Feeling Lucky? Multi-armed Bandits for Ordering Judgements in Pooling-based Evaluation

David E. Losada
Centro Singular de Investigación en Tecnoloxías da Información (CiTIUS)
Universidade de Santiago de Compostela
david.losada@usc.es

Javier Parapar
Information Retrieval Lab
Department of Computer Science
University of A Coruña
javierparapar@udc.es

Álvaro Barreiro
Information Retrieval Lab
Department of Computer Science
University of A Coruña
barreiro@udc.es

ABSTRACT

Evaluation is crucial in Information Retrieval. The Cranfield paradigm allows reproducible system evaluation by fostering the construction of standard and reusable benchmarks. Each benchmark or test collection comprises a set of queries, a collection of documents and a set of relevance judgements. Relevance judgements are often done by humans and thus expensive to obtain. Consequently, relevance judgements are customarily incomplete. Only a subset of the collection, the pool, is judged for relevance. In TREC-like campaigns, the pool is formed by the top retrieved documents supplied by systems participating in a certain evaluation task. With multiple retrieval systems contributing to the pool, an exploration/exploitation trade-off arises naturally. Exploiting effective systems could find more relevant documents, but exploring weaker systems might also be valuable for the overall judgement process. In this paper, we cast document judging as a multi-armed bandit problem. This formal modelling leads to theoretically grounded adjudication strategies that improve over the state of the art. We show that simple instantiations of multi-armed bandit models are superior to all previous adjudication strategies.

CCS Concepts

•Information systems → Evaluation of retrieval results; •Computing methodologies → Reinforcement learning;

Keywords

Multi-armed bandits, Pooling, Information Retrieval

SAC 2016, April 04-08, 2016, Pisa, Italy

DOI: http://dx.doi.org/10.1145/2851613.2851692

1. INTRODUCTION

Relevance judgements are a core component of Information Retrieval (IR) evaluation. But relevance judgements are produced by human assessors and, thus, expensive to obtain. This is why most experimental test collections are built through a process called *pooling*. In pooled test collections only a subset –or pool– of the entire document collection is judged for each topic [4]. Having enough relevant documents in the pool we can safely assume that unjudged documents are non-relevant, leading to a set of judgements –or *qrels*– that are sufficiently complete and unbiased. In this way, we can build test collections at an affordable cost.

Evaluation campaigns like TREC, CLEF or INEX follow a common approach: i) given a set of documents and a set of topics, the campaign's organisers define a search task, ii) different participants submit their system's results, iii) for each topic, the pool of documents to be judged is constructed by taking the union of the top k –being k typically 100– documents retrieved by the participating systems. With appropriate controls on pool depth (k), as well as on the variety of pooled retrieval systems (also known as pooled *runs*) the resulting benchmark can be reused to fairly compare retrieval algorithms [20].

With a given budget for doing assessments, the more relevant documents we find, the more reliable the evaluation will be [16]. Thus, the most productive use of assessor time is spent on judging relevant documents. A number of studies in the literature [6, 12, 5] have focused on how to adjudicate pooled documents for judgement. Effective adjudication methods employ different strategies to give priority to documents that are potentially relevant. These adjudication methods, when compared with random or arbitrary adjudication, can significantly reduce the number of judgements required to obtain a qrel file with sufficient volume of relevant documents [12].

Nonetheless, research on document adjudication has concentrated on adhoc or heuristic methods to prioritise judgements. We argue that document adjudication can be naturally cast as a reinforcement learning problem and we propose solutions based on multi-armed bandit algorithms. The judging process can be seen as a learning from interaction environment. As judgements come in, we learn which runs yield the most relevant documents. The multi-armed (or n-armed) bandit problem [15] is a traditional reinforcement

learning problem that has been studied for decades. It offers a theoretical framework for analysing the tradeoff between exploration and exploitation. When an automated agent interacts with an uncertain environment, the agent can opt for exploiting its current and reliable knowledge or, alternatively, it can choose to explore the environment. Exploitation is the right thing to do to maximize the expected reward on the next action, but exploration may produce the greater total reward in the long run. This dilemma arises frequently in practice in a wide range of areas, including clinical trials, online advertisement, ecology, finance and psychology. It also arises in document adjudication for IR evaluation: focusing only on effective runs implies a risk of missing relevant documents retrieved by other apparently inferior runs.

The contributions of this paper are:

- We model document adjudication in pooling-based evaluation as a multi-armed bandit problem. Our bandit-based methods are able to early identify relevant documents in the pools. This modelling is a natural, innovative and valuable contribution. Linking IR pooling to reinforcement learning permits to apply the lessons learned over the years in this active area of research.
- We conduct a thorough comparison of past adjudication strategies. Our experiments reveal that all sophisticated adjudication strategies are inferior to an early method, MoveToFront [6], which scans the rankings in a simple way.
- We analyse the exploration/exploitation dilemma in the context of document adjudication and show how the most effective methods behave with respect to this dilemma.
- We show that some simple bandit-based solutions are superior to all previous adjudication strategies, including MoveToFront.

2. BANDIT METHODS

The multi-armed bandit problem [15] is defined as follows. You are faced repeatedly with a choice among n slot machines. Each machine –or bandit– has an unknown probability of distributing a prize. After each play, you receive a numerical reward that depends on the bandit you selected. The objective is to maximize the expected total reward over some time period. Through repeated plays, you try to maximize your winnings by concentrating your plays on the best machines. But, if you have found a bandit that gives good results: do you keep drawing from it to maintain your good score, or do you try other bandits in hopes of finding a better one? This is the classical exploration versus exploitation dilemma. Whether it is preferable to explore or exploit depends on several factors, such as the uncertainty of the current estimates or the number of remaining plays. Although the optimal solution to this problem is difficult, there are many balancing methods that implement approximately-optimal solutions that scale well.

Existing solutions for the multi-armed bandit problem can be applied for adjudicating judgements in pooling-based IR evaluation. Given a query and a set of runs –rankings of documents in decreasing order of estimated relevance–, we are often interested in finding as many relevant documents as possible for the same amount of assessor effort. Initially, we know nothing about the relative quality of the runs. As we extract documents from the runs, we gain evidence on the quality of the runs and the judging process can be oriented towards the most effective runs. At any given point, we can opt for exploring runs that currently look suboptimal. These inferior runs can eventually become good suppliers of relevant items. Playing a machine means selecting a run and examining the next document supplied by the run. Every run supplies documents according to their ranks (i.e. the top 1 document is the first document supplied, and so forth). The document is judged and the outcome of the play is the relevance degree of the document. Documents that were already judged (another run had already supplied the same document) are simply skipped. In our experiments, the judgements are obtained from the official relevance judgements of the TREC adhoc track, which contains assessments for all pooled documents. As usual in most TREC evaluations, we consider binary relevance and, therefore, we constrain our analysis to bandits with binary rewards.

2.1 Allocation strategies

In the context of the multi-armed bandit problem, a policy, or allocation strategy, is an algorithm that chooses the next machine to play based on past plays and obtained rewards. Each allocation strategy captures distinct ideas on how to handle the exploration/exploitation tradeoff. Regret is the expected loss due to the fact that the policy does not always play the best machine. The following paragraphs explain the main features of well-known allocation methods.

2.1.1 Random

This is a naive allocation strategy that randomly chooses the next machine to play. It is commonly used as a baseline for comparison.

2.1.2 ϵ_n-greedy

A greedy approach consists of always playing the bandit with the highest average reward[1]. This method maximizes immediate rewards and spends no time at all sampling apparently inferior actions. The greedy method performs worse in the long run because it often gets stuck performing suboptimal actions. A simple alternative is to behave greedily most of the time and every once in a while select an action at random. A simple algorithm that implements this idea is ϵ-greedy [18]. At each round, ϵ-greedy prescribes to play with probability $1-\epsilon$ the machine with the highest average reward, and with probability ϵ a randomly chosen machine. ϵ-greedy eventually performs better than a purely greedy approach because ϵ-greedy continues to explore, improving its chances of recognising optimal actions.

Rather than having a constant exploration probability, it is usually good to make that ϵ decreases as our estimates become more accurate. To meet this aim, ϵ_n-greedy methods let ϵ go to zero with a certain rate:

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

where n is the round number, $c > 0$ is a parameter, K is the number of machines, and d is usually set to the difference (in expected reward) between the best choice and the second best[2].

[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

2.1.3 Upper Confidence Bound (UCB)

UCB policies work by associating a quantity called upper confidence index to each machine. Among all machines, the leader at round n is the machine with the largest empirical mean of obtained rewards. While we would like to sample from this apparently superior machine, we need to make sure that the other machines have been sampled enough for us to be reasonably confident that they are indeed inferior. One way of doing this is to compare certain upper confidence bounds for the mean of an apparently inferior population with the estimated mean of the leader. The index of *UCB1* policy [3] is the sum of two terms: the current average reward and a term related to the size of the one-sided confidence interval for the average reward. *UCB1-Tuned* [3] is an evolution over UCB1 that takes into account the variance of each machine. UCB1-Tuned often performs better in practice [3]. We implemented both algorithms and also found that UCB1-Tuned is more effective than UCB1. We therefore constrain our discussion to the UCB1-Tuned algorithm:

Algorithm 1 UCB1-Tuned algorithm

Play each machine once;
Loop
  Play machine j that maximises...
  $$\mu_j + \sqrt{\frac{\ln n}{n_j} \cdot min(1/4, \sigma_j^2 + \sqrt{\frac{2 \ln n}{n_j}})}$$

where μ_j and σ_j^2 are the sample mean and variance of the rewards obtained from machine j so far, n_j is the number of times machine j has been played, and n is the overall number of plays.

The quantity added to the sample average is steadily reduced as the bandit is played, and uncertainty about the reward probability is reduced. As a result, by always selecting the machine with the highest optimistic reward estimate, UCB1-Tuned gradually shifts from exploration to exploitation.

2.1.4 Bayesian Bandits

The methods described above take a frequentist approach, where expected mean rewards are considered as unknown deterministic quantities and the goal of the algorithm is to achieve the best parameter-dependent performance. In contrast, Bayesian approaches do quantitative weighting of evidence supporting alternative hypotheses.

Each machine is characterized by a parameter which is endowed with a prior distribution. This parameter encodes the probability of winning (probability of supplying a relevant document in our case). The Bayesian process begins by assuming complete ignorance of these probabilities and, therefore, applying a uniform prior, $\mathcal{U}(0,1)$, for each machine. From these distributions we select our next machine (more on this below) and observe the result of playing the machine. With binary rewards, the result is Bernoulli or, equivalently, Binomial with a single trial. This binary outcome is used to revise our belief about the probability of winning of the machine. Observe that the initial priors are $Beta(1,1)$ –Beta handles the uniform distribution as a particular case– and Beta is the conjugate prior distribution for Binomial. Given a prior distribution $Beta(\alpha, \beta)$ and a binary outcome O, the posterior distribution is also Beta: $Beta(\alpha + O, \beta + 1 - O)$. Bayesian inference is therefore a natural framework where we can formally handle our uncertainty about the probabilities of winning.

Bayesian Learning Automaton (BLA) [8] (Algorithm 2) follows this avenue and employs random sampling from the posterior distributions to select the next machine to play. BLA is parameter-free and, usually, performs significantly better than UCB or ϵ_n-greedy [8]. Besides BLA, we also implemented another Bayesian solution where the next machine is selected by taking the maximum expectation of the posterior distributions. This exploitation method will be referred to as MM (MaxMean)[3]: $next\_machine \leftarrow \arg\max_m \alpha_m/(\alpha_m + \beta_m)$.

Algorithm 2 Bayesian Learning Automaton

foreach $m \in machines$ **do**
  $\alpha_m \leftarrow 1$, $\beta_m \leftarrow 1$;
Loop
  foreach $m \in machines$ **do**
    Draw a sample x_m from $Beta(\alpha_m, \beta_m)$;
  $next\_machine \leftarrow \arg\max_m x_m$;
  Play $next\_machine$ and get $O_{next\_machine}$;
  $\alpha_{next\_machine} \leftarrow \alpha_{next\_machine} + O_{next\_machine}$;
  $\beta_{next\_machine} \leftarrow \beta_{next\_machine} + 1 - O_{next\_machine}$;

In our pooling setting, we can make further use of the binary outcomes. Rather than simply updating the played machine's distribution (i.e. the run that supplied the last document judged), we have opted for updating the Beta distributions of all runs that retrieved the same document. In this way, evidence about relevance early flows to other runs[4].

3. EXPERIMENTS

The experiments reported here are fully reproducible. Our implementations –R scripts– of all pooling algorithms are publicly available[5].

Table 1 presents the main statistics of the four TREC collections (ad-hoc retrieval task) that we used for experimentation. In TREC, it is common to include runs produced by humans (manual runs) in the pool. This type of runs, where humans can reformulate queries and use multiple queries, contribute many unique relevant documents to the pool. Combining automatic runs and manual runs is an effective strategy to build robust test collections that do not have a bias towards the systems contributing to the pool. For each collection, we considered all runs that contributed to the pool and ran the document judging process on a query-by-query basis.

3.1 Pooling Baselines

First, we evaluated several document selection strategies that have been proposed in the past:

that performance was insensitive to d and moderately sensitive to c (all $c \in (0, 0.1]$ produced equivalent results).

[3]The expectation of a distribution $Beta(\alpha, \beta)$ is $\alpha/(\alpha + \beta)$.
[4]With MM, this *reward distribution policy* often leads to several machines having the maximum mean. Ties are resolved by selecting the played machine.
[5]http://tec.citius.usc.es/ir/code/pooling_bandits.html

	TREC5	*TREC6*	*TREC7*	*TREC8*
# queries	50	50	50	50
# automatic runs	77	31	77	71
# manual runs	24	15	7	0
# assessed docs	133681	72270	80345	86830
avg. # docs judged per query	2673.6	1445.4	1606.9	1736.6
% of rel 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 collections and pooled runs

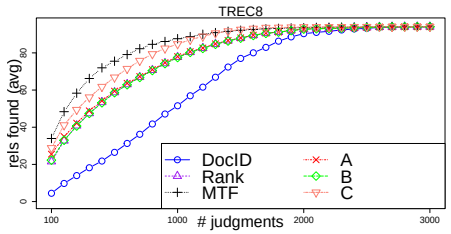

Figure 1: Comparison of baseline methods for ordering judgements: DocID, Rank, MoveToFront (MTF), and Moffat et al.'s [12] methods A, B and C. The plots show the number of relevant documents found (averaged over all queries) against the number of judgements.

DocID: The standard evaluation sequence followed by TREC assessors [19]. Each pool is simply sorted by document identifier.

Rank: Documents are selected in decreasing order of rank. First, all top 1 documents are judged: second, all top 2 documents are judged, and so forth.

MoveToFront (MTF) [6]: under MTF, the runs are prioritised (initial priorities are uniform) and the top-ranked document from the run with the top priority is judged. If it is relevant then documents from the same run continue to be examined until a non-relevant document is found. When a non-relevant document is found the priority of the current run is reduced and the judgement process jumps to another maximum priority run.

Moffat and colleagues [12] proposed several methods for ordering judgements. All of them require rank-biased precision (RBP). RBP [13] permits to weight the utility of a document obtained from a ranking based on the likelihood that users reach the document's rank. The underlying RBP model assumes that users examine documents in order and have a likelihood of 50% or so of reaching the 4th-ranked document. RBP approximates user behaviour as follows: i) the first document is always examined, ii) the user proceeds from each document to the next with probability p, or terminates her search with probability $1-p$. The overall utility of a ranking is estimated as:

$$RBP = (1-p) \cdot \sum_{i=1} u_i \cdot p^{i-1} \quad (2)$$

where $u_i \in [0,1]$ is the relevance of document at rank i. The average number of documents examined is $1/(1-p)$. It is common practice to set p to 0.8, leading to an average number of documents examined of 5, which is a reasonable approximation of actual user behaviour. This measure has been used for comparing ranking algorithms. With binary relevance and $p = 0.8$, a relevant document in the top 1 position adds 0.2 to the overall ranking score, a relevant document in the top 2 position adds 0.16, and so forth. In [12], these contributions were used for prioritising documents to be judged. The following methods were proposed:

Moffat et al.'s Method (A) [12], "*Summing contributions*": Given a document d and a set of runs S, the document is weighted as:

$$w_d = \sum_{s \in S} c_{s,d} \quad (3)$$

$$c_{s,d} = (1-p) \cdot p^{r_{d,s}-1} \quad (4)$$

where $c_{s,d}$ (which depends on d's rank, $r_{d,s}$), is the potential contribution of d to the RBP score of s (if d is relevant it would add $c_{s,d}$ to the RBP of s)[6].

This approach promotes documents that are highly ranked by many runs. The weight w_d is computed for all pooled documents and documents are judged in decreasing order of w_d. Observe that this is a *static* judgement ordering: document positions are used for prioritising judgements but it ignores the outcome of those judgements.

Moffat et al.'s Method (B) [12], "*Weighting by residual*": this is a variant that gives more weight to documents coming from runs whose *residual* RBP is large. At any given point in the judgement process, the *base* RBP is the RBP score that a run has achieved so far (computed from the documents that have been judged); while the *residual* RBP is the maximum increment in the RBP score that the run can get (if all unjudged documents that were retrieved by the run were relevant). Going to deeper judgements reduces the residual. Method B reinforces runs with large residuals, which avoids having runs with many unjudged documents. Given a document d and a set of runs S, the document weight, which is used for adjudicating judgements, is:

$$w_d = \sum_{s \in S} c_{s,d} \cdot res_s \quad (5)$$

where res_s is the residual of the run. Again, the method is static because the outcome of the judgements is ignored.

Moffat et al.'s Method (C) [12], "*Weighting by predicted score*": this method goes further and favours effective runs as follows:

$$w_d = \sum_{s \in S} c_{s,d} \cdot res_s \cdot (base_s + res_s/2)^3 \quad (6)$$

where $base_s$ is the base RBP obtained by the run so far. The range of the final RBP is $[base_s, base_s + res_s]$ and Method C takes the range's midpoint, $base_s + res_s/2$, as an estimation of the retrieval effectiveness of the run. In [12], this estimation was raised to the power of 3 to boost the strength of the effectiveness component. Method C is *dynamic* because the outcome of the judgements directly affects the value of the base RBPs of the runs.

We experimented with these six baseline strategies. The evolution of relevant documents found at varying judgement levels is depicted in Figure 1[7]. MTF is a clear winner. It is consistently superior to all other allocation methods. Not surprisingly, judging documents in order of DocID is the worst performing method. Our results also confirm the relative merits of methods A, B, C found by Moffat et al. [12]: C is superior to A and B. However, Moffat et al. did not compare the methods A/B/C against MTF. According to our experiments, MTF is clearly better than Moffat et al.'s

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

[7]Due to space constraints, we only report TREC8 data but trends were consistent across all collections.

Method	Number of judgements 100	300	500	700	900	1100	2000	all
				TREC5				
MTF	27.66	49.5	63.54	72.04	78.64	84.58	99.58	**109.9**
BLA	23.46	45.1	59.36	69.82	77.34	82.7	97.58	**109.9**
MM	**27.76**	**53.54**	**68.18**	**77.96**	**84.18**	**88.74**	**101.42**	**109.9**
RANDOM	20.94	41.48	53.38	63.42	71.28	76.36	93.92	**109.9**
UCB	20.82	46.42	58.56	67.5	74.44	79.8	96.4	**109.9**
ϵ_n-GREEDY	21.1	46.7	59.96	69.48	76.82	82.26	96.88	**109.9**
				TREC6				
MTF	31.96	55.7	66.68	75.82	82.04	86.28	**91.8**	
BLA	24.94	46.42	60.62	70.4	78.84	84.56	**91.8**	
MM	**32.12**	**56.1**	**68.98**	**78.06**	**83.16**	**87.24**	**91.8**	
RANDOM	25.56	46.9	59.98	69.7	77.4	83.82	**91.8**	
UCB	27.86	48.96	62.36	71.38	79.14	84.9	**91.8**	
ϵ_n-GREEDY	27.6	50.8	63.12	71.84	78.8	84.44	**91.8**	
				TREC7				
MTF	**35**	**58.04**	**70.58**	**78.52**	**83.48**	**86.94**	**92.7**	**92.84**
BLA	27.64	49.8	62.3	71.42	78.3	83.16	91.58	**92.84**
MM	34.62	56.4	70	78.18	83	86.36	92.4	**92.84**
RANDOM	27.48	50.74	62.86	71.86	78.44	83.44	91.3	**92.84**
UCB	30.32	52.4	64.44	72.44	79.32	83.68	91.06	**92.84**
ϵ_n-GREEDY	28	53.8	65.06	73.54	79.22	83.02	91.2	**92.84**
				TREC8				
MTF	34.06	58.48	71.78	79.22	84.5	87.58	93.22	**94.04**
BLA	27.14	50.42	64.9	73.96	80.36	84.96	93.36	**94.04**
MM	**34.4**	**59.34**	**72.9**	**80.82**	**85.56**	**88.8**	**93.54**	**94.04**
RANDOM	26.94	50.58	63.9	72.58	79.28	83.48	92.58	**94.04**
UCB	29.86	52.9	65.92	74.06	80.52	85.12	93.4	**94.04**
ϵ_n-GREEDY	28.16	53.66	66.76	74.88	80.6	84.88	93.2	**94.04**

Table 2: Bandit Allocation Strategies and Move To Front (MTF) for ordering judgements. Average number of relevant documents found at different number of judgements performed. For each judgement level and collection, the highest average of relevant documents found is bolded.

methods and better than any other pooling baseline strategy. We therefore set MTF as our reference pooling strategy.

3.2 Bandit models

Given a query and a set of runs, each run can be seen as a machine or bandit that we can play to extract a document to be judged. Playing a machine means here getting the next ranked document from a given run (starting from rank #1). Documents that were already judged from other run are simply skipped. The binary reward is here the binary relevance of the document with respect to the query. This can be obtained from the official *qrel* file, which contains judgements for all pooled documents. This sequential process simulates an iterative selection of runs, where we increasingly gain evidence about the relative merits of the runs.

Table 2 reports the number of relevant documents identified by each bandit allocation strategy described in section 2 and by MTF. At the end of the process, all strategies identify the same number of relevant documents (all pooled documents judged). However, some strategies are much quicker than others at identifying relevant documents. Early identifying relevant documents permits to reduce the judgement effort (we can just stop the judgement process when a sufficient number of relevant documents are found). The following conclusions can be drawn from these initial experiments:

- Not surprisingly, randomly selecting the next run to play is the worst performing allocation strategy. This naive method ignores how many relevant documents are found by each run and, therefore, it slowly finds relevant documents. Anyway, the counts of relevant documents found by random selection are not disproportionately low. This is because runs are selected at random, but documents are not randomly drawn from the selected run. The order of the document rankings is preserved and, therefore, the randomly selected run supplies its next top ranked document. All rankings are therefore explored starting from the top positions, which contain many relevant documents.

- Comparing BLA, UCB, and ϵ_n-GREEDY, which implement different ways to promote exploration, we observe that ϵ_n-GREEDY looks slightly superior to BLA and UCB. BLA and UCB are elaborated allocation strategies that take into account the history of wins and trials of each run and the uncertainty about the estimated quality of the runs. The jumps to explore other runs are influenced by the confidence intervals of the reward means (UCB) or by how sharp the posterior distributions are (BLA). In contrast, ϵ_n-GREEDY is a simpler allocation strategy: when it decides to explore, it jumps randomly to another run. Although the differences in average counts of relevant documents are small, these results seem to suggest that sophisticated exploration is not needed. Observe also that the supply of relevant items by the runs varies as we go down in the document rankings. In this respect, random exploration gives equal opportunities to runs regardless of the history. This randomness prevents focusing the judgement process too much on runs that were initially good but which are now exhausted.

- MM and MTF are the best performing adjudication methods. In TREC7, MTF is slightly superior to MM, but MM outperforms MTF in the other three collections. This empirical evidence suggests that MM is at least as effective as the best model in the literature. Furthermore, there is room for formally incorporating MTF-like behaviour into the Bayesian bandit models. MTF remains at the current run until a non-relevant document is found. Examining rankings in this way

entails a peculiar notion of history: only the last extraction counts. As we argue in the next subsection, this behaviour can be incorporated into the Bayesian allocation models, leading to improved bandit-based models.

3.3 Improved bandit models

With MTF, the last document examined is the only one affecting the next move. All previous documents extracted from the run are simply ignored. If the last document is relevant we remain in the same run. Otherwise, we jump to another run (based on the priorities of the runs). Forgetting quickly about previous rewards can be also incorporated into the Bayesian bandits models, as we explain next.

In stationary environments, the unknown probability of distributing a prize of the bandits does not change, and all rewards –recent or old– should be treated equally. This treatment often encounters problems where bandits change over time. In such cases, there are non-stationary solutions, e.g. based on weighting recent rewards more heavily than long-past ones [18]. Our judgement ordering problem is clearly non-stationary because the quality of the runs changes as we move down in the rankings. For instance, we cannot expect a good run constantly supplying 75% relevant documents all the way down. As we proceed examining documents, the probabilities of relevance of the runs change, and so do the relative merits of different runs. A run that was initially strong might be weak at lower rank positions when compared to the competing runs.

Stationary approaches for bandit problems run the risk of concentrating too much on runs with old wins, leading to suboptimal solutions. One popular way of tracking non-stationary problems is to use a parameter that makes that the accumulated rewards are computed as a weighted average of the past rewards and the last reward. This can be naturally incorporated into the Bayesian models BLA and MM. At any given point, the parameters of the posterior distribution of a given run s are:

$$\alpha = 1 + jrel_s \quad (7)$$

$$\beta = 1 + jret_s - jrel_s \quad (8)$$

where $jrel_s$ is the number of judged documents that are relevant and were retrieved by s, and $jret_s$ is the number of judged documents that were retrieved by s. Updating $jrel_s$ and $jret_s$ can be governed by a rate parameter that motivates the method to learn changing environments [7]. Given the binary relevance of the last document judged, rel_d, the parameters of the runs retrieving this document are updated as:

$$jrel_s \leftarrow rate \cdot jrel_s + rel_d \quad (9)$$

$$jret_s \leftarrow rate \cdot jret_s + 1 \quad (10)$$

If $rate = 1$ this is the standard method, where all rewards count the same. If $rate > 1$ the method gives more weight to early relevant documents. Conversely, if $rate < 1$ the method gives more weight to the last relevant document found. We tested different values of $rate$ and found that $rate = 0$ was the best performing setting. This resembles MTF, where only the last trial counts. Updating the parameters of the posterior distributions in this way leads to new Bayesian methods. We refer to these non-stationary Bayesian solutions ($rate = 0$) as BLA-NS and MM-NS. BLA-NS is the method that, once the distributions are updated, selects the next run by sampling from the posterior distribution. MM-NS is the method that simply selects the posterior distribution having the largest mean. Observe that setting $rate$ to 0 preserves the formality of the model: the evidence is still Bernoulli and the prior/posterior are still Beta. The effect of $rate = 0$ can be seen as a re-initialisation of the machine's counts right prior to playing the machine.

The comparison of these two alternatives against MTF, MM and BLA is reported in Table 3. In nearly all cases, MM-NS is the best performing approach. Although both MTF and MM-NS ignore previous rewards, they entail different models of exploration. When a relevant document is found both methods behave the same: they remain extracting documents from the same run. The main difference lies in the movement after finding a non-relevant document. In such a case, MTF takes into account the priorities of the runs. In practice, this makes that all runs are visited once before we re-visit any run; similarly, we only visit a run for the third time when all other runs have been visited twice, and so forth. With MM-NS, a non-relevant document makes that all runs retrieving it get $\alpha = 1$ and $\beta = 2$ and, therefore, the mean of their posterior distributions goes to 1/3. The means of the rest of the runs will be 1/3 (last update of the run was to account for a non-relevant document), 2/3 (last update of the run was to account for a relevant document), or 1/2 (still not judged documents for this run). Therefore, we tend to move towards runs whose last update was positive. This seems to be an advantage of MM-NS over MTF.

4. DISCUSSION

An exploration/exploitation dilemma arises in document adjudication for judging pooled documents. Exploiting current knowledge –selecting the run having the highest average reward– is more effective than alternative solutions such as UCB or ϵ_n-GREEDY, which incorporate some sort of exploration. Initially, UCB and ϵ_n-GREEDY tend to explore more. As we gain knowledge on the quality of the machines, UCB and ϵ_n-GREEDY increasingly reduce exploration. This behaviour does not fit well with our ranking scanning process. As matter of fact, MTF, the best performing approach in the literature, only moves away from a run when it supplies a non-relevant item. Since the quality of the runs tends to fall as we move down in the rankings, MTF tends to increase exploration at later stages. This type of scanning seems to be beneficial. We have also shown that MM, which is an exploitative-only strategy, is competitive with respect to MTF. Taken together, these results suggest: either do not explore at all or, if you want to explore, do it later.

Another interesting insight comes from our non-stationary variants. MM might be slow at reacting to falls in the quality of the rankings. For instance, consider a run that initially supplies five relevant documents and has no more relevant documents. MM would remain in this run until rank #10 (when the mean of this run goes again to 0.5). Our results suggest that we should prefer a stringent notion of exploitation, which is not based on the average supply of relevant documents but just on the last document examined (MM-NS). The models that follow this approach are capable of early finding more relevant documents when compared to MM.

MM-NS is a simple but formal model. Its definition comes from adopting non-stationary bandit techniques and apply-

Method	Number of judgements 100	300	500	700	900	1100	2000	all
				TREC5				
MTF	27.66	49.5	63.54	72.04	78.64	84.58	99.58	**109.9**
BLA	23.46	45.1	59.36	69.82	77.34	82.7	97.58	**109.9**
BLA-NS	22.8	44.8	58.74	68.78	76.04	81.44	97.4	**109.9**
MM	27.76	53.54	68.18	77.96	84.18	88.74	**101.42**	**109.9**
MM-NS	**30**	**56.76**	**70.96**	**79.06**	**85.18**	**89.42**	101.32	**109.9**
				TREC6				
MTF	31.96	55.7	66.68	75.82	82.04	86.28	**91.8**	
BLA	24.94	46.42	60.62	70.4	78.84	84.56	**91.8**	
BLA-NS	25.58	46.68	60.44	70	76.7	83.16	**91.8**	
MM	32.12	56.1	68.98	**78.06**	83.16	87.24	**91.8**	
MM-NS	**33.5**	**58.2**	**69.72**	77.9	**83.48**	**87.44**	**91.8**	
				TREC7				
MTF	35	58.04	70.58	78.52	83.48	86.94	**92.7**	**92.84**
BLA	27.64	49.8	62.3	71.42	78.3	83.16	91.58	**92.84**
BLA-NS	27.8	50.14	62.54	70.34	76.92	81.56	90.28	**92.84**
MM	34.62	56.4	70	78.18	83	86.36	92.4	**92.84**
MM-NS	**36.8**	**62.42**	**74.42**	**81.74**	**85.82**	**88.32**	92.58	**92.84**
				TREC8				
MTF	34.06	58.48	71.78	79.22	84.5	87.58	93.22	**94.04**
BLA	27.14	50.42	64.9	73.96	80.36	84.96	93.36	**94.04**
BLA-NS	27.12	49.5	63.68	72.06	77.56	82.54	92.42	**94.04**
MM	34.4	59.34	72.9	80.82	85.56	88.8	93.54	**94.04**
MM-NS	**36.96**	**64.62**	**77.3**	**82.5**	**86.34**	**89.2**	**93.6**	**94.04**

Table 3: MoveToFront (MTF), Stationary Bayesian Allocation strategies (BLA and MM), and Non-Stationary Bayesian Allocation strategies (BLA-NS and MM-NS). Average number of relevant documents found at different number of judgements performed. For each judgement level and collection, the highest average of relevant documents found is bolded.

ing them to replicate the strictness of MTF after extracting a non-relevant document. Furthermore, the superiority of MM-NS over MTF suggests that our propagation of evidence is more effective than MTF's system based on priorities. MTF's jumps are merely based on the number of times we have visited each run. MM-NS, instead, jumps to runs whose last update was for accounting for a relevant document. The empirical effectiveness of MM-NS indicates that our propagation of evidence works very well in practice.

Our comparison of existing pooling algorithms has also demonstrated that sophisticated methods, which compute document weights by aggregating evidence from all rankings, are not optimal at early finding relevant documents.

5. RELATED WORK

Pooling is fundamental in IR evaluation and, thus, has attracted much attention since its inception [19, 6, 5, 4, 10]. A number of studies have concentrated on efficient ways to scan pools, with the objective of extracting a sufficient number of relevant documents as quickly as possible. Move To Front [6] and Moffat et al.'s methods [12] are the most prominent algorithms in this area and we have thoroughly experimented with them. Our multi-armed bandit approach addresses document adjudication in a more formal way and leads to effective solutions that are very competitive with respect to the state of the art.

Multi-armed bandit algorithms have been recently employed in other areas of IR. Hofmann et al. [9] proposed models based on bandits to capture interactions between users and search engines for improving online learning to rank. In this context, an exploration/exploitation dilemma arises naturally: the web search engine has to exploit what is already known to be a good ranking, but it has also to explore by trying out variations of the current ranking algorithm. Therefore, two document lists are maintained: one exploitative (based on the current best ranking) and one exploratory (based on variations to explore potential improvements). The user is presented with a result list that interleaves documents from both lists, and preferences are inferred from user interactions (clicks). In a similar vein, Yue and Joachims [21] presented an online learning framework based on duelling bandits for comparing retrieval algorithms. The approach is based on implicit feedback gathered from users (ordinal judgements) and learns by observing interleaved results. Sloan and Wang [17] argued that the relevance of a document changes over time and proposed a dynamic model that learns from user clicks and tries to maximise the user's satisfaction by combining multi-armed bandits and the Portfolio Theory, which promotes diversity. Kleinberg and colleagues [14] presented a multi-armed bandit algorithm that learns a diverse ranking of results based on clickthrough data. The goal is to trade between relevance and diversity. The authors' approach analyses the clicking behaviour of users and attempts to minimise user abandonment in interactive experiments. Besides bandit models, other authors have proposed other ways to optimising the exploration/exploitation tradeoff. A recent work by Karimzadehgan and Zhai [11] employed a machine learning approach for optimising the overall utility of relevance feedback in an entire session of user interaction. The tradeoff here is between presenting search results with the highest immediate utility to a user and presenting search results with the best potential for collecting useful feedback information. For instance, judging documents that all have similar contents is not so useful for relevance feedback when compared to judging more diversified documents.

6. CONCLUSIONS AND FUTURE WORK

This paper has shown that multi-armed bandits are a natural solution for adjudicating documents in pooling-based evaluation. By linking pooling to duelling bandits, which is an active area of reinforcement learning, we have been able to define effective adjudication models with strong theoretical grounds. Within this framework, we have formally analysed how the exploration/exploitation dilemma affects in pooling and we have designed bandit methods that are

highly competitive at extracting relevant documents.

We have also conducted a thorough evaluation of past adjudication strategies. These experiments not only gave novel insights about the relative merits of past strategies, but also showed that our bandit-based solutions are superior to state-of-the-art models.

This formal framework opens challenging lines of future research. We have not adjusted for query-related variabilities. As argued in [12], some queries might require more judgements than others. In this respect, it might be good to extend the scope of bandit algorithms and run them globally. With multiple queries, this would result in different queries having different number of judgements; and exploration/exploitation algorithms could be employed to trade among queries. This has the potential of further improving the counts of relevant documents found. However, this needs to be done with care because we run the risk of biasing the evaluation towards certain types of queries [19].

Another intriguing line of future research is hierarchical bandits, which model some type of structure associated to the bandits (e.g., high-level bandits on the top of smaller bandits). This leads to allocation methods where we initially select a top-level bandit and, next, the chosen bandit makes an internal selection as to which bandit to pull. Pooled runs are also structured: every research team often contributes several runs to the pool. Runs from the same team are inherently associated and, therefore, it makes sense to apply hierarchical methods.

In this paper we have not been concerned about when to stop doing judgements. We have concentrated on comparing adjudication algorithms with varying number of judgements done. In the future, it will be interesting to employ existing methodologies to evaluate subset pooling strategies. For instance, a stopping criterion can be inferred from studying Kendall correlation between subset qrels and official qrels, or by doing a power/bias analysis of the subset qrels [5].

Other possible lines of research include comparing our models with models of metasearch [1, 2], studying other types of non-stationary allocation methods, and considering bandit models whose reward is not binary (e.g., by considering non-binary relevance, or by estimating the importance of the judged document for the evaluation).

With Bayesian bandits, the tradeoff between exploration and exploitation can also be adjusted by reshaping the posterior distributions. This might also be worth exploring (e.g., to further reduce/augment our uncertainty about the quality of the runs). The initialisation of the judgement process has also room for improvement. For instance, we could set non-uniform priors –or priorities– for the runs from post-retrieval query difficulty predictors, or from a score distribution analysis.

Acknowledgments

This work was supported by the *"Ministerio de Economía y Competitividad"* of the Goverment of Spain and FEDER Funds under the research projects TIN2012-33867 and TIN2015-64282-R.

7. REFERENCES

[1] 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.

[2] J. A. 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.

[3] 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.

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

[5] G. V. Cormack and T. R. Lynam. Power and bias of subset pooling strategies. In *Proc. of the 30th Annual Int. Conf. on Research and Development in Information Retrieval*, pages 837–838, USA, 2007. ACM.

[6] G. V. Cormack, C. R. Palmer, and C. L. A. Clarke. Efficient construction of large test collections. In *Proc. of the 21st Annual Int. Conf. on Research and Development in Information Retrieval*, pages 282–289, USA, 1998. ACM.

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

[8] 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.

[9] 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.

[10] G. K. Jayasinghe, W. Webber, M. Sanderson, and J. S. 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.

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

[12] 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.

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

[14] 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.

[15] H. Robbins. Some aspects of the sequential design of experiments. *Bull. Amer. Math. Soc.*, 58(5):527–535, 1952.

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

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

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

[19] 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.

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

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