | | | | |
| | 30 | 100 | 300 | 500 | 700 | 900 | 1100 | 2000 |
| MTF | .2767$^{\downarrow}$ | .5134$^{\downarrow}$ | .7191$^{\downarrow}$ | .8264$^{\downarrow}$ | .8868 | .9265 | **.9580** | **.9983** |
| HEDGE | **.3319** | **.5826** | **.7678** | **.8467** | .8923 | .9214$^{\downarrow}$ | .9480$^{\downarrow}$ | .9982 |
| BLA | .2460$^{\downarrow}$ | .4687$^{\downarrow}$ | .6874$^{\downarrow}$ | .7907$^{\downarrow}$ | .8578$^{\downarrow}$ | .9069$^{\downarrow}$ | .9419 | .9915 |
| MM | .2930$^{\downarrow}$ | .5062$^{\downarrow}$ | .7272$^{\downarrow}$ | .8270 | **.8970** | **.9380** | .9574 | .9982 |
| RANDOM | .2313$^{\downarrow}$ | .4538$^{\downarrow}$ | .6869$^{\downarrow}$ | .7922$^{\downarrow}$ | .8606$^{\downarrow}$ | .9139$^{\downarrow}$ | .9471 | .9909 |
| UCB | .2403$^{\downarrow}$ | .4703$^{\downarrow}$ | .7035$^{\downarrow}$ | .8039$^{\downarrow}$ | .8675$^{\downarrow}$ | .9187$^{\downarrow}$ | .9446 | .9938 |
| $\epsilon_n$-GREEDY | .2197$^{\downarrow}$ | .4433$^{\downarrow}$ | .7119$^{\downarrow}$ | .8035$^{\downarrow}$ | .8746$^{\downarrow}$ | .9167$^{\downarrow}$ | .9425 | .9961 |
| | | | | ***TREC8*** | | | | |
| | | | | *Number of judgments* | | | | |
| | 30 | 100 | 300 | 500 | 700 | 900 | 1100 | 2000 |
| MTF | .2978$^{\downarrow}$ | .5041$^{\downarrow}$ | .7220$^{\downarrow}$ | .8166$^{\downarrow}$ | .8763$^{\downarrow}$ | .9172$^{\downarrow}$ | .9460$^{\downarrow}$ | .9944 |
| HEDGE | **.3894** | **.6087** | **.7883** | .8450 | .8755$^{\downarrow}$ | .9039$^{\downarrow}$ | .9334$^{\downarrow}$ | .9914$^{\downarrow}$ |
| BLA | .2526$^{\downarrow}$ | .4577$^{\downarrow}$ | .6806$^{\downarrow}$ | .7943$^{\downarrow}$ | .8587$^{\downarrow}$ | .9015$^{\downarrow}$ | .9332$^{\downarrow}$ | .9959 |
| MM | .3288$^{\downarrow}$ | .5427$^{\downarrow}$ | .7509$^{\downarrow}$ | **.8470** | **.8969** | **.9380** | **.9571** | **.9963** |
| RANDOM | .2524$^{\downarrow}$ | .4442$^{\downarrow}$ | .6753$^{\downarrow}$ | .7833$^{\downarrow}$ | .8500$^{\downarrow}$ | .8964$^{\downarrow}$ | .9276$^{\downarrow}$ | .9926 |
| UCB | .2545$^{\downarrow}$ | .4931$^{\downarrow}$ | .6943$^{\downarrow}$ | .8030$^{\downarrow}$ | .8618$^{\downarrow}$ | .9037$^{\downarrow}$ | .9332$^{\downarrow}$ | .9937 |
| $\epsilon_n$-GREEDY | .2547$^{\downarrow}$ | .4680$^{\downarrow}$ | .6968$^{\downarrow}$ | .8036$^{\downarrow}$ | .8610$^{\downarrow}$ | .9023$^{\downarrow}$ | .9263$^{\downarrow}$ | .9907 |

Table 2: Bandit Allocation Strategies against two baselines (MTF and HEDGE). Average recall at different number of judgments performed. For each judgment level and collection, the highest average is bolded. The symbol $\downarrow$ indicates a significant decrease over the corresponding best average (one-sided paired t-test, $\alpha = .05$).

Let us now evaluate the performance of the innovative methods based on multi-armed bandits. Table 2 compares the recall achieved by MTF and Hedge against the recall achieved by the bandit allocation methods described in section 4. At the end of the assessment process, all pooled documents are judged and, therefore, all strategies get to *perfect* recall (all relevant documents in the pool have been identified). Nonetheless, some methods are much quicker than others at identifying relevant documents. Extracting relevant documents early permits judgment effort to be reduced. We can just stop the assessment process when a sufficient number of relevant documents have been judged.

The main conclusions that can be drawn from these experiments are:

- Randomly picking the next run to examine performs poorly. This naïve method ignores the quality of the runs and, therefore, it slowly extracts relevant documents. However, observe that the average recall achieved by this random method are not disproportionately low. The reason is that runs are chosen randomly, but documents are not randomly sampled from the selected run. The order of the documents' ranks is preserved and, therefore, the randomly selected run supplies the top ranked unjudged document. The runs are therefore examined in a top-down fashion, facilitating the early identification of relevant documents.

- BLA, UCB, and $\epsilon_n$-GREEDY encode different ways to promote exploration, but their performance in our experiments is roughly the same. BLA and UCB are sophisticated allocation methods that consider the outcomes

of past plays and the uncertainty about current estimations. The exploration movements are influenced by the confidence intervals of the reward averages (UCB) or by how peaky the posterior distributions are (BLA). $\epsilon_n$-GREEDY is simpler than UCB and BLA: when it wants to explore it jumps to a random run. Nevertheless $\epsilon_n$-GREEDY's performance is not inferior to the performance achieved by UCB and BLA. This suggests that no form of educated exploration is required. Note also that the quality of the runs varies as we extract documents from them. By giving too much credit to past rewards we run the risk of always exploring towards runs with a good track of judged documents. Past performance, however, is not necessarily indicative of future results. In this respect, random exploration ignores the history and gives equal opportunities to all runs.

- Hedge, MM and MTF are the best performing adjudication methods. Hedge identifies many relevant items at the beginning of the judgment process (first 300 documents assessed) but it performs worse towards the end. At 30 or 100 judgments, Hedge is usually the best performing method and the improvements over the competing methods are often statistically significant. However, after 500 judgments, MM is either the best performing method (TREC5, TREC6 and TREC8) or it is not statistically different from the best (TREC7). Moreover, the improvements over Hedge achieved by MM are often statistically significant. On the other hand, MTF looks inferior to MM: in the few cases where MTF yielded a Recall@n value higher than the Recall@n yielded by MM the difference was not statistically significant.

Overall, we can draw two main conclusions from these experiments. First, if we are interested in judging a small number of documents then we should definitely go for Hedge. Second, if we can afford a more thorough assessment process then we should probably prefer MM. When we can judge more than 500 documents, chances are that MM will be a more profitable model. In such cases, our empirical evidence suggests that MM is at least as effective as the best model in the literature and, in actual fact, in many cases it is superior to Hedge in a statistically significant way.

### *7.5. Non-stationary bandit models*

Under MTF, the last document judged is the only evidence considered. As long as it is relevant, we remain extracting and judging documents from the same run. Otherwise, we decrease the priority of the current run and jump to another run. Forgetting quickly about previously judged documents can be also incorporated into the Bayesian bandit methods. This is the topic of this section.

In stationary environments, the (unknown) probability of winning does not change, and all previous rewards should receive the same treatment. In non-stationary environments, a uniform treatment of the history is deficient. If bandits change over time we need to react by weighting recent and old rewards in a non-uniform way (e.g., by weighting recent rewards more heavily than long-past ones [44]). In our assessment task, the environment is clearly non-stationary. The probability of supplying a relevant document changes as we move down in the rankings. Not only the probabilities associated to the runs change, so do the relative merits of different runs. For instance, a run produced by a precision-oriented search system may be initially preferable to a run produced by a recall-oriented search system but, after judging the top retrieved documents, we may be safer moving to the recall-oriented run.

One popular way of reacting to changes in non-stationary problems is to incorporate a rate parameter to tune the influence of recent rewards and old rewards. The Bayesian methods BLA and MM can be naturally adapted as follows. At any moment, the parameters of the posterior distribution of a run $s$ are:

$$\alpha_s = 1 + jrel_s \tag{11}$$

$$\beta_s = 1 + jret_s - jrel_s \tag{12}$$

where $jrel_s$ is the number of documents retrieved by $s$ and judged as relevant, and $jret_s$ is the number of documents retrieved by $s$ that have been assessed so far. The update of $jrel_s$ and $jret_s$ can be driven by a rate parameter that motivates the method to learn changing environments [16]. Given the last judgment, $rel_d \in \{0, 1\}$, the parameters of the runs retrieving this document are updated as:

$$jrel_s \leftarrow rate \cdot jrel_s + rel_d \tag{13}$$

$$jret_s \leftarrow rate \cdot jret_s + 1 \tag{14}$$

If $rate = 1$ we get the stationary case, where all rewards count the same. If $rate > 1$ we give more weight to early relevant documents. Conversely, if $rate < 1$ we give more weight to the last judgment. We experimented with different values of $rate$ and found that $rate = 0$ was the best performing setting. This particular instance of the non-stationary case implements a notion of history –only the last judgment counts– that is equivalent to MTF's. Updating the distributions in this way leads to new Bayesian methods, which will be referred to as BLA-NS and MM-NS. After updating the distribution's parameters, BLA-NS chooses the next run by sampling from the posterior distribution. MM-NS simply selects the posterior distribution with the largest mean. Note that fixing $rate$ to 0 preserves the formality of the model. The outcome of every trial is still Bernoulli and the prior/posterior are still Beta distributions. Setting $rate$ to 0 can be actually seen as a re-initialization of the run's counts that happens before examining its next document.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ***TREC5*** | | | | | | | | |
| | *Number of judgments* | | | | | | | |
| | 30 | 100 | 300 | 500 | 700 | 900 | 1100 | 2000 |
| MTF | .2310 | .4033$^{\downarrow}$ | .5761$^{\downarrow}$ | .6945$^{\downarrow}$ | .7555$^{\downarrow}$ | .8118$^{\downarrow}$ | .8465$^{\downarrow}$ | .9436$^{\downarrow}$ |
| HEDGE | .2158 | **.4383** | .6228 | .6977 | .7425$^{\downarrow}$ | .7819$^{\downarrow}$ | .8251$^{\downarrow}$ | .9411$^{\downarrow}$ |
| BLA | **.2312** | .3861$^{\downarrow}$ | .5809$^{\downarrow}$ | .6768$^{\downarrow}$ | .7567$^{\downarrow}$ | .8101$^{\downarrow}$ | .8463$^{\downarrow}$ | .9502$^{\downarrow}$ |
| BLA-NS | .2024 | .3769$^{\downarrow}$ | .5747$^{\downarrow}$ | .6848$^{\downarrow}$ | .7631$^{\downarrow}$ | .8358$^{\downarrow}$ | .8745$^{\downarrow}$ | .9526$^{\downarrow}$ |
| MM | .2157 | .4116$^{\downarrow}$ | .6170 | .7191 | .7890 | .8326 | .8609$^{\downarrow}$ | .9640 |
| MM-NS | .2177 | .4206 | **.6401** | **.7418** | **.8187** | **.8621** | **.8962** | **.9685** |
| ***TREC6*** | | | | | | | | |
| | *Number of judgments* | | | | | | | |
| | 30 | 100 | 300 | 500 | 700 | 900 | 1100 | |
| MTF | .3633$^{\downarrow}$ | .5333$^{\downarrow}$ | .7302$^{\downarrow}$ | .8102$^{\downarrow}$ | .8783$^{\downarrow}$ | .9250$^{\downarrow}$ | .9585$^{\downarrow}$ | |
| HEDGE | **.3852** | **.5850** | **.7629** | .8343 | .8721$^{\downarrow}$ | .9118$^{\downarrow}$ | .9426$^{\downarrow}$ | |
| BLA | .2933$^{\downarrow}$ | .4793$^{\downarrow}$ | .6870$^{\downarrow}$ | .7975$^{\downarrow}$ | .8632$^{\downarrow}$ | .9143$^{\downarrow}$ | .9505$^{\downarrow}$ | |
| BLA-NS | .2890$^{\downarrow}$ | .4859$^{\downarrow}$ | .6944$^{\downarrow}$ | .7965$^{\downarrow}$ | .8575$^{\downarrow}$ | .9006$^{\downarrow}$ | .9324$^{\downarrow}$ | |
| MM | .3555$^{\downarrow}$ | .5410$^{\downarrow}$ | .7354 | .8401 | .8960 | **.9423** | .9649 | |
| MM-NS | .3573$^{\downarrow}$ | .5358$^{\downarrow}$ | .7515 | **.8491** | **.9001** | .9419 | **.9681** | |
| ***TREC7*** | | | | | | | | |
| | *Number of judgments* | | | | | | | |
| | 30 | 100 | 300 | 500 | 700 | 900 | 1100 | 2000 |
| MTF | .2767$^{\downarrow}$ | .5134$^{\downarrow}$ | .7191$^{\downarrow}$ | .8264$^{\downarrow}$ | .8868$^{\downarrow}$ | .9265$^{\downarrow}$ | .9580$^{\downarrow}$ | .9983 |
| HEDGE | **.3319** | **.5826** | **.7678** | **.8467** | .8923 | .9214$^{\downarrow}$ | .9480$^{\downarrow}$ | .9982$^{\downarrow}$ |
| BLA | .2460$^{\downarrow}$ | .4687$^{\downarrow}$ | .6874$^{\downarrow}$ | .7907$^{\downarrow}$ | .8578$^{\downarrow}$ | .9069$^{\downarrow}$ | .9419$^{\downarrow}$ | .9915 |
| BLA-NS | .2317$^{\downarrow}$ | .4538$^{\downarrow}$ | .6701$^{\downarrow}$ | .7780$^{\downarrow}$ | .8445$^{\downarrow}$ | .8833$^{\downarrow}$ | .9170$^{\downarrow}$ | .9884$^{\downarrow}$ |
| MM | .2930$^{\downarrow}$ | .5062$^{\downarrow}$ | .7272$^{\downarrow}$ | .8270 | .8970$^{\downarrow}$ | .9380 | .9574$^{\downarrow}$ | .9982$^{\downarrow}$ |
| MM-NS | .2829$^{\downarrow}$ | .5249$^{\downarrow}$ | .7461 | **.8467** | **.9085** | **.9437** | **.9671** | **.9992** |
| ***TREC8*** | | | | | | | | |
| | *Number of judgments* | | | | | | | |
| | 30 | 100 | 300 | 500 | 700 | 900 | 1100 | 2000 |
| MTF | .2978$^{\downarrow}$ | .5041$^{\downarrow}$ | .7220$^{\downarrow}$ | .8166$^{\downarrow}$ | .8763$^{\downarrow}$ | .9172$^{\downarrow}$ | .9460$^{\downarrow}$ | .9944 |
| HEDGE | **.3894** | **.6087** | **.7883** | .8450 | .8755$^{\downarrow}$ | .9039$^{\downarrow}$ | .9334$^{\downarrow}$ | .9914$^{\downarrow}$ |
| BLA | .2526$^{\downarrow}$ | .4577$^{\downarrow}$ | .6806$^{\downarrow}$ | .7943$^{\downarrow}$ | .8587$^{\downarrow}$ | .9015$^{\downarrow}$ | .9332$^{\downarrow}$ | .9959 |
| BLA-NS | .2545$^{\downarrow}$ | .4495$^{\downarrow}$ | .6540$^{\downarrow}$ | .7709$^{\downarrow}$ | .8407$^{\downarrow}$ | .8833$^{\downarrow}$ | .9141$^{\downarrow}$ | .9882$^{\downarrow}$ |
| MM | .3288$^{\downarrow}$ | .5427$^{\downarrow}$ | .7509$^{\downarrow}$ | .8470$^{\downarrow}$ | .8969 | .9380 | .9571 | **.9963** |
| MM-NS | .3120$^{\downarrow}$ | .5474$^{\downarrow}$ | .7645$^{\downarrow}$ | **.8591** | **.9026** | **.9382** | **.9608** | .9958 |

Table 3: MoveToFront (MTF), Hedge, Stationary Bayesian Allocation methods (BLA and MM), and Non-Stationary Bayesian Allocation methods (BLA-NS and MM-NS). Average recall at different number of judgments performed. For each judgment level and collection, the highest average is bolded. The symbol $^{\downarrow}$ indicates a significant decrease over the corresponding best average (one-sided paired t-test, $\alpha = .05$).

Table 3 reports the results obtained with these two alternatives against those achieved with MTF, Hedge, MM and BLA. MM-NS is often superior to MM. MM-NS is still inferior to Hedge at low judgment counts but, after 500 judgments, it is always the best performing approach. Furthermore, after 700 judgments, the improvements of MM-NS over Hedge are often statistical significant.

Both MTF and MM-NS ignore previous rewards but they entail different models of exploration. After a relevant document, both methods behave the same: they remain in the same run. The difference lies in the movement after judging a non-relevant document. The priority-based mechanism of MTF makes that all runs are visited once before we re-visit any run; similarly, a run is visited for the third time only when all other runs have been visited twice, and so forth. Under MM-NS, the runs that retrieved the judged non-relevant document are assigned $\alpha = 1$ and $\beta = 2$ and, thus, the mean of their posterior distributions goes to $1/3$. The runs that did not retrieve the document have their distributions unchanged, and their means can be either $1/3$ (last update of the run had been to account for a non-

relevant document), $2/3$ (last update of the run had been to account for a relevant document), or $1/2$ (the distribution of the run has no updates so far because all documents ranked by the run are unjudged). MM-NS selects a run with the highest mean and, therefore, it tends to move towards runs whose last update was to account for a relevant document (the largest mean is $2/3$). This appears to be an advantage of MM-NS over MTF.

## 8. Bias analysis

The experiments reported above demonstrate that Hedge, MTF and MM-NS are effective at early identifying relevant documents and, therefore, we may want to employ them to create qrels with reduced human effort. However, only judging a subset of the pool introduces a *bias* with respect to judging all pooled documents. It is therefore necessary to perform a bias analysis of the different subset pooling methods. This is precisely the aim of the analysis reported in this section.

A standard approach to measure the bias induced by subset pooling methods is to use the official system rankings as the basis for comparison. The official system rankings are computed from the qrel file created with the whole pool. This is a good gold standard because it is built from multiple retrieval algorithms explored up to some large depth (typically 100). By computing Kendall's $\tau$ correlation between the official ranking of the systems and the ranking of the systems under some subset pooling strategy[8] we can measure the bias of the sub-setting method. Furthermore, the bias can act as a guidance to estimate which number of judgments is required to have a sufficiently high correlation with respect to the official ranking.

Kendall's $\tau$ consists of computing the minimum number of pairwise adjacent swaps needed to turn one ranking into the other and normalizing by the number of items being ranked. Two identical rankings have a correlation of 1, the correlation between a ranking and its perfect inverse equals -1, and the expected correlation of two rankings chosen at random equals 0. Figure 4 plots the correlation between the official ranking of systems and the ranking of systems under Hedge, MTF and MM-NS (at different cutoffs of documents judged). This correlation analysis, which goes up to about 1000 documents judged, reveals that all these adjudication methods lead to high levels of correlation. The correlation is higher than 0.9, even with only 100 documents judged. Initially, Hedge's correlation is higher than MM-NS or MTF's correlation. As more judgments come in, MM-NS gets to higher levels of correlations when compared to MTF or Hedge (e.g. in TREC5 around 500 documents judged).

Table 4 provides additional data on the subset pooling methods and their biases. In this table we not only analyze Kendall's $\tau$, but also AP correlation ($\tau_{AP}$). AP correlation is a rank correlation coefficient proposed by Yilmaz and colleagues [52]. The main idea is that, when comparing two rankings of search systems, discrepancies among those systems having high positions are more important than those among systems having low positions. Kendall's $\tau$ equally penalizes errors both at high and low positions. AP correlation, instead, gives more penalties to errors at high positions. In IR evaluation, we are often concerned about correctly identifying the best search strategies (those at the top of the system ranking) and, therefore, AP correlation is well suited for evaluating subset pooling strategies. Table 4 shows the number of judgments required by each subset pooling method to achieve a level of correlation above 0.90, 0.95, or 0.99 (the table shows the counts for both correlation metrics). For the sake of comparison, we also report the number of judgments required by DOCID, which is the standard ordering approach followed by NIST when creating the official qrels. If we want to produce a highly reliable benchmark ($\tau, \tau_{AP} \geq .99$) then MM-NS should be preferred because it needs a substantially lower number of judgments. This stringent level of correlation (.99) is indicative of the point in the judgment process after which the influence of having more judgments becomes really negligible. With a few hundred judgments, MM-NS is able to create a qrel file that ranks systems very much like the full pool does. The counts of judgments reveal a remarkable gain over judging the entire pool. For instance, in TREC5, where the average pool size is 2692, MM-NS needs only 595 judgments to get to $\tau_{AP} \geq .99$. This means that we can save up a substantial fraction of the judgments (78%) and still produce a highly robust benchmark. If we can accept .95 or .90 correlation levels then Hedge and MM-NS require a similar number of judgments (but Hedge looks slightly better). These results confirm our previous findings: Hedge must be our subset pooling method when building shallow pools but MM-NS must be our method of choice when building deeper pools.

[8] Both rankings produced by computing the Mean Average Precision of the systems.

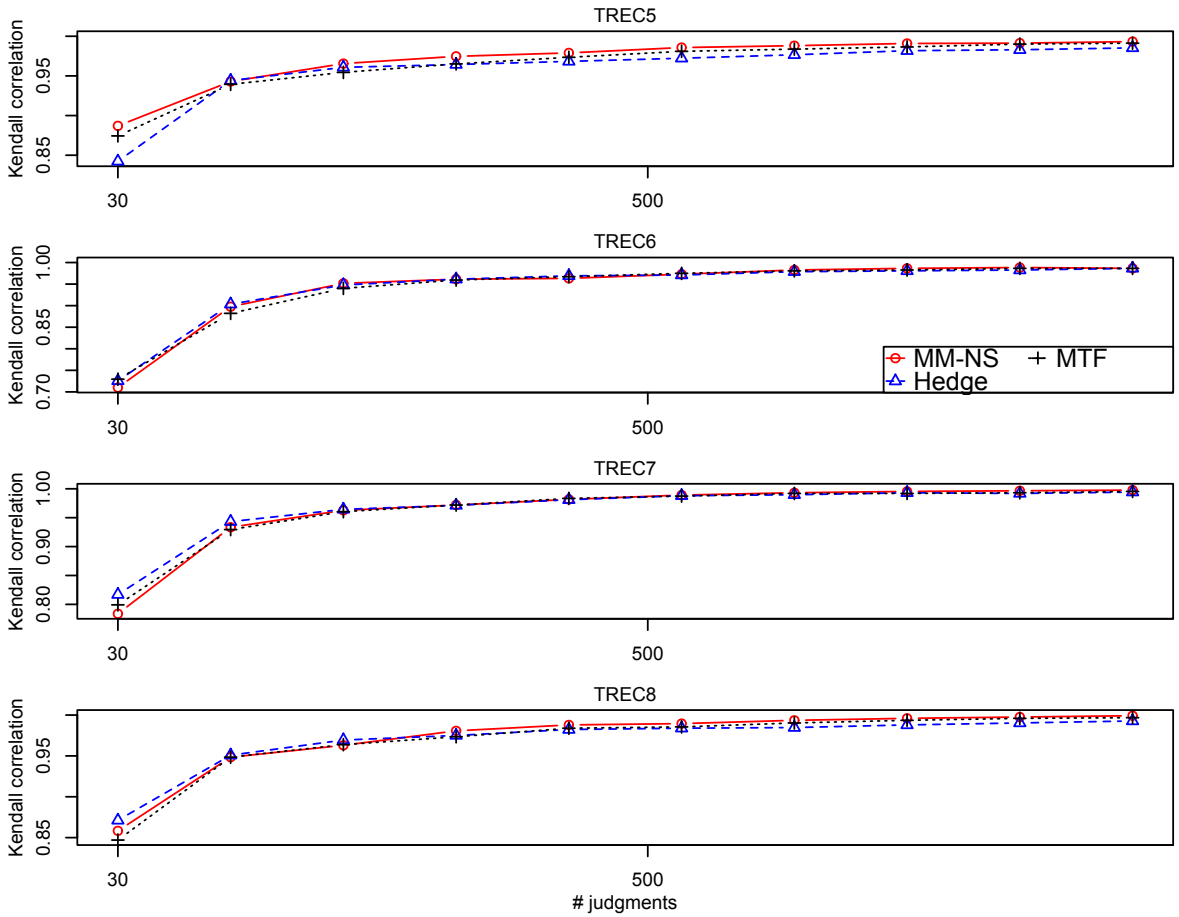


Figure 4: Kendall's $\tau$ correlation between the official ranking of systems and the ranking of systems built from Hedge, MM-NS and MTF at different number of documents judged.

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | ***TREC5*** | | | |
| | $\tau \geq .9$ | $\tau \geq .95$ | $\tau \geq .99$ | $\tau_{AP} \geq .9$ | $\tau_{AP} \geq .95$ | $\tau_{AP} \geq .99$ |
| DOCID | 1181 | 1442 | 2503 | 1181 | 1406 | 2563 |
| MTF | 45 | 200 | 827 | **33** | 162 | 708 |
| HEDGE | 55 | 171 | 1176 | 56 | 155 | 958 |
| MM-NS | **34** | **158** | **670** | 34 | **132** | **595** |
| | | | ***TREC6*** | | | |
| | $\tau \geq .9$ | $\tau \geq .95$ | $\tau \geq .99$ | $\tau_{AP} \geq .9$ | $\tau_{AP} \geq .95$ | $\tau_{AP} \geq .99$ |
| DOCID | 1054 | 1399 | 1726 | 1069 | 1396 | 1702 |
| MTF | 168 | 276 | 1500 | 130 | 260 | 1500 |
| HEDGE | **128** | **238** | 1555 | 136 | 224 | 1555 |
| MM-NS | 136 | 252 | **986** | **120** | **209** | **986** |
| | | | ***TREC7*** | | | |
| | $\tau \geq .9$ | $\tau \geq .95$ | $\tau \geq .99$ | $\tau_{AP} \geq .9$ | $\tau_{AP} \geq .95$ | $\tau_{AP} \geq .99$ |
| DOCID | 954 | 1312 | 2011 | 939 | 1313 | 2047 |
| MTF | 81 | 197 | 595 | 92 | 203 | 595 |
| HEDGE | **63** | **175** | 602 | **69** | **176** | 621 |
| MM-NS | 76 | 176 | **532** | 85 | 180 | **531** |
| | | | ***TREC8*** | | | |
| | $\tau \geq .9$ | $\tau \geq .95$ | $\tau \geq .99$ | $\tau_{AP} \geq .9$ | $\tau_{AP} \geq .95$ | $\tau_{AP} \geq .99$ |
| DOCID | 1194 | 1441 | 2815 | 1114 | 1362 | 2587 |
| MTF | 46 | 173 | 575 | 46 | 111 | 570 |
| HEDGE | **36** | **116** | 816 | **31** | **79** | 715 |
| MM-NS | 51 | 126 | **469** | 35 | 105 | **380** |

Table 4: Number of judgments required to achieve .9, .95 or .99 Kendall's $\tau$ correlation or AP correlation between the ranking of systems produced by each subset pooling method and the official ranking of systems (done with the full pool). For each correlation metric, correlation level and collection, the lowest number of judgments is bolded.

Other forms of bias, such as *out-of-the-pool* bias, which analyzes the effect on systems not contributing to the pool, can also be studied [25]. However, this form of bias exists even after evaluating the whole pool and we defer this study to future work.

## 9. Discussion

Selecting documents for assessment poses an exploration versus exploitation dilemma. Our experiments suggest that some limited form of exploration is needed, but it should be simple and we should only trigger it under certain conditions. Methods such as UCB or $\epsilon_n$-GREEDY, which explore more at the beginning of the process and gradually reduce exploration, largely fail to identify relevant documents early. Other simpler models, like MTF, only move away from a run when it supplies a non-relevant document. The population of non-relevant documents tends to grow as we move down in the rankings and, thus, MTF tends to increase exploration at later stages. This type of late exploration looks beneficial. Our empirical comparison also reveals that MM, an exploitative-only strategy, is competitive with respect to MTF and Hedge. Summing up, the best performing models either do not explore at all or, if they do explore, they do it later.

Another interesting insight comes from the non-stationary models. The stationary models are slow at reacting to drops in the quality of the runs. For example, imagine a run that initially supplied five relevant documents but has no more relevant documents. MM remains extracting documents from this run until rank #10, which is when the mean of this run goes again to $0.5$. The superiority of MM-NS over MM demonstrates the benefits of a strict management of the exploitation versus exploration tradeoff: forget about the average supply of relevant documents, just look at the last document extracted and, if non-relevant, go to explore elsewhere.

Furthermore, MM-NS is more effective than MTF, suggesting that the propagation of evidence induced by MM-NS works better than MTF's mechanism, based on priorities. Under MTF, the next run is only decided by the number of previous visits to each run. MM-NS, instead, goes on to explore runs whose last update was for accounting for a relevant document. This encodes a more intelligent way of exploration.

The Hedge algorithm, which implements a loss-based method to weight runs and select documents, works very well at identifying relevant items at the beginning of the process. However, our results demonstrate that, after having assessed several hundred documents, it is inferior to other alternatives. This suggests that, if we want to evaluate more than 500 documents, simpler exploration methods should be preferred. Hedge effectively selects documents from the top ranks but, once we have accounted for these initial items, it is inferior to other alternatives that quickly forget about past successes.

A potential criticism about bandit-based methods, MTF and Hedge is that such methods are dynamic and, thus, the next relevance assessment cannot start until the previous assessment has finished. This serialization complicates the assessment exercise, but our results suggest that these dynamic methods are worth consideration. The most effective dynamic methods require far fewer total judgments than the static methods. For example, MM-NS needs 34 judgments while DOCID needs 1181 judgments (see Table 4, TREC5, $\tau \geq .9$). Although the 1181 judgments required by DOCID can be evaluated in parallel, the total effort and time-to-task completion are likely lower with MM-NS. Other static methods, e.g. Borda or Moffat's A, are more effective than DOCID at early identifying relevant documents but, still, our experiments show that they are substantially inferior to Hedge, MTF and the bandit-based methods. Furthermore, dynamic methods could be adapted to allow parallel judgments. For example, the bandit-based methods could be modified to select the best $n$ bandits, assess $n$ documents in parallel, and update the distributions as appropriate. The effectiveness of such a hybrid method will be the subject of further research.

Our new formal models and our comparative study has also the potential to be of interest for researchers in other areas. In many data mining applications, prioritizing items from a pool of unlabeled items is a crucial step in the process of creating training data. Here, we have been concerned with the specifics of IR pooling but the lessons learned from our research are potentially applicable to other areas.

## 10. Conclusions and Future Work

This paper has shown that multi-armed bandits are a formal and effective solution for adjudicating judgments in pooling-based evaluation. By linking pooling to multi-armed bandits we have been able to define effective adjudication methods with strong theoretical grounds. For this adjudication task, we have formally analyzed the exploitation versus exploration tradeoff and we have proposed bandit-based methods that are highly competitive at early identifying relevant documents.

We have also performed a thorough evaluation of existing adjudication methods. Our experimental study gave novel insights about the relative merits of past methods and showed that our bandit-based methods are superior to state-of-the-art models.

This formal modeling opens challenging lines of future research. So far, we have not considered query-related variability. As argued in [29], some queries might require more relevance assessments than others. In this line, it might be interesting to extend the scope of bandit algorithms and run them globally. This would result in different queries having different number of assessments; and exploration versus exploitation methods could be employed to trade among queries. This has the potential of further improving the average count of relevant documents per assessment. Nevertheless, this has to be done with care because we run the risk of biasing the evaluation towards certain queries [48].

Another intriguing line of future work is to model some type of structure associated to the bandits. For instance, high-level bandits on the top of low-level bandits. To meet this aim, we will study models of hierarchical bandits, where we initially pick a top-level bandit and, next, the chosen bandit makes an internal selection as to which low-level bandit to play. These models fit well with the characteristics of our task, where many research teams contribute to the pool with multiple runs. Runs are therefore associated and it makes sense to apply hierarchical methods.

Other possible lines of future research include studying other types of non-stationary allocation methods and considering the case when rewards are not binary (e.g., handling non-binary relevance). In this work, we have not considered the scores assigned to the documents by the runs. Document scores have been effectively employed for search result merging [51]. In the near future, we also want to explore how to inject score-based evidence into our bandit-based methods.

Under the Bayesian models, the initialization of the assessment process has room for improvement. For example, we could set non-uniform priors for the runs. These prior distributions could be estimated from post-retrieval query difficulty predictors or from a score distribution analysis. And the tradeoff between exploration and exploitation can be further adjusted by reshaping the posterior distributions. This adjustment can be advantageous to tune our uncertainty about the quality of the runs. These issues might be worth exploring.

## Acknowledgements

This work has received financial support from the i) "Ministerio de Economía y Competitividad" of the Government of Spain and FEDER Funds under the research project TIN2015-64282-R, ii) Xunta de Galicia (project GPC 2016/035), and iii) Xunta de Galicia – "Consellería de Cultura, Educación e Ordenación Universitaria" and the European Regional Development Fund (ERDF) through the following 2016-2019 accreditations: ED431G/01 ("Centro singular de investigacion de Galicia") and ED431G/08.

We would also like to thank David Elsweiler for his useful comments and suggestions, and Aldo Lipani for his feedback on our implementation of the Hedge Algorithm.

## References

[1] R. Agrawal. Sample mean based index policies with O(log n) regret for the multiarmed bandit problem. *Advances in Applied Probability*, 27:1054–1078, 1995.

[2] S. Agrawal and N. Goyal. Analysis of thompson sampling for the multi-armed bandit problem. *CoRR*, abs/1111.1797, 2011.

[3] J. Aslam and M. Montague. Models for metasearch. In *Proc. of the 24th Annual Int. ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 276–284, NY, USA, 2001.

[4] J. Aslam, V. Pavlu, and R. Savell. A unified model for metasearch, pooling, and system evaluation. In *Proc. of the 12th Int. Conference on Information and Knowledge Management*, pages 484–491. ACM, 2003.

[5] J. A. Aslam, V. Pavlu, and E. Yilmaz. A statistical method for system evaluation using incomplete judgments. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 541–548, New York, NY, USA, 2006. ACM.

[6] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, May 2002.

[7] D. Bodoff and P. Li. Test theory for assessing ir test collections. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 367–374, New York, NY, USA, 2007. ACM.

[8] C. Buckley, D. Dimmick, I. Soboroff, and E. Voorhees. Bias and the limits of pooling for large collections. *Inf. Retr.*, 10(6):491–508, December 2007.

[9] B. Carterette. Robust test collections for retrieval evaluation. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 55–62, New York, NY, USA, 2007. ACM.

[10] B. Carterette, J. Allan, and R. Sitaraman. Minimal test collections for retrieval evaluation. In *Proc. of the 29th Annual Int. ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 268–275, New York, NY, USA, 2006. ACM.

[11] B. Carterette, V. Pavlu, E. Kanoulas, J. A. Aslam, and J. Allan. Evaluation over thousands of queries. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 651–658, New York, NY, USA, 2008. ACM.

[12] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2249–2257. Curran Associates, Inc., 2011.

[13] P. Clough and M. Sanderson. Evaluating the performance of information retrieval systems using test collections. *Information Research*, 18(2), 2013.

[14] G. Cormack and T. Lynam. Power and bias of subset pooling strategies. In *Proc. of the 30th Annual Int. ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 837–838, New York, NY, USA, 2007. ACM.

[15] G. Cormack, C. Palmer, and C. Clarke. Efficient construction of large test collections. In *Proc. of the 21st Annual Int. ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 282–289, NY, USA, 1998. ACM.

[16] C. Davidson-Pilon. *Probabilistic Programming & Bayesian Methods for Hackers*. Addison-Wesley Data & Analytics Series, 2015.

[17] Nicola Ferro and Gianmaria Silvello. 3.5k runs, 5k topics, 3m assessments and 70m measures: What trends in 10 years of adhoc-ish CLEF? *Information Processing & Management*, 2016 (in press).

[18] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[19] O.-C. Granmo. A bayesian learning automaton for solving two-armed bernoulli bandit problems. In *Proc. of Seventh Int. Conference on Machine Learning and Applications*, ICMLA '08, pages 23–30, Dec 2008.

[20] K. Hofmann, S. Whiteson, and M. de Rijke. Contextual bandits for information retrieval. In *NIPS 2011 Workshop on Bayesian Optimization, Experimental Design, and Bandits*, Granada, 2011.

[21] G. Jayasinghe, W. Webber, M. Sanderson, and J. Culpepper. Extending test collection pools without manual runs. In *Proc. of the 37th Int. ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '14, pages 915–918, New York, NY, USA, 2014. ACM.

[22] M. Karimzadehgan and C. Zhai. A learning approach to optimizing exploration-exploitation tradeoff in relevance feedback. *Inf. Retr.*, 16(3):307–330, 2013.

[23] G. Kazai, N. Gövert, M. Lalmas, and N. Fuhr. *The INEX Evaluation Initiative*, pages 279–293. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[24] Cheng Li, Paul Resnick, and Qiaozhu Mei. Multiple queries as bandit arms. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1089–1098, New York, NY, USA, 2016. ACM.

[25] A. Lipani, M. Lupu, and A. Hanbury. Splitting water: Precision and anti-precision to reduce pool bias. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 103–112. ACM, 2015.

[26] A. Lipani, G. Zuccon, M. Lupu, B. Koopman, and A. Hanbury. The impact of fixed-cost pooling strategies on test collection bias. In *Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval, ICTIR 2016, Newark, DE, USA, September 12- 6, 2016*, pages 105–108, 2016.

[27] D. Losada, J. Parapar, and A. Barreiro. Feeling lucky? multi-armed bandits for ordering judgements in pooling-based evaluation. In *Proc. of the 31st ACM Symposium on Applied Computing*, SAC '16, pages 1027–1034. ACM, 2016.

[28] D. Losada, J. Parapar, and A. Barreiro. A rank fusion approach based on score distributions for prioritizing relevance assessments in information retrieval evaluation. *Information Fusion*, 39:56 – 71, 2018.

[29] A. Moffat, W. Webber, and J. Zobel. Strategic system comparisons via targeted relevance judgments. In *Proc. 30th Annual Int. ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 375–382, NY, USA, 2007. ACM.

[30] A. Moffat and J. Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Trans. Inf. Syst.*, 27(1):2:1–2:27, December 2008.

[31] D. Mollá, I. Amini, and D. Martínez. Document distance for the automated expansion of relevance judgements for information retrieval evaluation. *CoRR*, abs/1501.06380, 2015.

[32] D. Oard, D. Soergel, D. Doermann, X. Huang, G.C. Murray, J. Wang, B. Ramabhadran, M. Franz, S. Gustman, J. Mayfield, L. Kharevych, and S. Strassel. Building an information retrieval test collection for spontaneous conversational speech. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 41–48, New York, NY, USA, 2004. ACM.

[33] W.H. Press. Bandit solutions provide unified ethical models for randomized clinical trials and comparative effectiveness research. In *Proc. of the National Academy of Sciences of the United States of America*, pages 22387–22392, 2009.

[34] F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *Proc. of the 25th Int. Conference on Machine Learning*, ICML '08, pages 784–791, New York, NY, USA, 2008. ACM.

[35] T. Reitmaier and B. Sick. Let us know your decision: Pool-based active training of a generative classifier with the selection strategy 4ds. *Information Sciences*, 230:106 – 131, 2013.

[36] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

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

[38] M. Sanderson and H. Joho. Forming test collections with no system pooling. In *Proc. of the 27th Annual Int. ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 33–40, New York, NY, USA, 2004. ACM.

[39] M. Sanderson and J. Zobel. Information retrieval system evaluation: Effort, sensitivity, and reliability. In *Proc. 28th Annual Int. ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 162–169, NY, USA, 2005.

[40] M. Sloan and J. Wang. Dynamical information retrieval modelling: A portfolio-armed bandit machine approach. In *Proc. of the 21st Int. Conference Companion on World Wide Web*, WWW '12 Companion, pages 603–604, NY, USA, 2012. ACM.

[41] I. Soboroff, C. Nicholas, and P. Cahan. Ranking retrieval systems without relevance judgments. In *Proc. of the 24th Annual Int. ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 66–73, NY, USA, 2001. ACM.

[42] I. Soboroff and S. Robertson. Building a filtering test collection for TREC 2002. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 243–250, New York, NY, USA, 2003. ACM.

[43] K. Sparck Jones and C.J. van Rijsbergen. Report on the need for and provision of an "ideal" information retrieval test collection. Technical report, University of Cambridge, Computer Laboratory, 1975.

[44] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[45] Ling-Xiang Tang, Shlomo Geva, Andrew Trotman, Yue Xu, and Kelly Y. Itakura. An evaluation framework for cross-lingual link discovery. *Information Processing & Management*, 50(1):1 – 23, 2014.

[46] W. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 3-4(25):285–294, 1933.

[47] A. Tonon, G. Demartini, and P. Cudré-Mauroux. Pooling-based continuous evaluation of information retrieval systems. *Information Retrieval*, 18(5):445–472, October 2015.

[48] E. Voorhees. The philosophy of information retrieval evaluation. In *Proc. of 2nd Workshop of the Cross-Language Evaluation Forum on Evaluation of Cross-Language Information Retrieval Systems*, pages 355–370, Berlin, Heidelberg, 2002.

[49] E. Voorhees and D. Harman. *TREC: Experiment and Evaluation in Information Retrieval*. The MIT Press, 2005.

[50] Ellen M. Voorhees. Variations in relevance judgments and the measurement of retrieval effectiveness. *Information Processing & Management*, 36(5):697 – 716, 2000.

[51] S. Wu and F. Crestani. A geometric framework for data fusion in information retrieval. *Information Systems*, 50:20 – 35, 2015.

[52] E. Yilmaz, J. A. Aslam, and S. Robertson. A new rank correlation coefficient for information retrieval. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 587–594, New York, NY, USA, 2008. ACM.

[53] Y. Yue and T. Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proc. of the 26th Annual Int. Conference on Machine Learning*, ICML '09, pages 1201–1208, NY, USA, 2009. ACM.