When to Stop Making Relevance Judgments? A Study of Stopping Methods for Building Information Retrieval Test Collections

David E. Losada
Centro Singular de Investigación en Tecnoloxías da Información (CiTIUS)
Universidade de Santiago de Compostela
Campus Vida
15782, Santiago de Compostela (Spain)
david.losada@usc.es

Javier Parapar Alvaro Barreiro
IRLab, Department of Computer Science
Universidade da Coruña
Campus de Elviña
15071, A Coruña (Spain)
javierparapar@udc.es barreiro@udc.es

Abstract

In Information Retrieval evaluation, pooling is a well-known technique to extract a sample of documents to be assessed for relevance. Given the pooled documents, a number of studies have proposed different prioritization methods to adjudicate documents for judgment. These methods follow different strategies to reduce the assessment effort. However, there is no clear guidance on how many relevance judgments are required for creating a reliable test collection. In this paper we investigate and further develop methods to determine when to stop making relevance judgments. We propose a highly diversified set of stopping methods and provide a comprehensive analysis of the usefulness of the resulting test collections. Some of the stopping methods introduced here combine innovative estimates of recall with time series models used in Financial Trading. Experimental results on several representative collections show that some stopping methods can reduce up to 95% of the assessment effort and still produce a robust test collection. We demonstrate that the reduced set of judgments can be reliably employed to compare search systems using disparate effectiveness metrics such as Average Precision, NDCG, P@100 and Rank Biased Precision. With all these measures, the correlations found between *full pool* rankings and *reduced pool* rankings is very high.

1 Introduction

In Information Retrieval (IR), there is a long tradition of assessing the effectiveness of IR systems with test collections and evaluation measures. Through building and sharing test collections, the IR community fosters meaningful comparisons among retrieval engines. Shared test collections are pervasive in well-known evaluation campaigns, such as TREC (Voorhees & Harman, 2005), or NTCIR (Kando, Sakai & Sanderson, 2016). Furthermore, research teams sometimes need to build their own testbeds, e.g. to evaluate retrieval algorithms in specific domains (Balog & Neumayer, 2013; Losada & Crestani, 2016). However, creating an IR test collection is expensive and time-consuming.

Generating relevance assessments is a major bottleneck in building test collections. It is customary to engage humans in this process and, thus, producing large amounts of judgments becomes burdensome. The pooling method is a standard approach to extract a query-biased sample of documents. In this method, each query is sent to different search systems, and the top ranked documents supplied by the systems are merged into a pool of documents. The pooled documents become candidates to be judged for relevance. However, a key question still remains: how many pooled documents should we judge for each query?

Full-pool evaluation procedures are demanding. For example, in the TREC5 adhoc task, the pools were formed by union of the top 100 retrieved documents. On average, there were more than 2,500 pooled documents per query

and the overall judgment effort consisted of evaluating $133,681$ query-document pairs. Even with a large budget at our disposal, judging deeply the pools does not make the most of our resources. Some queries have many relevant documents, while other queries have very few relevant documents. Therefore, judging the entire pool for all queries is a waste of resources. Some evaluation exercises opted for subset pooling methods. For example, judging documents up to depth k (with a low k). Such subsetting approach reduces the judgment effort, but still ignores the specifics of the queries. Although there has been a steady stream of papers in cost-effective evaluation methods (Aslam, Pavlu & Savell, 2003; Carterette, Allan & Sitaraman, 2006; Moffat, Webber & Zobel, 2007; Cormack & Lynam, 2007; Losada, Parapar & Barreiro, 2016), we claim that there has been little research on when to stop making relevance assessments. Stopping is a practical problem in creating IR test collections, and we provide here a comprehensive study on this topic.

Combining effective adjudication methods with smart stopping permits to reduce the number of judgments per query. Following this strategy, evaluation exercises could spend the available budget on larger sets of queries. As a matter of fact, there is a growing tendency to do fewer judgments per query (Sanderson & Zobel, 2005; Bodoff & Li, 2007; Webber, Moffat & Zobel, 2008). Building test collections with more topics and less judgments per topic has been also a priority in challenges like the Million Query TREC Track (Carterette, Pavlu, Kanoulas, Aslam & Allan, 2008). However, existing methods often take arbitrary decisions to reduce the judgment effort. Our paper provides systematic ways to reduce the effort required to create relevance judgments for each query.

The specific contributions of this paper are:

- To the best of our knowledge, this is the first study that works with a diversified set of performance metrics and studies when to stop making relevance assessments. We provide a comprehensive analysis of different stopping strategies, and propose ways to evaluate them.
- We define, implement and test a series of new methods for stopping the judgment process. This covers a wide range of methods, including methods adapted from Time Series Analysis and Financial Trading.
- We propose an innovative form to estimate recall. This new estimate is employed here to support some stopping methods, but it can also contribute in other areas beyond IR evaluation.
- The results clearly demonstrate that some stopping methods can substantially reduce the assessment effort and still produce reliable test collections. As a matter of fact, the reduced set of judgments can be reliably employed to compare search systems using recall-based and utility-based metrics.

2 Stopping Methods

Let us define the problem of stopping in building an IR test collection. For each query, we have multiple *runs* (rankings of documents in decreasing order of estimated relevance). These runs are supplied by different search systems. Let us consider a document adjudication method that takes the set of runs and iteratively extracts documents to be judged for relevance. We aim at reducing the costs associated to building test collections and, therefore, we assume that the document adjudication method selects documents following some estimation of relevance (e.g. top ranked documents go first). A stopping method is a mechanism that decides when to stop doing judgments. A process with no stopping criterion would be equivalent to judging the entire set of retrieved documents.

2.1 Basic stopping methods

We distinguish two main classes of stopping methods. A *fixed-length* method stops when a given number of judgments is reached:

- stop_after_n_judgments: It consists of judging n documents, and stopping after the nth relevance assessment.

Variable-length stopping methods follow different strategies in making the stopping decision:

- stop_after_judging_x%_of_the_pool: It consists of judging a given percentage of the pool. For each query, the pool is the union of the top ranked documents supplied by the contributing runs.

- stop_after_n_rels: A natural approach consists of stopping after finding a given number of relevant documents. However, this stopping method is questionable because the number of relevant documents per query is known to have a large variance. We would produce many unnecessary judgments for queries with few relevant documents, and we would miss many relevant documents for other queries.

- stop_after_n_non_rels: This method stops right after extracting the nth non-relevant document. So, a query with many relevant documents will be deeply explored. However, this method is problematic for queries with few relevant documents. We could stop with no relevant documents extracted.

- stop_after_n_consecutive_non_rels: We only stop with a long sequence of non-relevant items. The occurrence of n consecutive non-relevant documents might indicate that the pool has become exhausted of relevant documents. This method keeps extracting documents from a pool that has supplied many non-relevant documents, provided that non-relevant documents alternate with relevant documents.

Observe that stop_after_n_rels, stop_after_n_non_rels and stop_after_n_consecutive_non_rels might produce exhaustive judgments (when the respective stopping criterion is never met).

We also propose another class of stopping methods that implement different strategies to estimate our chances of finding relevant documents in the upcoming assessments. All these methods need an estimate of the number of relevant documents that we can find in the unjudged documents. First, we propose an innovative way to make this estimation. Then, we present a number of stopping methods based on such predictions.

2.2 Predicting relevance in the unjudged set of documents

Zobel (Zobel, 1998) analyzed the depth of the rank used to form the pools and proposed a method for estimating how many unjudged documents are relevant. Consider a set of runs, explored up to depth k. This $depth_k$ pool identifies n relevant documents. Now, imagine that we increase the depth to $k+1$. The $depth_{k+1}$ pool contains n relevant documents plus n_{k+1} new arrivals of relevant documents. Zobel made a global analysis (by averaging over a set of queries) and concluded that a power law distribution provides a good estimate of the number of new relevant documents found at each pool depth. He also suggested that it would be interesting to investigate whether similar estimates can be obtained for individual queries. We took up this challenge and adapted Zobel's proposal in a number of ways. We do not work directly with the contributing runs, but iteratively examine the pooled documents as provided by the adjudication method. As a consequence, the input to our stopping process can be seen a ranked list of pooled documents. We will therefore study the relationship between the rank position in this list and the (binary) relevance of the documents at each position. Zobel hypothesized that good predictions could be made from an initial assessment of some documents from the top of the runs. We tested this proposal and found that many assessments are needed to obtain a reliable fit. For example, a curve fitted to the first 10 judgments strongly overestimates the number of relevant documents at lower positions. The high proportion of relevant documents at the initial positions makes predictions overly optimistic. The need of several dozens of judgments to have reliable fits prevents the implementation of early stopping methods. We opted for a different strategy based on training queries. Our innovative approach is explained in the next paragraphs.

2.2.1 Training stage

Given a set of training queries ($Qtrain = \{q_1, \ldots, q_{n_{tq}}\}$), with full pool judgments, we study the pattern of occurrence of relevant documents in the sequence of judgments. For each query we have a list of documents ranked by decreasing estimated relevance (as provided by the document adjudication method), and we know the relevance value ($\in \{0, 1\}$) of each document in the list. The length of each ranked list equals the size of the pool for the training query and, therefore, the lengths of the n_{tq} rankings ($l_1, \ldots, l_{n_{tq}}$) are not the same.

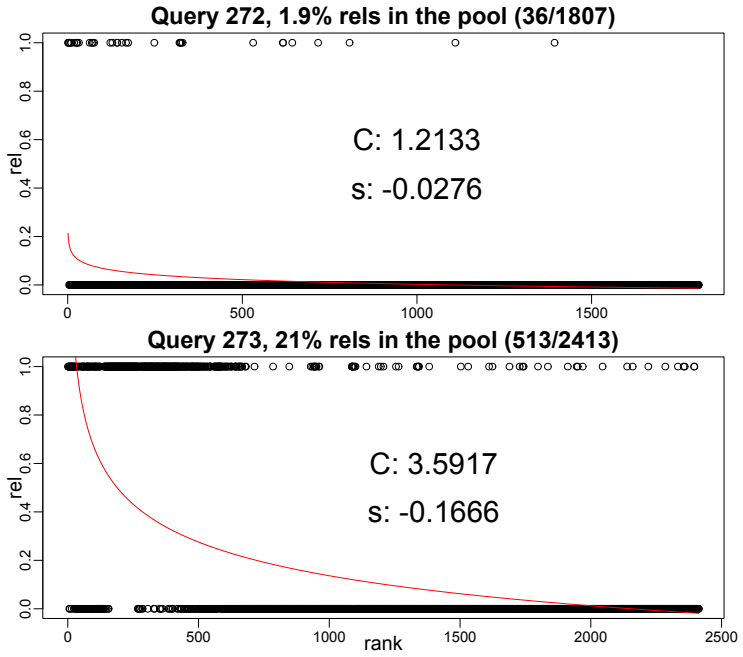

Figure 1: Fits obtained with two training queries. The X axis (rank) represents the rank positions and the Y axis (rel) represents our chances of finding a relevant document at each rank position. The circles in the plot are the relevant documents (rel=1) or non-relevant documents (rel=0) found in the ranking. The solid line is the fitted curve estimated with linear regression.

For each training query ($q \in Qtrain$), we fit a curve from the relevant and non-relevant documents found at the different rank positions. This curve represents the pattern of occurrence of relevant documents as we go down in the ranked lists. More specifically, we use the following function:

$$rel \quad = \quad C \cdot p^s - 1 \tag{1}$$

where p is the position in the ranked list, rel weights our chances of finding a relevant item at that position, and C and s are parameters of the function[1]. Doing some maths:

$$\log(rel + 1) \quad = \quad \log C + s \cdot \log p \tag{2}$$

Observe that rel can be equal to 0 and, thus, the subtraction of 1 in eq. 1 avoids here a $\log(0)$ term. With linear regression we can straightforwardly obtain s and C, inject them into eq. 1, and produce our estimates of rel for any position p. By s_q and C_q we refer to the parameters of the function associated to query q. Figure 1 shows two fits, one obtained with a query with few relevant documents and another one obtained with a query with many relevant documents.

Given the fitted curves, we can estimate the average number of relevant documents in any part of a ranking (e.g., from position i to position j, $j \geq i$):

$$estimated\_rels(i,j)[q] \quad = \quad \sum_{p=i}^{j} (C_q \cdot p^{s_q} - 1) \tag{3}$$

$$avg\_rel(i,j)[q] \quad = \quad \frac{estimated\_rels(i,j)[q]}{j - i + 1} \tag{4}$$

[1]Observe that rel does not follow a probability distribution (the proposed fitting does not constrain rel to be in $[0, 1]$).

With a sufficient number of training queries we can extract a variety of patterns of relevance, and employ them to make online predictions for test queries.

2.2.2 Test stage

Given a test query, the training queries' fitted curves can be used to estimate what we can expect from the ranking associated to the test query. The idea is to i) start judging documents from the pool of the test query, ii) estimate the similarity between the pattern of relevance of the current test query and the pattern of relevance of each training query, and iii) make a weighted estimation of relevance.

Prediction should be mainly guided by training queries with high resemblance to the test query. Imagine a test query whose first 10 assessments identified 8 relevant documents. Our prediction for the upcoming assessments should give more weight to training queries whose $P@10$ was about 0.8. First, we need a performance measure, $Perf@n$, which computes the effectiveness of the ranking at a given position n. We consider two alternatives:

$$Perf@n[q] = P@n[q] \quad (5)$$

$$Perf@n[q] = \frac{\sum_{i \in pos\_rels} P@i[q]}{n} \quad (6)$$

where pos_rels is the set of positions of the relevant documents in the top n.

The first formula computes the precision at the given cutoff value and, therefore, ignores the positions of the top n documents that are relevant. The second formula accumulates the precision scores at the relevant documents. Observe that we do not divide this sum by the total number of relevant documents (that is, we do not compute Average Precision) because we only know the total number of relevant documents for the training queries. In the following, the labels P and $avgP$ refer to the estimation variant shown in eqs 5 and 6, respectively.

Imagine that we have judged n documents from the ranking of a test query ($test_q$) and we want to predict the number of relevant documents in the remaining set of unjudged documents. Each training query can be used for such prediction, but the queries whose performance is closer to $test_q$'s performance should be given more weight. To meet this aim, we define the following measure of closeness between queries:

$$closeness@n(q_a, q_b) = 1 - |Perf@n[q_a] - Perf@n[q_b]| \quad (7)$$

Next, we can make a weighted prediction of relevance for any range of positions $\{i..j\}$ associated to unjudged documents (such that $n + 1 \leq i \leq j$):

$$predicted\_rel@n(i,j)[test_q] = \frac{\sum_{q \in Qtrain} closeness@n(test_q, q) \cdot avg\_rel(i,j)[q]}{\sum_{q \in Qtrain} closeness@n(test_q, q)} \quad (8)$$

And we can estimate the total number of relevant documents:

$$total\_rels@n[test_q] = nrels\_so\_far@n[test_q] + (l_{test_q} - n) \cdot predicted\_rel@n[test_q](n+1, l_{test_q}) \quad (9)$$

where $nrels\_so\_far@n[test_q]$ ($\in \{0, \ldots, n\}$) is the number of documents judged as relevant among the top n documents. This estimate can be used to estimate F at the current point n:

$$estimated\_F@n[test_q] = \frac{2 \cdot P@n[test_q] \cdot R@n[test_q]}{P@n[test_q] + R@n[test_q]} \quad (10)$$

$$P@n[test_q] = \frac{nrels\_so\_far@n[test_q]}{n} \quad (11)$$

$$R@n[test_q] = \frac{nrels\_so\_far@n[test_q]}{total\_rels@n[test_q]} \quad (12)$$

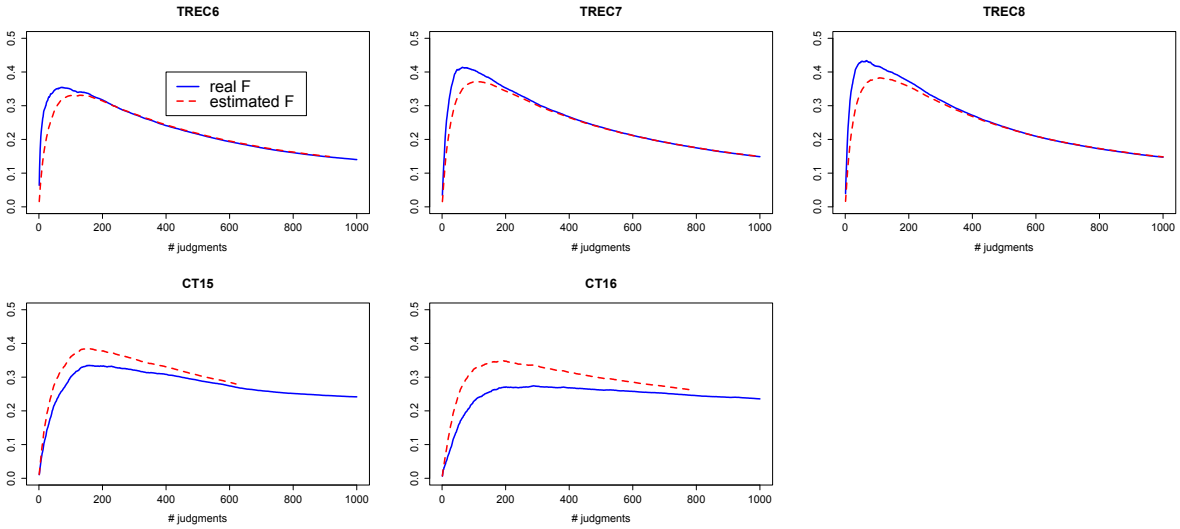

Figure 2: Real F vs Estimated F in the test collections.

Observe that the F's estimate has perfect knowledge on precision, and estimates recall based on predicting the total number of relevant documents. Figure 2 plots this estimate for three TREC adhoc collections (TREC6-8, whose estimates were done using TREC5 for training) and two Clinical Track collections (CT15-16, where CT14 was the training collection). The plots show that our procedure makes a good job at learning the shape of the curves. In TREC6-8, we tend to slightly underestimate performance. In CT15-16, instead, we tend to overestimate performance. We believe that this is due to the characteristics of training queries (relative to the test queries). This suggests that we could further improve our estimates. For example, the current weighted average (eq. 8) takes into account all training queries and, thus, it is dependent on the overall occurrence of relevant documents in the training pools. A possible improvement could be to simply ignore all training queries that are dissimilar to the test query (e.g. by doing the estimation based only on the closest queries). This is left for future work.

2.3 Stopping methods based on estimated F

Stopping the assessment process based on predicting F is judicious. Figure 3 provides evidence to support this claim. It plots the (real) F against the number of judgments (averaged over the 50 queries)[2]. The graph also plots the Kendall correlation between the official ranking of systems (full pool) and the ranking of systems built with each subset of judgments (computed with Average Precision). In all collections, we achieve high levels of correlation with few assessments, and the point with the highest F always has high correlation.

In practice, the real F is unknown, but the stopping decision can be guided by the estimate of F (eq. 10). We propose several methods that iteratively update the estimate of F (based on the available judgments) and make the stopping decision based on the evolution of the estimated F:

- stop_if_bearish_crossover. After n judgments, we have a series of n values of estimated F (estimated F after the 1st judgment, estimated F after the 2nd judgment, etc). We might want to keep assessing documents as long as the overall direction of the series of estimated F is upward. In Time Series analysis, the moving average (MA) model is a well-known method for modeling univariate time series (Cootner, 1962). An MA analyzes data points by computing series of averages of different subsets of the full set of points. Imagine a series of data points (e.g. temperature of a patient, collected at 1h intervals), and a fixed window size (e.g. 5h). The first element of the MA is the average of the initial 5 temperatures available. Then, the window is

[2]The judgments were ordered by Hedge, which is an online algorithm that is highly effective for prioritizing relevance judgments (see section 3.1).

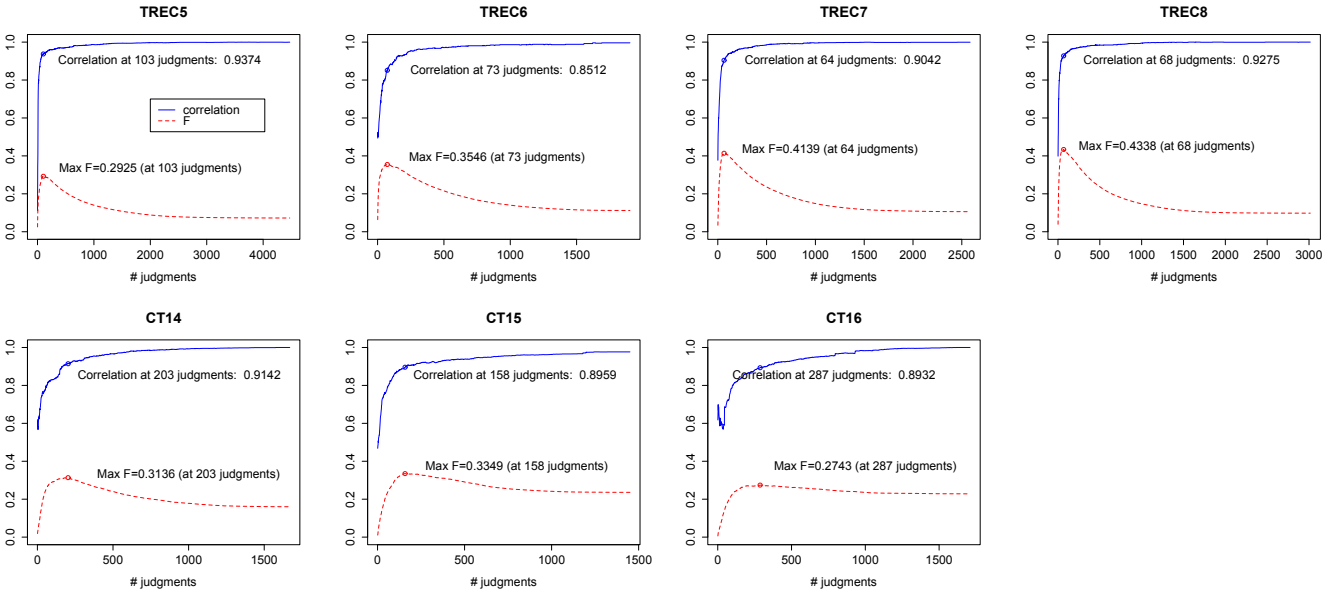

Figure 3: Evolution of F with increasing number of judgments. The graph also plots the Kendall correlation between the official ranking of systems and the ranking of systems built with the subqrel obtained at each possible stop point.

shifted forward: the first temperature is excluded and the 6th temperature is included. The second element of the MA is the average of this new window of 5 values. This shifting is repeated over the entire series of values. The MA plot connects all these numbers, and it is effective at smoothing out short-term fluctuations of the original data and highlighting long-term trends. In Economics, MA models are regularly employed to track financial data (for example, stock prices). We propose a stopping method based on MA and inspired by Financial Trading. As stock prices are moving up, the MA will be below the price, and when stock prices are moving down the MA will be above the current price. A *crossover* is a signal used by many traders to identify shifts in momentum. A basic type of crossover happens when the price of a stock moves from one side of the MA and closes on the other. Traders track these movements to make decisions on entry/exit strategies. A *cross below MA* –or bearish crossover– occurs when the stock price breaks below the MA and is often interpreted as a *sell signal* (beginning of a downtrend, the stock should be sold). Conversely, a *buy signal*, suggesting the beginning of an uptrend, is associated to a close above the MA from below (the stock price breaks above the MA, bullish crossover). Figure 4 plots an example of a stock price and its MA (smooth dashed line). The lines show several crossovers. In early September, we had a bearish crossover because the stock price broke below the MA. In early November, we had a bullish crossover because the stock price broke above the MA.

We are interested in tracking the estimated F and keep assessing documents as long as the curve of estimated F is moving up. We iteratively construct a curve of estimated F, compute its MA, and stop the assessment process when we detect the beginning of a downtrend (F crosses below its MA). This stopping method has one parameter: the size of the window used to compute the MA (w). The pseudocode of the three algorithms presented in this subsection is available at our institutional website [3].

- stop_if_no_better_expectations. The previous method estimates F with perfect knowledge on precision. For example, after 5 judgments, we know the precision achieved so far, and we just need an estimate of the total

[3] http://tec.citius.usc.es/ir/code/jasist_submission2017_stopping_algs.pdf

Figure 4: The plot shows the price of a stock (solid line) and the 50-day moving average (dashed line).

number of relevant documents in order to get an estimate of $F@5$. However, at any given position n, we can estimate not only $F@n$, but also $F@p$ ($\forall p > n$). The estimation of $F@p$ ($\forall p > n$) needs to estimate both $P@p$ and $R@p$. We can predict the number of relevant documents in any range of positions and, therefore, it is straightforward to use our predictions (see section 2.2) to estimate not only recall but also precision. A natural stopping strategy is to terminate the assessment process at a point n whose estimated $F@n$ is greater than the estimated $F@p, \forall p > n$ (if we do not expect to improve F then why should we keep assessing documents?). A nice feature of this method is that it only requires the training queries (it has no additional parameters).

- stop_if_fall_below_max. This method stores the maximum estimated F achieved so far, and stops when the current estimation of F is below a given proportion of the maximum. The intuition is that if we improve over the maximum then we should keep doing judgments; but if the current estimated F substantially falls below the maximum then we should stop. This method has a parameter, $prop$, which sets the proportion. For example, if the maximum estimated F is 0.6 and the proportion is 0.9 then we would stop with an estimated F below 0.54.

3 Experiments

The experiments described here are fully reproducible. The reader can download our code and instructions from our institutional website[4]. This includes an implementation in R of all the stopping methods and the adjudication strategy (Hedge).

[4] http://tec.citius.usc.es/ir/code/pooling_stopping.html

	TREC5	TREC6	TREC7	TREC8	CT14	CT15	CT16
	(train)	(test)	(test)	(test)	(train)	(test)	(test)
# queries	50	50	50	50	30	30	30
avg. pool size (# docs)	2692.3	1445.1	1611.1	1785.9	1264.9	1022.8	1256.9
min pool size (# docs)	1623	914	1025	1114	908	620	783
max pool size (# docs)	4472	1902	2585	3015	1669	1453	1710
avg % of relevant docs in the pool	4.1%	6.4%	5.8%	5.4%	9.2%	15.1%	14.8%
avg. # relevant docs in the pool	110.5	92.2	93.5	94.6	111.9	143.0	182.0

Table 1: Main statistics of the collections used for experimentation.

3.1 Test Collections

Table 1 presents the statistics of the collections used for experimentation. TRECs 5-8 are classical adhoc collections, while CTs 14-16 are newer collections developed under the TREC Clinical decision support Track (search task in the medical domain). We obtained from TREC all the runs (ranked lists of documents for each topic) that contributed to the pool. For each topic, we ordered the pooled documents following the Hedge algorithm, and simulated the assessment process on a sequential basis[5]. First, the top ranked document (as selected by Hedge) is assessed for relevance[6]. Next, Hedge re-ranks the remaining pooled documents and the process continues until the stopping method resolves to stop.

A number of studies have proposed different prioritization methods to adjudicate pooled documents for judgment (Cormack, Palmer & Clarke, 1998; Moffat et al., 2007; Carterette, 2007; Aslam, Pavlu & Yilmaz, 2006; Losada et al., 2016). A recent comparison of subset pooling methods can be found in (Losada, Parapar & Barreiro, 2017). This comparative study concludes that the Hedge algorithm performs the best in building shallow judgment sets. We therefore adopted Hedge as our reference method to adjudicate documents for judgment and focused on evaluating the stopping methods discussed above. Hedge is an online learning method that maintains a set of weights for the participating systems, ranks the pooled documents based on the system weights and document positions, and reacts to the assessments by updating the system weights and re-ranking the remaining unjudged documents. The update is governed by a learning rate parameter that determines how quickly the algorithm reacts to new judgments[7]. A full description of the Hedge algorithm can be found in (Aslam et al., 2003) (see also (Losada et al., 2017), Algorithm 4).

3.2 Evaluation Metrics

For each query, every stopping method determines a point where to stop doing relevance judgments. After applying the stopping method on all queries, we obtain a set of relevance judgments that can be used for quality assessment. Two main dimensions will be used for evaluation:

- Rank correlation measures. A standard way to measure the quality of subset pooling methods consists of employing the official ranking of systems as the basis for comparison. More specifically, the method ranks the runs submitted to TREC using the official judgments (full pool), and ranks the same runs using a different set of judgments. An effectiveness metric is needed for producing the rankings of the runs. The effectiveness metric plays here a key role. Evaluating stopping methods based on a single measure, such as Average Precision, would give little clue as to what extent the subqrels are reusable to compare systems using other effectiveness measures. There is a complex interplay between effectiveness metrics and judgment pools. We therefore considered several effectiveness metrics and studied the implications of the

[5]In TRECs 5-8 the pool was the union of the top 100 documents retrieved by any run. In the CTs, the pool was formed from all documents retrieved in ranks 1-20 by any run in union with a 20% sample of documents not retrieved in the first set that were retrieved in ranks 21-100 by some run.

[6]Every time a relevance judgment is required we obtain it from the official TREC judgments.

[7]Following standard practice, this parameter was fixed to 0.1 in all our experiments.

stopping methods for all of them. Following (Lu, Moffat & Culpepper, 2016), our study included two main families of evaluation metrics: *recall-based* metrics, Average Precision (AP) and Normalized Discounted Cumulative Gain (NDCG), and *utility-based* metrics, Precision at 100 and Rank Biased Precision (RBP)[8].

High correlation means that the judgment sets ranked the runs similarly. Low correlation, instead, means that each judgment set has ranked the runs differently and, thus, we cannot trust the subset pooling method. Kendall's τ has been commonly applied to compare these two rankings. Another rank correlation measure that has been used in comparing rankings in IR is AP correlation (Yilmaz, Aslam & Robertson, 2008). Unlike Kendall's τ, AP correlation (τ_{AP}) considers that discrepancies among those systems at high positions in the rankings are more important than those among systems at low positions in the rankings. AP correlation fits well with IR evaluation, where we are often concerned about identifying the best performing systems. In our experimental report, we provide results for both correlation measures. A number of researchers have considered that levels of correlation above 0.85 indicate that the rankings of runs, though not identical, are highly similar (Voorhees, 2000, 2001). Such high correlation would permit to conclude that variations in relevance judgments did not impact noticeably on the reliability of the test collection.

- Number of judgments required. We pursue stopping methods that are reliable and *low cost* (the fewer judgments, the better). We therefore report the minimum, maximum and average number of judgments done (over all queries).

For descriptive reasons, we will also report the F-score associated to the resulting qrels (averaged over all queries). The most advantageous use of assessors' time occurs when the assessors focus on relevant documents, and the aim of the evaluation task is to identify as many relevant documents as possible. A natural way to evaluate these two aspects of the assessment process consists of extracting the set of judgments done for each query and compute a classic set-based measure of effectiveness, such as F1. Under this setting, precision is the fraction of assessed documents that are relevant and, thus, it is a good estimator of how well we have used the assessors' time. Recall is the fraction of (pooled) relevant documents that have been assessed and, thus, it gives us an indication on how well we have identified the existing relevant documents.

3.3 Results

The values of the parameters of the stopping methods were learned by optimizing F over the training collections (TREC5 and CT14, respectively). These two collections were also used for learning the fits of the training queries (see Fig. 1). These fits together with the performances of the training queries ($Perf@n\ \forall n$) were stored and used for supporting recall predictions of test queries. Although this training stage requires a collection with full pool judgments, a single training collection can be employed for supporting early stopping in many other similar evaluation exercises.

The parameters learned with the training collection (and tuning grids) are reported in Table 2, and the test results are shown in Tables 3 and 4. The fixed-length strategy (stop_after_n_judgments) requires more judgments than other competing methods and, furthermore, it leads to correlations below .85 in some collections and metrics (CT16-RBP). Judging a given percentage of the pool is a low cost strategy (low average stopping points) but it has many correlations below .85 (TREC6-AP, CT15-RBP, CT16-AP/RBP). Stop_after_n_rels and stop_if_no_better_expectations are costly strategies and they have some correlations below .85 (CT15-RBP, CT16-AP/RBP). Stopping after finding n non-relevant documents, stop_after_n_consecutive_non_rels and stop_if_fall_-below_max have a moderate cost but they show correlations below .85 in CT15 and/or CT16. The most consistent strategy is stop_if_bearish_crossover (avgP). It is low cost and exhibits a solid behavior over all collections. For all the metrics tested (AP, NDCG, P@100 and RBP), stop_if_bearish_crossover (avgP) led to a subset of judgments that ranked retrieval systems similarly to a full pool approach. Both τ and τ_{AP} show very high correlations for the

[8]NDCG was computed following the traditional setting incorporated in $trec\_eval$ (log_2 discounting factor and gain levels set to the relevance levels available in the test collections). The RBP parameter was set to 0.8.

Stopping Method	Tuned Parameter (TREC5,CT14)	Tuning Grid
stop_after_n_judgments	$n = 103, 203$	$1, 2, \ldots, max\ pool\ size$
stop_after_judging_x%_of_the_pool	$x = 4\%, 15\%$	$1\%..10\%, 15\%, 20\%$
stop_after_n_rels	$n = 60, 50$	10..100 (steps of 10)
stop_after_n_non_rels	$n = 80, 80$	10..100 (steps of 10)
stop_after_n_consecutive_non_rels	$n = 15, 20$	$1..10, 15, 20, 50, 100$
stop_if_bearish_crossover (P)	$window\ size\ (MA) = 40, 70$	10..100 (steps of 10)
stop_if_bearish_crossover (avgP)	$window\ size\ (MA) = 30, 80$	10..100 (steps of 10)
stop_if_no_better_expectations (P)	-	-
stop_if_no_better_expectations (avgP)	-	-
stop_if_fall_below_max (P)	$prop = 0.95, 0.93$	0.90..0.99
stop_if_fall_below_max (avgP)	$prop = 0.93, 0.90$	0.90..0.99

Table 2: Tuned parameters after optimization with the training collections (TREC5 and CT14).

two classes of effectiveness metrics. Although this method appears to be the most preferable choice, an important outcome of these experiments is that there are a number of reasonable stopping methods whose resulting set of judgments can be reliably used to compare search systems under different performance measures. Depending on the available budget and other constraints, an experimenter can use our comparison to select her method of choice.

4 Discussion

Our evaluation provides a thorough understanding of when to stop doing relevance judgments. The results clearly demonstrate that some stopping methods can substantially reduce the assessment effort, and the suitability of the resulting relevance assessments as a tool for evaluating retrieval performance was not compromised.

With stop_if_bearish_crossover (avgP), the average required effort is about 100 or 185 judgments per query (in TREC and CT collections, respectively). This is a substantial reduction over a full-pool approach. In TREC6, the average pool size is 1445.1 documents, and stop_if_bearish_crossover (avgP) stops on average after 92.88 judgments. This means that only 6.4% of the pooled documents had to be assessed for relevance. In TREC7-8 and CT15-16, the percentages of pooled documents that are judged are 6.4%, 5.9%, 18%, and 15.1%, respectively. Saving up to 93% of the judgments is really significant. These savings directly reduce the costs of creating the test collection. The creators of test collections can spend these extra savings on building testbeds with a much larger set of topics.

					TREC6							
					AP		NDCG		P@100		RBP	
Stopping Method	Stop pt.(Avg Min Max)	F	P	R	τ	τ_{AP}	τ	τ_{AP}	τ	τ_{AP}	τ	τ_{AP}
after_n_judgments	103 103 103	.347	.37	.59	.88	.88	.94	.94	.88	.89	.90	.89
after_judging_x%_of_the_pool	57.8 37 76	.348	.45	.49	.80	.81	.90	.91	.87	.87	.88	.87
after_n_rels	842.9 64 1894	.212	.28	.76	.96	.97	.96	.97	.89	.89	.90	.91
after_n_non_rels	136.2 80 341	.381	.32	.67	.91	.90	.95	.96	.92	.94	.94	.95
after_n_consecutive_non_rels	128.2 15 652	.435	.36	.64	.86	.86	.92	.93	.93	.95	.94	.95
if_bearish_crossover (P)	107.9 41 341	.434	.38	.65	.89	.89	.94	.95	.92	.94	.93	.92
if_bearish_crossover (avgP)	92.9 31 328	.441	.41	.62	.87	.86	.93	.94	.91	.91	.92	.92
if_no_better_expectations (P)	516.5 71 1629	.294	.37	.65	.90	.89	.94	.95	.88	.89	.90	.89
if_no_better_expectations (avgP)	532.5 57 1651	.277	.37	.64	.89	.89	.94	.95	.87	.89	.89	.89
if_fall_below_max (P)	132.9 23 611	.436	.35	.67	.88	.87	.93	.94	.93	.95	.95	.93
if_fall_below_max (avgP)	131.3 8 628	.426	.34	.67	.88	.88	.93	.94	.94	.96	.94	.93
					TREC7							
					AP		NDCG		P@100		RBP	
Stopping Method	Stop pt.(Avg Min Max)	F	P	R	τ	τ_{AP}	τ	τ_{AP}	τ	τ_{AP}	τ	τ_{AP}
after_n_judgments	103 103 103	.405	.40	.59	.93	.93	.94	.96	.94	.94	.94	.94
after_judging_x%_of_the_pool	64.5 41 103	.415	.50	.49	.89	.87	.93	.95	.93	.93	.91	.91
after_n_rels	858.2 70 2322	.244	.29	.74	.95	.95	.95	.96	.94	.94	.93	.93
after_n_non_rels	136.9 83 270	.427	.35	.67	.94	.94	.94	.96	.96	.96	.95	.96
after_n_consecutive_non_rels	149.3 16 622	.488	.39	.67	.93	.92	.95	.96	.97	.97	.98	.98
if_bearish_crossover (P)	118.4 50 284	.479	.41	.65	.94	.93	.94	.96	.96	.96	.95	.96
if_bearish_crossover (avgP)	104.3 42 274	.490	.44	.63	.93	.93	.93	.95	.96	.96	.95	.95
if_no_better_expectations (P)	384.9 72 1576	.363	.42	.62	.93	.93	.94	.95	.93	.94	.92	.93
if_no_better_expectations (avgP)	391.6 58 1600	.345	.42	.60	.92	.92	.94	.95	.93	.94	.92	.93
if_fall_below_max (P)	138.8 26 544	.468	.38	.66	.93	.94	.94	.96	.97	.97	.97	.97
if_fall_below_max (avgP)	141.3 21 560	.468	.38	.66	.93	.92	.94	.96	.97	.97	.97	.97
					TREC8							
					AP		NDCG		P@100		RBP	
Stopping Method	Stop pt.(Avg Min Max)	F	P	R	τ	τ_{AP}	τ	τ_{AP}	τ	τ_{AP}	τ	τ_{AP}
after_n_judgments	103 103 103	.417	.42	.62	.95	.96	.91	.92	.94	.94	.96	.96
after_judging_x%_of_the_pool	71.5 45 121	.445	.51	.56	.93	.94	.89	.91	.94	.94	.95	.96
after_n_rels	865.8 67 2562	.239	.30	.72	.95	.96	.93	.94	.94	.92	.97	.97
after_n_non_rels	141.9 86 283	.446	.37	.71	.96	.97	.93	.93	.97	.97	.98	.98
after_n_consecutive_non_rels	127.4 15 428	.533	.45	.71	.96	.97	.92	.92	.98	.98	.97	.97
if_bearish_crossover (P)	120.7 43 382	.516	.44	.70	.96	.97	.92	.92	.98	.98	.98	.98
if_bearish_crossover (avgP)	105.6 34 330	.530	.48	.68	.95	.96	.92	.92	.96	.96	.96	.96
if_no_better_expectations (P)	394.5 70 1603	.351	.43	.60	.93	.95	.91	.92	.94	.93	.95	.95
if_no_better_expectations (avgP)	400.9 61 1626	.340	.43	.59	.93	.95	.91	.92	.94	.92	.94	.95
if_fall_below_max (P)	133.5 28 432	.496	.41	.71	.96	.97	.93	.93	.98	.98	.98	.97
if_fall_below_max (avgP)	137.5 27 450	.501	.41	.71	.95	.97	.92	.92	.98	.98	.98	.98

Table 3: Test Results. The columns τ and τ_{AP} report the Kendall τ correlation and AP correlation between the official ranking of systems (computed with the full pools) and the ranking of systems computed with the corresponding subqrels. These correlations are computed for each performance measure (AP, NDCG, P@100 and RBP).

CT15												
					AP		NDCG		P@100		RBP	
Stopping Method	Stop pt.(Avg Min Max)	F	P	R	τ	τ_{AP}	τ	τ_{AP}	τ	τ_{AP}	τ	τ_{AP}
after_n_judgments	203 203 203	.333	.32	.49	.91	.92	.90	.92	.90	.91	.88	.87
after_judging_x%_of_the_pool	153.4 93 218	.325	.37	.41	.89	.90	.88	.89	.89	.90	.84	.84
after_n_rels	479.9 50 1453	.251	.38	.59	.86	.87	.87	.88	.87	.88	.82	.83
after_n_non_rels	151.1 81 465	.352	.34	.41	.87	.88	.85	.87	.89	.89	.88	.88
after_n_consecutive_non_rels	236.6 20 1092	.345	.28	.47	.84	.84	.84	.85	.92	.92	.90	.89
if_bearish_crossover (P)	167.6 71 408	.366	.35	.44	.89	.91	.86	.87	.90	.91	.88	.87
if_bearish_crossover (avgP)	184.4 86 416	.372	.34	.47	.91	.92	.90	.91	.91	.92	.90	.89
if_no_better_expectations (P)	302.1 73 1244	.269	.38	.48	.86	.88	.86	.87	.86	.88	.83	.83
if_no_better_expectations (avgP)	303.5 63 1260	.262	.39	.47	.85	.87	.85	.86	.86	.87	.82	.82
if_fall_below_max (P)	180.2 13 663	.355	.32	.43	.83	.83	.85	.86	.91	.91	.90	.91
if_fall_below_max (avgP)	198.9 15 826	.349	.30	.45	.83	.84	.85	.85	.91	.91	.91	.91
CT16												
					AP		NDCG		P@100		RBP	
Stopping Method	Stop pt.(Avg Min Max)	F	P	R	τ	τ_{AP}	τ	τ_{AP}	τ	τ_{AP}	τ	τ_{AP}
after_n_judgments	203 203 203	.271	.29	.37	.87	.85	.89	.89	.89	.89	.81	.82
after_judging_x%_of_the_pool	188.5 117 256	.266	.29	.34	.86	.84	.89	.89	.89	.89	.80	.80
after_n_rels	447.0 54 1428	.217	.31	.49	.84	.84	.90	.90	.87	.87	.81	.82
after_n_non_rels	144.6 81 447	.281	.31	.29	.84	.83	.87	.87	.89	.89	.81	.79
after_n_consecutive_non_rels	293.9 20 1461	.279	.23	.38	.75	.77	.86	.87	.92	.91	.83	.84
if_bearish_crossover (P)	185.3 72 460	.302	.31	.35	.85	.84	.88	.88	.90	.89	.85	.85
if_bearish_crossover (avgP)	190.6 85 460	.302	.31	.36	.86	.85	.88	.88	.90	.89	.85	.85
if_no_better_expectations (P)	294.3 74 1182	.215	.33	.38	.82	.81	.88	.88	.87	.87	.76	.77
if_no_better_expectations (avgP)	287.1 60 1186	.203	.33	.36	.79	.79	.87	.86	.85	.86	.74	.75
if_fall_below_max (P)	192.4 13 1238	.261	.26	.30	.66	.67	.82	.81	.89	.89	.82	.81
if_fall_below_max (avgP)	207.4 16 1238	.268	.26	.32	.69	.70	.81	.82	.91	.88	.83	.82

Table 4: Test Results. The columns τ and τ_{AP} report the Kendall τ correlation and AP correlation between the official ranking of systems (computed with the full pools) and the ranking of systems computed with the corresponding subqrels. These correlations are computed for each performance measure (AP, NDCG, P@100 and RBP).

The premise of these stopping methods is that pooled documents are ordered by predicted relevance. A potential criticism is that this ordering of documents might impact on the assessments. However, this type of ordering of assessments has been adopted by some evaluation campaigns. For example, NTCIR sorts the pooled documents primarily by the number of the systems that returned the document (Sakai & Lin, 2010). As argued by Sakai and colleagues (Sakai et al., 2008), there is no evidence that ordering the assessments by predicted relevance really biases them and affects results. Ordering the assessments by predicted relevance makes it easier for the human assessors to make judgments more consistently and efficiently than when relevant documents are spread across a list of judgments (Sakai et al., 2008). Aslam (Aslam et al., 2006) also argued that more judging effort should be placed on documents that are likely relevant. Similar conclusions were drawn by Zobel (Zobel, 1998). Although there is still room for debate, we believe that a relevance-based ordering of the assessments should not be an obstacle in practice.

Another potential criticism about this assessment process is that Hedge, the method used for adjudicating documents for judgment, is *dynamic*. Dynamic methods select a document from the pool, send the document for judgment, and the outcome of the assessment affects the decision on the next pick. The next relevance assessment cannot start until the previous assessment has finished. Although such serialization complicates the assessment exercise, the benefits of Hedge far outweight its disadvantages. Hedge requires much fewer total judgments than other static (non-adaptive) methods. For a given query, the judgments required by static methods can be done in parallel. However, static approaches require at least ten times more judgment effort than Hedge in order to achieve high levels of correlation with respect to a full pool approach (Losada et al., 2017). Carterette and colleagues (Carterette, 2007) also argued for dynamic methods and proposed stopping rules and document orderings that adapt to distributions of relevance (as they are discovered through judging).

A natural question is whether a test collection built from these stopping procedures may be able to evaluate systems that did not participate in the creation of the test collection. Following (Büttcher, Clarke, Yeung & Soboroff, 2007), we tested reusability by analysing the ranks achieved by runs under *leave-one-group-out* (LOGO)

	AP	NDCG	P@100	RBP
TREC6	0.174	-0.217	-0.087	-0.478
TREC7	-0.393	-0.440	-0.417	-0.405
TREC8	0.014	-0.085	-0.070	0.000
CT15	0.225	0.314	-0.137	0.196
CT16	-0.757	-0.470	0.026	-0.322

Table 5: Reusability of the relevance judgments produced by stop_if_bearish_crossover (avgP). The table reports the average difference in the rank position of a system (position of the system using the subqrels with all systems - position of the system using the subqrels with the system's group removed).

experiments. For each run that contributed documents to the pools we simulated how removing all runs submitted by the same group would affect its ranking. For each run, we repeated the following procedure over all queries: i) those documents contributed only by runs from the same team are removed from the pool, ii) the remaining pooled documents are ordered by the Hedge algorithm (using only the runs associated to other teams), iii) the prioritisation of pooled documents induced by Hedge is given to the stopping method, which produces a set of judgments, iv) we ranked all systems (including the runs removed) with the *LOGO qrels* and with the qrels produced by the stopping method with no system removed[9], and v) by comparing the position of the runs in these two rankings we can understand the effect that the stopping procedure had on systems whose group did not have the opportunity to contribute to the pool. Table 5 reports the results of this experiment. On average, removing the run's group leads to negligible changes of positions in the ranking (all averages below 1 position). In some cases, the non-contributing runs tend to be favored (for example, TREC6-AP). This may seem counterintuitive but observe that this is not a full pool evaluation procedure and, therefore, the fact that a team inserts some unique documents into the set of candidate documents does not guarantee that these unique documents are actually assessed for relevance (the stopping procedure might stop earlier). In any case, the effect on runs from non-contributing teams is unnoticeable, suggesting that the test collection can be reused reliably. Observe also that the absolute position differences are lower than those reported in previous studies (Büttcher et al., 2007). By judging a selected set of top retrieved candidates, we avoid a deep exploration of the contributing runs. Such shallow strategy appears to be good at avoiding biases towards specific runs or teams.

5 Related Work

Zobel (Zobel, 1998) suggested that variable-length pools could be a cost-effective way to build retrieval collections, and outlined a method to predict the overall number of relevant documents at different pool depths. As argued in Section 2, some stopping methods proposed here are evolutions over Zobel's proposal. Our prediction approach works with training queries because the original approach suggested by Zobel leads to unreliable estimates for individual queries.

Other authors have proposed sampling methods to extract a small set of judgments from the pool. For example, Aslam and colleagues (Aslam et al., 2006) experimented with a method that produces a reduced set of judgments. They showed that you could sample as little as 4% of the pools and still produce accurate results. The main goal was to efficiently and accurately estimate standard measures of retrieval performance, such as AP. Their work is theoretically appealing, but every performance measure leads to a different sampling distribution. Furthermore, Aslam et al. did not investigate on how to determine the ideal number of judgments. They simply experimented with two main configurations for determining the size of the judgment set (one of them led to an average of 29 judgments/query, and the other produced an average of 200 judgments/query). Carterette also contributed to this area with a number of low-cost and robust evaluation strategies (Carterette., 2008). In (Carterette et al., 2006), he and his colleagues proposed the Minimal Test Collection (MTC) method, which can be used to rank retrieval systems with a minimal number of judgments. MTC is oriented to rank systems with AP. Given a set of judged

[9]This experiment was done with the stop_if_bearish_crossover (avgP) method, whose resulting qrels have been shown to have high correlation with respect to full pool judgments.

documents and a set of unjudged documents, MTC defines a stopping condition based on how AP would change if unjudged documents were judged relevant. This stopping condition was further developed into a probabilistic stopping criterion. Our study focuses on stopping methods that work with past assessments and/or estimates of relevance in the upcoming assessments; and stopping is not governed by how well we can estimate a single performance metric, such as AP. We have been specifically concerned with when to stop and we showed that the methods lead to robust relevance assessments for recall-based and utility-based effectiveness metrics. However, in the future we will study how to apply the lessons learned by these teams to our research.

Cormack (Cormack et al., 1998) studied the efficient construction of IR test collections and proposed a method to prioritize relevant assessments. The approach, called MoveToFront, consisted on focusing assessor effort onto those contributing runs richer in relevant documents. Although Cormack's study was not concerned with when to stop, it showed that you could assess 10% of the pooled documents and still produce robust qrels. We did not employ MoveToFront as our reference method to prioritize relevance assessments because MoveToFront was shown to be inferior to Hedge in a recent study that compared many document adjudication strategies (Losada et al., 2017).

Our paper is also related to the problem of finding an optimal point to stop reading a ranked list (Arampatzis, Kamps & Robertson, 2009). Arampatzis et al. developed methods to select the cut-off value that optimizes a given effectiveness metric. They casted the problem as a score distributional threshold optimization problem. Other teams have applied score distributions to feedback tasks (Parapar, Presedo-Quindimil & Barreiro, 2014). Threshold optimization was also examined for threshold calibration in information filtering (Zhai, Jansen & Evans, 2000). These threshold-based models could be also employed for implementing methods to stop making relevance assessments. However, query–document retrieval scores would be required. This limits the applicability of such an approach. The stopping methods proposed in our paper do not need scores and, thus, they can be applied to a wider variety of cases. In any case, the use of score distributions to support IR evaluation requires further investigation. A recent exploration on this topic (Losada, Parapar & Barreiro, 2018) showed that, if scores are available, score distribution models can effectively prioritize relevance judgments.

Cormack and Grossman (Cormack & Grossman, 2016) were interested in *total recall* in technology-assisted reviews. They presented stopping methods oriented to estimate when nearly all relevant documents have been identified (e.g., based on detecting a knee on the recall versus rank curve). Although these methods afford a good balance between high recall and low effort, we think that they are not well suited for our current purposes. Building an effective test collection does not necessarily require to identify all relevant documents. The main goal is to obtain a set of relevance assessments that are reliable to rank search systems (the fewer, the better). As shown in our experiments, effective stopping methods skip many judgments (i.e. we miss many documents that are potentially relevant) but still produce robust qrels. In any case, we will further investigate whether these high-recall stopping methods can be adapted for building IR test collections.

6 Conclusions

In this paper we have proposed and compared a number of methods to determine when to stop doing relevance judgments. We defined and implemented several methods that follow different intuitions to set a stopping point. These stopping methods are guided by the past judgments or by estimates of relevance in the upcoming judgments. We also proposed an innovative way to estimate recall, based on training queries, and we employed this estimate to plot a curve of F versus rank. Tracking this curve with indicators derived from financial trading led to very effective stopping methods. Some of them only required to judge $5 - 7\%$ of the pooled documents, and the resulting qrels ranked retrieval systems in a very accurate way. Furthermore, the new approach to estimate recall and to track the curve of estimated F is potentially useful in other areas beyond IR evaluation.

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 investigación de Galicia") and ED431G/08.

We also thank the anonymous reviewers for their careful reading of our paper and their insightful suggestions, which helped us to improve the quality of this research.

References

Arampatzis, A., Kamps, J. & Robertson, S. (2009). Where to stop reading a ranked list?: Threshold optimization using truncated score distributions. In *Proc. ACM SIGIR* (pp. 524–531).

Aslam, J. A., Pavlu, V. & Savell, R. (2003). A unified model for metasearch, pooling, and system evaluation. In *Proc. ACM CIKM* (pp. 484–491).

Aslam, J. A., Pavlu, V. & Yilmaz, E. (2006). A statistical method for system evaluation using incomplete judgments. In *Proc. ACM SIGIR* (pp. 541–548).

Balog, K. & Neumayer, R. (2013). A test collection for entity search in dbpedia. In *Proc. ACM SIGIR* (pp. 737–740).

Bodoff, D. & Li, P. (2007). Test theory for assessing ir test collections. In *Proc. ACM SIGIR* (pp. 367–374).

Büttcher, S., Clarke, C., Yeung, P. & Soboroff, I. (2007). Reliable information retrieval evaluation with incomplete and biased judgements. In *Proc. ACM SIGIR* (pp. 63–70).

Carterette, B. (2007). Robust test collections for retrieval evaluation. In *Proc. ACM SIGIR* (pp. 55–62).

Carterette., B. (2008). *Low-cost and robust evaluation of information retrieval systems* (Unpublished doctoral dissertation). University of Massachusetts Amherst.

Carterette, B., Allan, J. & Sitaraman, R. (2006). Minimal test collections for retrieval evaluation. In *Proc. ACM SIGIR* (pp. 268–275).

Carterette, B., Pavlu, V., Kanoulas, E., Aslam, J. A. & Allan, J. (2008). Evaluation over thousands of queries. In *Proc. ACM SIGIR* (pp. 651–658).

Cootner, P. H. (1962). Stock prices: Random vs. systematic changes. *Industrial Management Review*, *3*, 24–45.

Cormack, G. & Grossman, M. (2016). Engineering quality and reliability in technology-assisted review. In *Proc. ACM SIGIR* (pp. 75–84).

Cormack, G. & Lynam, T. (2007). Power and bias of subset pooling strategies. In *Proc. ACM SIGIR* (pp. 837–838).

Cormack, G., Palmer, C. & Clarke, C. (1998). Efficient construction of large test collections. In *Proc. ACM SIGIR* (pp. 282–289).

Kando, N., Sakai, T. & Sanderson, M. (Eds.). (2016). *Proc. 12th NTCIR Conference on Evaluation of Information Access Technologies.*

Losada, D. & Crestani, F. (2016). A test collection for research on depression and language use. In *Proc. CLEF* (pp. 28–39).

Losada, D., Parapar, J. & Barreiro, A. (2016). Feeling lucky? multi-armed bandits for ordering judgements in pooling-based evaluation. In *Proc. ACM SAC* (pp. 1027–1034).

Losada, D., Parapar, J. & Barreiro, A. (2017). Multi-armed bandits for adjudicating documents in pooling-based evaluation of information retrieval systems. *Information Processing & Management*, *53*(3), 1005–1025.

Losada, D., Parapar, J. & Barreiro, A. (2018). A rank fusion approach based on score distributions for prioritizing relevance assessments in information retrieval evaluation. *Information Fusion*(39), 56–71.

Lu, X., Moffat, A. & Culpepper, J. (2016). The effect of pooling and evaluation depth on IR metrics. *Information Retrieval Journal*, *19*(4), 416–445.

Moffat, A., Webber, W. & Zobel, J. (2007). Strategic system comparisons via targeted relevance judgments. In *Proc. ACM SIGIR* (pp. 375–382).

Parapar, J., Presedo-Quindimil, M. A. & Barreiro, A. (2014). Score distributions for pseudo relevance feedback. *Information Sciences*, *273*, 171 - 181.

Sakai, T., Kando, N., Lin, C.-J., Mitamura, T., Shima, H., Ji, D., ... Nyberg, E. (2008). Overview of the NTCIR-7 ACLIA IR4QA task. In *Proc. NTCIR*.

Sakai, T. & Lin, C. (2010). Ranking retrieval systems without relevance assessments – revisited. In *Proc. 3rd International Workshop on Evaluating Information Access*. Tokyo, Japan.

Sanderson, M. & Zobel, J. (2005). Information retrieval system evaluation: Effort, sensitivity, and reliability. In *Proc. ACM SIGIR* (pp. 162–169).

Voorhees, E. (2000). Variations in relevance judgments and the measurement of retrieval effectiveness. *Information Processing & Management*, *36*(5), 697 - 716.

Voorhees, E. (2001). Evaluation by highly relevant documents. In *Proc. ACM SIGIR* (pp. 74–82).

Voorhees, E. & Harman, D. (2005). *TREC: Experiment and evaluation in information retrieval*. The MIT Press.

Webber, W., Moffat, A. & Zobel, J. (2008). Statistical power in retrieval experimentation. In *Proc. ACM CIKM* (pp. 571–580).

Yilmaz, E., Aslam, J. A. & Robertson, S. (2008). A new rank correlation coefficient for information retrieval. In *Proc. ACM SIGIR* (pp. 587–594).

Zhai, C., Jansen, P. & Evans, D. (2000). Exploration of a heuristic approach to threshold learning in adaptive filtering (poster session). In *Proc. ACM SIGIR* (pp. 360–362).

Zobel, J. (1998). How reliable are the results of large-scale information retrieval experiments? In *Proc. ACM SIGIR* (pp. 307–314).