A Rank Fusion Approach based on Score Distributions for Prioritizing Relevance Assessments in Information Retrieval Evaluation

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

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

Abstract

In this paper we study how to prioritize relevance assessments in the process of creating an Information Retrieval test collection. A test collection consists of a set of queries, a document collection, and a set of relevance assessments. For each query, only a sample of documents from the collection can be manually assessed for relevance. Multiple retrieval strategies are typically used to obtain such sample of documents. And rank fusion plays a fundamental role in creating the sample by combining multiple search results. We propose effective rank fusion models that are adapted to the characteristics of this evaluation task. Our models are based on the distribution of retrieval scores supplied by the search systems and our experiments show that this formal approach leads to natural and competitive solutions when compared to state of the art methods. We also demonstrate the benefits of including pseudo-relevance evidence into the estimation of the score distribution models.

Keywords: Rank Fusion, Information Retrieval, Evaluation, Pooling, Score Distributions, Pseudo-relevance

1. Introduction

Evaluation is crucial to making progress in science. In data intensive disciplines, creating the *gold standard* is often a major bottleneck in the process of building a test collection or benchmark for evaluation. Gold standards –also known as ground truth– are typically produced by humans and, therefore, they are expensive. A case in point is the creation of Information Retrieval (IR) test collections for evaluating search algorithms. Given a set of test queries, representing different information needs, and a large collection of documents, we would like to have exhaustive judgments (relevance information on every query-document pair). But this is unfeasible with current large-scale collections. For each query, we can only afford to judge a selected sample of documents from the collection. It is standard practice to run each query against multiple search engines, fuse the rankings supplied, and manually assess for relevance the documents at the top of the list. Focusing the judgment effort on these selected documents makes the most of human assessors' time and leads to pools of judged documents that can be reliably used to compare retrieval strategies [41].

With multiple search systems contributing to the evaluation task, rank fusion becomes an essential component. An effective rank fusion strategy leads to high counts of relevant documents in the judged set of documents and, consequently, to a robust benchmark. However, most fusion methods employed to date for ranking to-be-judged documents are rather simplistic. We claim that the process of prioritization of to-be-judged documents should take into account all available evidence. In particular, most search systems participating into a given evaluation initiative supply scores that measure the degree of relevance between documents and test queries. Modeling the distributions of these scores has found to be effective for a broad range of tasks [3] but it was never used for rank fusion in evaluating IR systems. We show here that fusion models based on Score Distributions (SDs) lead to highly competitive methods for allocating documents for judgment.

We define and experiment with two main classes of prioritization strategies: i) *static methods*, which build a merged ranking of documents that remains unchanged during the whole assessment process, and ii) *dynamic methods*, where the priority of documents changes as we obtain relevance assessments. Dynamic methods fit well with SD models because we can update the estimations of the distributions of relevant and non-relevant documents after each assessment. This iterative update of distributions is a natural consequence of employing SD models for ranking to-be-judged documents. Furthermore, we propose an innovative way to estimate the initial score distributions of relevant and non-relevant documents for each search system. In the absence of relevance information, it is customary to build these two distributions using only the list of scores supplied by each system. But we argue that pseudo-relevance information can be inferred from the list of available rankings, and such pseudo-relevance evidence can be effectively used for initializing the SD models. Initializing SD models of metasearch in this way is novel, and our experiments demonstrate that incorporating pseudo-relevance information is beneficial.

This paper addresses the following research questions:

- Are rank fusion models based on SDs effective for prioritizing assessments in the context of search system evaluation? How do they compare with state-of-the-art static and dynamic prioritization strategies?
- How can we use pseudo-relevance information to estimate the initial score distributions of relevant and non-relevant documents? Does the incorporation of pseudo-relevance information into SD models lead to improved models?

The rest of the paper is organized as follows. Section 2 presents models of score distributions that have been employed in different Information Access tasks, and section 3 explains our proposal to use SD models for rank fusion in pooling-based evaluation of IR systems. The experiments are reported in section 4 and section 5 offers some discussing remarks. Section 6 reviews some studies that are related to our research. The paper ends with some conclusions and future lines of work.

2. Modeling Score Distributions of Search Systems

Given a user query most retrieval systems calculate a score per document that measures the degree of relevance to the query. These scores are employed for ranking retrieved documents, and their range and distribution varies across different systems. Score distributions have been effectively modeled in multiple Information Access areas, such as Information Filtering or Distributed Information Retrieval. For instance, Manmatha and colleagues [31] exploited score distributions for combining the outputs of different search engines (meta-search problem). Arampatzis and his colleagues [1, 2] formulated the threshold optimization problem and worked with score distributions models for locating a good cut-off point in a legal search task. Other researchers have applied score distributions to tasks such as query performance prediction [17], image retrieval [4] and pseudo-relevance feedback [36].

Under binary relevance, the score distributions on a per query basis may be fitted as a mixture of two distributions: one for relevant documents and another one for non-relevant documents. This mixture model is used to map the scores to probabilities. This formal modeling process is essential in Information Fusion, Metasearch, Filtering or Thresholding. SD models have been shown to work for a large number of retrieval systems, particularly for those contributing to well-known IR evaluation campaigns like TREC [49].

A number of modeling alternatives have been explored in the literature. Various combinations of distributions have been employed, but we will focus on a combination of two Log-Normal distributions[1], which is a general and consistent approach for preserving relevance information across a variety of search systems [16]. This mixture follows the recall-fallout hypothesis [39] and offers better goodness of fit than other alternatives [16, 18]. A full comparison of different models was performed by Cummins [16]. He studied different combinations of distributions and concluded that a mixture of two Log-Normal distributions is the best performing model. His study considered the Normal distribution, the Log-Normal distribution and the Gamma distribution for modeling the scores of relevant documents; and the Exponential distribution, the Normal distribution, the Log-Normal distribution and the Gamma distribution for modeling the scores of non-relevant documents.

The document score distributions are modeled as a mixture of relevant and non-relevant distributions as follows:

$$p(score) \quad = \quad \lambda \cdot p(score|rel) + (1-\lambda) \cdot p(score|nonrel) \tag{1}$$

where $\lambda \in [0, 1]$ is the mixing weight, and $p(.|rel)$ ($p(.|nonrel)$) is the probability density function of the relevant (non-relevant) distribution. This two-component mixture model makes explicit that i) each score in a ranked list is associated to a document that is either relevant or non-relevant, and ii) the distribution of scores in relevant and non-relevant documents are not necessarily the same (the separation into two components, $p(.|rel)$ and $p(.|nonrel)$, permits to make a distinction between the pattern of scores of relevant and non-relevant documents).

Modeling the scores with two Log-Normal distribution leads to:

$$p(score|rel) \quad = \quad \frac{1}{score \cdot \sigma_{rel} \cdot \sqrt{2\pi}} \cdot e^{-\frac{(ln\ score\ -\ \mu_{rel})^2}{2\sigma_{rel}^2}}, score > 0 \tag{2}$$

$$p(score|nonrel) \quad = \quad \frac{1}{score \cdot \sigma_{nonrel} \cdot \sqrt{2\pi}} \cdot e^{-\frac{(ln\ score\ -\ \mu_{nonrel})^2}{2\sigma_{nonrel}^2}}, score > 0 \tag{3}$$

where μ_{rel} and σ_{rel} (resp. μ_{nonrel}, σ_{nonrel}) are the parameters of the Log-Normal distribution. Note that scores need to be positive[2]. Log-Normal distributions have

[1]The Log-Normal distribution is a continuous distribution of a random variable whose logarithm has a Normal Distribution.

[2]The occurrence of negative scores can be overcome by shifting all scores by some constant factor. It is standard practice to make first this normalization and, next, apply the SD models. For systems that supply negative scores, we adjust each score s as follows: $s = s - min(scores) + 1$, where $min(scores)$ is the minimum score computed by the system.

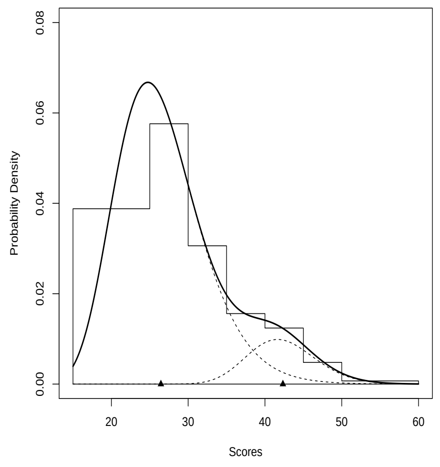

Figure 1: A mixture of two Log-Normals fitting the scores of a retrieval system. The histogram represents the distribution of scores of the system, the dashed line on the left is the Log-Normal associated to non-relevant documents (peak centered at a low score), the dashed line on the right is the Log-Normal associated to relevant documents (peak centered at a high score), and the solid line is the mixture of both Log-Normals.

shown to be a good fit for modeling the scores of multiple retrieval systems [16]. Figure 1 shows an example of two Log-Normal distributions (dashed lines) that have been fitted from the scores of relevant and non-relevant documents computed by a retrieval system in response to a query. The dashed line on the right (Log-Normal distribution with the highest mean of scores) is the distribution associated to relevant documents. The dashed line on the left (Log-Normal distribution with the lowest mean of scores) is the distribution associated to non-relevant documents (retrieval systems tend to give higher scores to relevant documents and lower scores to non-relevant documents). The figure also plots the resulting mixture of the two Log-Normals (solid line) and the histogram representing the distribution of all scores.

In the absence of relevance data, a standard approach to estimate the mixing parameter and the parameters of the component Log-Normals is to run the Expected Maximization algorithm [19], which quickly converges to a solution.

Once the distributions are fitted, Bayes' rule can be employed to compute the probability of relevance given any score:

$$p(rel|score) = \frac{p(score|rel) \cdot p(rel)}{p(score|rel) \cdot p(rel) + p(score|nonrel) \cdot p(nonrel)} \quad (4)$$

where $p(rel)$ is set to the fitted λ, and $p(nonrel)$ is set to $1 - \lambda$. This estimated probability, $p(rel|score)$, can be used in a number of ways. For instance, it has been used for threshold optimization in recall-oriented retrieval [2] and pseudo-relevance feedback [36].

3. Score Distributions for Rank Fusion in Pooling-based Evaluation of Search Systems

Let us now discuss our proposal to employ SDs in pooling-based evaluation of search systems. An IR test collection consists of a document collection, a set of test queries, and a set of relevance judgments. The first two components, documents and queries, are often easy to obtain. The third component is of critical importance, but it is costly because it requires human intervention. Currently, full relevance judgments are not an option. The pooling method is a standard approach to create relevance judgments. It is based on making relevance judgments on a sample of documents selected by various retrieval systems. For each test query, documents are adjudicated for judgment as follows. Each participating system supplies a ranking of documents in decreasing order of estimated relevance. We are concerned here with systems that explicitly compute a score for each retrieved document and, therefore, we constrain our discussion to rankings of documents where the associated relevance score is available. The availability of scores is the norm in many evaluation campaigns (for example, in TREC). Figure 2 shows an example with m rankings of estimated relevance to a query. Each $d_{s,j}$ stores a unique identifier for the document retrieved at the *jth* position by $system_s$ (e.g. in web retrieval, $d_{s,j}$ could store the document's url). Multiple systems can retrieve the same document at different positions. For instance, $d_{3,1}$ can be equal to $d_{1,2}$ (the top-ranked document of the third system is the same as the 2-nd ranked document of the first system); but the associated retrieval scores ($s_{3,1}$ and $s_{1,2}$) are not necessarily the same (each system implements its own retrieval scoring method).

$system_1$	$system_2$	$\ldots$	$system_m$
$1.d_{1,1}(s_{1,1})$	$1.d_{2,1}(s_{2,1})$	$\ldots$	$1.d_{m,1}(s_{m,1})$
$2.d_{1,2}(s_{1,2})$	$2.d_{2,2}(s_{2,2})$	$\ldots$	$2.d_{m,2}(s_{m,2})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Figure 2: Rankings of estimated relevance computed by m different retrieval systems. $d_{s,j}$ is the document retrieved at the *jth* position by $system_s$ and $s_{s,j}$ is the score assigned by $system_s$ to that document.

The set of documents that are candidate for judgment is built by union of the top k documents across all rankings (most often $k = 100$ or $k = 50$). Given the *pool depth*, k, the pool is defined as:

$$pool_k \quad = \bigcup_{1 \le s \le m, 1 \le j \le k} \left\{ d_{s,j} \right\} \tag{5}$$

We can define a document adjudication strategy as a method that takes the set of system's rankings and the pool depth (k) as inputs and returns a sequence of documents to be judged for relevance ($PoolOrder_k$). $PoolOrder_k$ is a permutation of the pooled documents: $PoolOrder_k \in Perm(pool_k)$. If we are going to judge the whole pool then any permutation leads to the same set of relevance assessments. But judging a subset of the pool is often preferred. Subset pooling methods do fewer judgments per query by only assessing selected samples from the pool (for example, the first items in $PoolOrder_k$). This strategy reduces the assessment effort and has been adopted in evaluation exercises like INEX [24], in the TREC filtering track [45], in Speech Retrieval [35], or in the Million Query TREC Track [10]. Many studies have demonstrated that high quality and reusable benchmarks can be constructed by doing a reduced set of relevance judgments per query [32, 7, 9, 14, 8].

With *static* adjudication, a full ordering of the pool is determined prior to the beginning of the assessment process, and this ordering is not altered thereafter. With *dynamic* adjudication, documents are iteratively selected from the pool, each document's relevance is assessed, and this relevance information affects the decision on the next extraction. Dynamic methods use the most information and can quickly adapt to the outcome of the assessments done so far. For instance, we can dynamically avoid poor retrieval systems; or we can re-rank the remaining unjudged documents. These adaptive strategies can improve our chances of early finding relevant documents. But dynamic methods put an additional burden on building the benchmark [29]. The whole process needs to be coordinated such that the assessment of the next document cannot start until the previous assessment has finished. This coordination complicates the evaluation exercise because the time for doing an assessment varies with factors such as the cognitive abilities of the assessors, or their workload and scheduling. Furthermore, standard speed-up strategies, based on parallel distribution across multiple assessors of the documents retrieved for the same query, cannot be implemented. In our study, we propose and experiment with different static and dynamic methods. As argued above, dynamic methods are more effective at early identifying relevant items but they cannot always be put in practice.

In general, we will prefer adjudication methods that supply sequences of judgments with many relevant documents at the beginning. A common way to compare adjudication methods is to plot the number of documents judged as relevant against the number of assessments done. An optimal method would choose all relevant documents first, and all non-relevant documents would be located at the end of the sequence. With an effective adjudication method, we could stop doing assessments after extracting a sufficient number of relevant items.

3.1. Static Adjudication Methods based on Score Distribution Models

Static adjudication methods induce, for each test query, an order on the pooled documents. Once produced, this order is never changed. Recall that the query is first sent to the participating systems and each system provides a ranking of documents by estimated relevance to the query. The input of the adjudication process is, therefore, multiple rankings of documents.

3.1.1. Rank fusion with Score Distribution Models

Our fusion proposal to create an order on the pooled documents works as follows. Given the input rankings, we fit an individual SD model for each ranking. As argued in section 2, this fitting process takes the sequence of scores generated by the retrieval system as an input and estimates two Log-Normal distributions and a mixing weight. We performed this estimation task with the mixdist package for R, which contains functions for fitting mixture models to data by maximum likelihood using a combination of Newton-type algorithms and the EM algorithm[3]. The static methods do this fitting once, and the resulting parameters are not altered thereafter.

A document is often retrieved by many retrieval systems (at different rank positions and with different retrieval scores assigned). Given the document score assigned by each system, and the probabilities $p(score|rel)$, $p(score|nonrel)$, $p(rel)$ and $p(nonrel)$ –learnt with the SD model of the system–, the probability $p(rel|score)$ can be computed with eq. 4. A document has multiple $p(rel|score)$ probabilities (one for each retrieval system, and computed from the document score and the SD model of the system), and a natural way to combine these probabilities is to average them. This simple strategy has been successfully employed for metasearch [31]. Observe that the approach is static: the systems' fits are done before beginning the assessment process, the averaged probabilities are used for ranking the pooled documents, and the resulting ranking of pooled documents is not altered thereafter. Algorithm 1 sketches the whole process. The first for loop computes the fits for all retrieval systems and the second for loop goes over the pooled documents and computes the average score from the posterior distributions (the inner for loop computes and accumulates $p(rel|score)$ for each system-document pair).

[3]Version 0.5-4, `https://cran.r-project.org/web/packages/mixdist/mixdist.pdf`.

Algorithm 1: Static Adjudication Method based on SD (no pseudo-qrels)

```
input     : $m$ rankings produced by $m$ systems: $\langle ranking_1, \ldots, ranking_m \rangle$,
            $ranking_s = \langle d_{s,1}(s_{s,1}), d_{s,2}(s_{s,2}), \ldots \rangle$, $d_{s,j}$: doc retrieved at the $jth$
            position by $system_s$, $s_{s,j}$: score assigned by $system_s$ to that doc.
output    : an order of the pooled documents
parameter: $k$ (pool depth)

foreach $s \in \{1, .., m\}$ do
    build a SD model for $system_s$ using mix (R's mixdist package)
    (mix takes the scores $s_{s,1}, s_{s,2}, \ldots$ and outputs $\lambda_s, \mu_{rel_s}, \sigma_{rel_s}, \mu_{nonrel_s}, \sigma_{nonrel_s}$)
$pool_k \leftarrow \bigcup_{1 \le s \le m, 1 \le j \le k} \{d_{s,j}\}$
foreach $d \in pool_k$ do
    $score_d \leftarrow 0$
    foreach $s$ that retrieved $d$ do
        $score \leftarrow$ score assigned to $d$ by $s$
        $psr \leftarrow \frac{1}{score \cdot \sigma_{rel_s} \cdot \sqrt{2\pi}} \cdot e^{-\frac{(ln\ score\ -\ \mu_{rel_s})^2}{2\sigma^2_{rel_s}}}$
        $psnr \leftarrow \frac{1}{score \cdot \sigma_{nonrel_s} \cdot \sqrt{2\pi}} \cdot e^{-\frac{(ln\ score\ -\ \mu_{nonrel_s})^2}{2\sigma^2_{nonrel_s}}}$
        $prs \leftarrow (psr \cdot \lambda_s)/(psr \cdot \lambda_s + psnr \cdot (1 - \lambda_s))$
        $score_d \leftarrow score_d + prs$
    $score_d \leftarrow score_d / m$
order the docs of $pool_k$ by decreasing $score_d$
return the resulting ordered list
```

3.1.2. Improving Score Distribution Models with Pseudo-Relevance Evidence

The existence of multiple rankings supplies a valuable source of evidence that can be used to improve the SD models. More specifically, we propose to estimate pseudo-relevance judgments from the rankings of retrieved documents and use them to build better fits of scores. This innovative proposal has the potential to be of interest beyond IR evaluation. Pseudo-relevance judgments have been proposed in the past for evaluating retrieval systems without relevance assessments [44, 40]. But system evaluation with no relevance assessments does not predict well the performance of some systems (particularly the top systems), and we claim that pseudo-relevance judgments are best utilized if combined with SD models. Our innovative proposal consists of employing pseudo-relevance judgments for improving the initial fits and, next, we use the ranking induced by the SD-based approach for prioritizing judgments. This approach can be seen as an intermediate solution between no judgments at all and judging the whole pool. First, we present the method implemented to estimate pseudo-relevance judgments and, next, we proceed to explain how we use the pseudo-relevance judgments to feed the SD models.

Some methods of forming pseudo-relevance judgments (“pseudo-qrels”) have been introduced in the past. An extensive experimental study can be found in [40]. We experimented with the method proposed by Soboroff and colleagues [44], the method proposed by Sakai & Lin [40], and a method based on Condorcet proposed by Nuray & Can [34]. We found no substantial difference among them and, therefore, we constrain our discussion to Soboroff’s method, which is the one we used for experimentation. Soboroff’s method works as follows. Given multiple rankings of documents, the extraction of pseudo-relevant documents is centered on the top 30 documents. The method randomly samples 10% of the depth-30 pool (containing duplicates) and treats the extracted samples as relevant documents. This simple and efficient approach is at least as effective as other more sophisticated methods.

Given the pseudo-qrels, we estimated the SD models as follows. For each system, the scores of the documents estimated as relevant by Soboroff’s method were collected and fitted to a Log-Normal distribution. We performed this fitting with the fitdistr function of the MASS package (R). This function implements maximum likelihood fitting of some univariate distributions, including the Log-Normal distribution[4]. We followed the same procedure with the scores of the documents estimated as non-relevant. Next, we obtained a mixture of two Log-Normals by combining both distributions (with a mixing weight computed as the proportion

[4]https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/fitdistr.html

of pseudo-relevant documents). Observe that we obtain a mixture of two Log-Normals. So, the estimation process with pseudo-qrels is different to the one with no pseudo-qrels, but the form of the resulting SD models is the same (a mixture of two Log-Normal distributions). This homogeneity is a convenient feature of our approach. As done previously for the no pseudo-qrels case, the last step consists of using the estimated SD models for building a ranking of pooled documents by averaging the posterior probabilities. In the following, "SD (no pseudo-qrels)" refers to the standard SD estimations, based only on the scores of the retrieved documents, while "SD+pseudo-qrels" refers to SD models estimated from pseudo-relevance judgments. Algorithm 2 sketches the whole process. The first three lines of the main block implement the computation of the pseudo-relevance judgments (following Soboroff's method). The first for loop computes the SD models for all systems (invoking the fit_distrib function, which implements the fitting). The last for loop assigns a score for each document in the pool (invoking the doc_score function, which computes and averages the posterior probabilities).

3.2. Dynamic Adjudication Methods based on Score Distribution Models

Let us now explain our SD-based proposal for dynamically adjudicating documents for judgment. Dynamic adjudication methods do not induce, for each test query, a static order on the pooled documents. A dynamic method selects a document from the pool, sends it for relevance assessment, uses the resulting assessment to update the internal state of the algorithm and, finally, makes the next pick based on the updated state. The process continues iteratively until all pooled documents have been judged.

The characteristics of dynamic adjudication fit well with our SD-based approach: we can iteratively update the SD models as assessments come in. First, we build an initial ranking of the pooled documents. This initial ranking is the same that would be produced by the static method with pseudo-qrels: we build multiple SD models from pseudo-relevance assessments (one for each system), and we construct an initial ranking of pooled documents from the posterior distributions. Next, the assessment process begins: the top document is extracted and judged for relevance. If the relevance value of the document assigned by the human assessor and the relevance value assumed by the pseudo-qrels is not the same then we need to: a) update the pseudo-qrels, b) recompute the SD models with the updated pseudo-qrels, and c) re-rank the remaining unjudged documents. Recomputing the SD models means that for each system we re-run the fitting (using the fitdistr function with the updated pseudo-qrels) and adjust the mixing weight (the proportion of pseudo-relevant documents changes). If there is no conflict between the human assessment and the pseudo-qrels then we make no changes and just proceed to judge the next top ranked document. As the process runs, we have more assessments

Algorithm 2: Static Adjudication Method based on SD (with pseudo-qrels)

```
input     : $m$ rankings produced by $m$ systems: $\langle ranking_1, \ldots, ranking_m \rangle$,
            $ranking_s = \langle d_{s,1}(s_{s,1}), d_{s,2}(s_{s,2}), \ldots \rangle$, $d_{s,j}$: doc retrieved at the $jth$
            position by $system_s$, $s_{s,j}$: score assigned by $system_s$ to that doc.
output    : an order of the pooled documents
parameter : $k$ (pool depth)

function $fit\_distrib$ $(scores\_rel, scores\_nonrel)$
    $\lambda \leftarrow |scores\_rel| / (|scores\_rel| + |scores\_nonrel|)$
    fit $scores\_rel$ to a Log-Normal distribution (using fitdistr from R)
      (fitdistr takes $scores\_rel$ and outputs $\mu_{rel}$, $\sigma_{rel}$)
    fit $scores\_nonrel$ to a Log-Normal distribution (using fitdistr from R)
      (fitdistr takes $scores\_nonrel$ and outputs $\mu_{nonrel}$, $\sigma_{nonrel}$)
    return $\lambda, \mu_{rel}, \sigma_{rel}, \mu_{nonrel}, \sigma_{nonrel}$
function $doc\_score(d, \langle ranking_s \rangle_{1 \leq s \leq m}, \langle (\lambda_s, \mu_{rel_s}, \sigma_{rel_s}, \mu_{nonrel_s}, \sigma_{nonrel_s}) \rangle_{1 \leq s \leq m})$
    $score_d \leftarrow 0$
    foreach $s$ that retrieved $d$ do
        $score \leftarrow$ score assigned to $d$ by $s$
        $psr \leftarrow \frac{1}{score \cdot \sigma_{rel_s} \cdot \sqrt{2\pi}} \cdot e^{-\frac{(ln\ score\ -\ \mu_{rel_s})^2}{2\sigma^2_{rel_s}}}$
        $psnr \leftarrow \frac{1}{score \cdot \sigma_{nonrel_s} \cdot \sqrt{2\pi}} \cdot e^{-\frac{(ln\ score\ -\ \mu_{nonrel_s})^2}{2\sigma^2_{nonrel_s}}}$
        $prs \leftarrow (psr \cdot \lambda_s)/(psr \cdot \lambda_s + psnr \cdot (1 - \lambda_s))$
        $score_d \leftarrow score_d + prs$
    return $score_d / m$

main
    $candidate\_pos \leftarrow \{(1,1), \ldots, (1,30), \ldots, (m,1), \ldots, (m,30)\}$
    $sample \leftarrow$ random sample of 10% of the pairs from $candidate\_pos$
    $pseudo\_rel\_docs \leftarrow \bigcup_{(i,j) \in sample} \left\{ d_{i,j} \right\}$
    foreach $s \in \{1, .., m\}$ do
        $scores\_rel \leftarrow \langle s_{s,j} \rangle \ \forall_j$ such that $d_{s,j} \in pseudo\_rel\_docs$
        $scores\_nonrel \leftarrow \langle s_{s,j} \rangle \ \forall_j$ such that $d_{s,j} \notin pseudo\_rel\_docs$
        $\lambda_s, \mu_{rel_s}, \sigma_{rel_s}, \mu_{nonrel_s}, \sigma_{nonrel_s} \leftarrow fit\_distrib(scores\_rel, scores\_nonrel)$
    $pool_k \leftarrow \bigcup_{1 \leq s \leq m, 1 \leq j \leq k} \left\{ d_{s,j} \right\}$
    foreach $d \in pool_k$ do
        $s_d \leftarrow doc\_score(d, \langle ranking_s \rangle_{1 \leq s \leq m}, \langle (\lambda_s, \mu_{rel_s}, \sigma_{rel_s}, \mu_{nonrel_s}, \sigma_{nonrel_s}) \rangle_{1 \leq s \leq m})$
    order the docs of $pool_k$ by decreasing $s_d$
    return the resulting ordered list
```

and, therefore, the SD models become more reliable (we increasingly move from estimated relevance to human assessed relevance).

Algorithm 3 describes in detail our dynamic method. The only difference between Algorithm 3 and Algorithm 2 is that, now, we include a while loop after assigning the document's scores (last part of the main function). This loop implements the assessment process: we iteratively select the unjudged pooled document with the highest score, obtain the relevance assessment for this document, and, if needed, update the pseudo-qrels, the SD models and the document's scores.

4. Experiments

We performed a series of experiments to evaluate our static and dynamic adjudication methods, and we report here a comparison of their performance against existing methods to prioritize judgments.

We worked with four standard retrieval benchmarks from the well-known Text Retrieval Conference (TREC). These four collections (TREC5 to TREC8) were built under the TREC adhoc track, which is a search task where the participating systems are given a set of queries, a document collection, and they are asked to rank documents from the collection for every query. Each test collection contains 50 queries. For each query, the organizers of TREC kindly provided us with the rankings produced by the retrieval systems that participated in the evaluation campaign. Our proposal is oriented towards the analysis of retrieval scores and, therefore, we discarded those rankings that had no scores assigned to the retrieved documents. Table 1 presents the main statistics of the four test collections. For each collection, the number of retrieval systems refers to the number of retrieval systems that supply scores for the retrieved documents.

The evaluation was done on a query by query basis as follows. Given the input rankings, every static or dynamic adjudication method induces an order on the list of documents that are candidates for judgment ($pool_{100}$). We evaluated each adjudication method in terms of its ability to identify relevant documents. To meet this aim, we employed the actual judgments from the official assessments of the track (in these four campaigns all pooled documents were assessed for relevance by humans). We analyzed the sequence of judgments induced by each adjudication method in groups of 100 judgments, by computing the number of relevant documents found in each group and accumulating the counts (relevant documents found after 100 judgments, relevant documents found after 200 judgments, and so forth). We computed these accumulated statistics for each query and, next, we obtained a global accumulative plot by group-level averaging over queries (average number of relevant documents found after 100 judgments, average number of relevant documents found after 200 judgments, and so forth).

Algorithm 3: Dynamic Adjudication Method based on SDs and pseudo-qrels

```
input     : $m$ rankings produced by $m$ systems: $\langle ranking_1, \ldots, ranking_m \rangle$,
            $ranking_s = \langle d_{s,1}(s_{s,1}), d_{s,2}(s_{s,2}), \ldots \rangle$, $d_{s,j}$: doc retrieved at the $jth$
            position by $system_s$, $s_{s,j}$: score assigned by $system_s$ to that doc.
output    : a sequence of assessments (qrels)
parameter: $k$ (pool depth)
function fit_distrib (scores_rel, scores_nonrel)
    $\lambda \leftarrow |scores\_rel| / (|scores\_rel| + |scores\_nonrel|)$
    fit $scores\_rel$ to a Log-Normal distribution (using fitdistr from R)
      (fitdistr takes $scores\_rel$ and outputs $\mu_{rel}$, $\sigma_{rel}$)
    fit $scores\_nonrel$ to a Log-Normal distribution (using fitdistr from R)
      (fitdistr takes $scores\_nonrel$ and outputs $\mu_{nonrel}$, $\sigma_{nonrel}$)
    return $\lambda, \mu_{rel}, \sigma_{rel}, \mu_{nonrel}, \sigma_{nonrel}$
function doc_score($d, \langle ranking_s \rangle_{1 \le s \le m}$, $\langle (\lambda_s, \mu_{rel_s}, \sigma_{rel_s}, \mu_{nonrel_s}, \sigma_{nonrel_s}) \rangle_{1 \le s \le m}$)
    $score_d \leftarrow 0$
    foreach $s$ that retrieved $d$ do
        $score \leftarrow$ score assigned to $d$ by $s$
        $psr \leftarrow \frac{1}{score \cdot \sigma_{rel_s} \cdot \sqrt{2\pi}} \cdot e^{-\frac{(ln\ score\ -\ \mu_{rel_s})^2}{2\sigma^2_{rel_s}}}$
        $psnr \leftarrow \frac{1}{score \cdot \sigma_{nonrel_s} \cdot \sqrt{2\pi}} \cdot e^{-\frac{(ln\ score\ -\ \mu_{nonrel_s})^2}{2\sigma^2_{nonrel_s}}}$
        $prs \leftarrow (psr \cdot \lambda_s) / (psr \cdot \lambda_s + psnr \cdot (1 - \lambda_s))$
        $score_d \leftarrow score_d + prs$
    return $score_d / m$
main
    $candidate\_pos \leftarrow \{(1,1), \ldots, (1,30), \ldots, (m,1), \ldots, (m,30)\}$
    $sample \leftarrow$ random sample of 10% of the pairs from $candidate\_pos$
    $pseudo\_rel\_docs \leftarrow \bigcup_{(i,j) \in sample} \{d_{i,j}\}$
    foreach $s \in \{1, .., m\}$ do
        $scores\_rel \leftarrow \langle s_{s,j} \rangle\ \forall_j$ such that $d_{s,j} \in pseudo\_rel\_docs$
        $scores\_nonrel \leftarrow \langle s_{s,j} \rangle\ \forall_j$ such that $d_{s,j} \notin pseudo\_rel\_docs$
        $\lambda_s, \mu_{rel_s}, \sigma_{rel_s}, \mu_{nonrel_s}, \sigma_{nonrel_s} \leftarrow fit\_distrib(scores\_rel, scores\_nonrel)$
    $pool_k \leftarrow \bigcup_{1 \le s \le m, 1 \le j \le k} \{d_{s,j}\}$
    foreach $d \in pool_k$ do
        $s_d \leftarrow doc\_score(d, \langle ranking_s \rangle_{1 \le s \le m}, \langle (\lambda_s, \mu_{rel_s}, \sigma_{rel_s}, \mu_{nonrel_s}, \sigma_{nonrel_s}) \rangle_{1 \le s \le m})$
    while $pool_k \neq \emptyset$ do
        $d_{max} \leftarrow \arg\max_{d \in pool_k} s_d$
        Assess the relevance of $d_{max}$
        $pool_k \leftarrow pool_k \setminus \{d_{max}\}$
        if relevance value of $d_{max}$ is not equal to the assumed relevance then
            update $pseudo\_rel\_docs$
            recompute $\langle (\lambda_s, \mu_{rel_s}, \sigma_{rel_s}, \mu_{nonrel_s}, \sigma_{nonrel_s}) \rangle_{1 \le s \le m}$ (calls to $fit\_distrib$)
            recompute $\langle s_d \rangle_{d \in pool_k}$ (calls to $doc\_score$)
```

	TREC5	TREC6	TREC7	TREC8
# queries	50	50	50	50
# retrieval systems	38	48	62	88
avg size of pool_{100} (# docs)	950.6	1278.9	1092.1	1342.6
min size of pool_{100} (# docs)	284	457	608	712
max size of pool_{100} (# docs)	1939	2165	1778	3173
avg. # rel docs in pool_{100}	71.8 (7.5%)	74.4 (5.8%)	70.3 (6.4%)	75.7 (5.6%)

Table 1: Statistics of the test collections

To facilitate reproducibility, the R code implementing our proposed solutions and the baseline methods is publicly available. Full instructions on how to run the code against standard test collections can be found at our website[5].

4.1. Static methods

First, we experimented with a number of static methods that have been proposed in the past. Some of them have been traditionally employed in IR evaluation (e.g. DocID or Moffat's methods) while other methods are well-known rank aggregation (or *merging*) techniques (e.g. Borda, CombSum or CombMNZ):

- DocId: This is an ordering strategy that does not try to early detect relevant documents. The pooled documents are simply sorted by document identifier. This evaluation sequence is the standard sequence followed by TREC assessors when the budget allows to judge the whole pool [47].
- Rank: This adjudication method selects documents from all the rankings in a top-down fashion: all top 1 documents (removing duplicates) go first; next, the documents ranked at the 2nd position are selected (removing documents already judged), and so forth.
- Moffat's method: Moffat and his co-authors [32] proposed a static ordering method based on rank-biased precision (RBP). RBP [33] is an effectiveness metric that estimates the utility of a ranked document from the document's position. RBP assumes that, when users inspect a ranked list of documents, the top document is always examined and users go from each document to the next with probability p, or terminate their search with probability $1 - p$.

[5]`http://tec.citius.usc.es/ir/code/pooling_if.html`

Under these assumptions, the utility of a ranking can be estimated as:

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

where l is the length of the ranking and u_i is the relevance of document at position i. The assumptions of this model imply that, on average, $1/(1 - p)$ are examined during each search[6]. A reasonable setting is $p = 0.8$, which yields to 5 documents examined on average (a reasonable model of user behavior). With binary relevance ($u_i \in \{0, 1\}$), the top-ranked document contributes 0.2 to the overall RBP, the top 2 document contributes 0.16, and so forth. In [32], these weights were employed for prioritizing documents in the pool. Each document (d) gets a weight ($c_{s,d}$) from the document's position ($r_{d,s}$) in each ranking (s), and all the document's weights are added to obtain a global score for the document:

$$\begin{aligned} w_d &= \sum_s c_{s,d} && (7) \\ c_{s,d} &= (1 - p) \cdot p^{r_{d,s}-1} && (8) \end{aligned}$$

Observe that $c_{s,d}$ is the potential contribution of d to the RBP score of the system s (if d is relevant it would contribute $c_{s,d}$ to the RBP of s)[7].

The weight w_d is computed for all pooled documents and documents are judged in decreasing order of w_d. This adjudication approach favors documents that are highly ranked by many retrieval systems.

- Borda: This is a fusion approach based on Borda Count [5]. Borda Count is a well-known voting method that has been regularly applied in metasearch and data fusion. Borda can be naturally employed to rank the documents in the pool. Each retrieval system acts as a voter and each pooled document plays the role of a candidate. Each system gives n points to the document ranked at the top position, the top 2 document gets $n - 1$ points, and so forth (n is the size of the pool). The pooled documents that were not retrieved by the system get an uniform portion of the remaining points. Each pooled document is assigned an overall score, which is the sum of points obtained

[6]See [33], page 2:14, for a proof.

[7]If d is not retrieved then $r_{d,s}$ is set to ∞ and, therefore, $c_{s,d}$ goes to 0.

from each retrieval systems. Finally, the documents are ranked by decreasing score.

  Borda is a simple and efficient fusion approach that requires no retrieval scores and no training data [5]. It has been shown that it is effective at combining the rankings returned by multiple retrieval systems [5].

- CombSum: This is a merging –or rank aggregation– method designed by Shaw and Fox [42]. CombSum sums the document's retrieval scores from all search systems that retrieved the document. This approach rewards documents that have high scores and appear in multiple ranked lists. Given the pooled documents, we computed the CombSum score for each document and, next, we ranked the documents by decreasing CombSum.

- CombMNZ: This is another merging –or rank aggregation– method designed by Shaw and Fox [42]. It is an evolution over CombSum, which consists of multiplying the CombSum score by the number of systems that retrieved the document. Given the pooled documents, we computed the CombMNZ score for each document and, next, we ranked the documents by decreasing CombMNZ. Both CombSum and CombMNZ are among the most effective merging methods [43].

Throughout the years, there has been a steady stream of research on rank aggregation methods [20, 43, 26, 12]. Some effective methods require training data or other sources of evidence. For instance, Weighted Borda Fuse needs training queries to weight the documents. This requirement makes it unsuitable for document adjudication in IR evaluation. We focus here on aggregation methods that require no prior knowledge. Building an IR benchmark starts with no knowledge of the quality of documents, queries or retrieval systems. Consequently, learning methods, such as LambdaMerge [43], which learns to merge multiple ranked list by injecting training data, or ULARA [26], which requires a training stage to produce the systems' weights, are not directly applicable to fusing pooled systems.

TREC5

	Number of judgments						
	100	300	500	700	900	1100	all
DocId	12.64$^{\downarrow}$	32.28$^{\downarrow}$	44.14$^{\downarrow}$	53.56$^{\downarrow}$	59.42$^{\downarrow}$	63.6$^{\downarrow}$	**71.78**
Rank	20.68$^{\downarrow}$	40.22$^{\downarrow}$	51.5	58.98	**63.1**	**65.54**	**71.78**
Moffat	21.2	40.36	51.52	59	63.02	65.46$^{\downarrow}$	**71.78**
Borda	22.82	**44.64**	52.58	**59.36**	62.66	65.48	**71.78**
CombSum	22.64	42.18$^{\downarrow}$	52.30$^{\downarrow}$	57.46	61.48	64.26	**71.78**
CombMNZ	**23.06**	42.76$^{\downarrow}$	**53.22**	58.28	62.12	64.82	**71.78**

TREC6

	Number of judgments							
	100	300	500	700	900	1100	2000	all
DocId	3.2$^{\downarrow}$	12.8$^{\downarrow}$	24.14$^{\downarrow}$	36.16$^{\downarrow}$	46.6$^{\downarrow}$	57.68$^{\downarrow}$	74.28	**74.44**
Rank	20.54$^{\downarrow}$	38.42$^{\downarrow}$	49.66$^{\downarrow}$	57.02$^{\downarrow}$	63.62$^{\downarrow}$	67.8$^{\downarrow}$	74.36	**74.44**
Moffat	21.18$^{\downarrow}$	38.74	49.86$^{\downarrow}$	57.16$^{\downarrow}$	63.6$^{\downarrow}$	67.92$^{\downarrow}$	74.36	**74.44**
Borda	23.2	**42.26**	**54.62**	**62.22**	**67.94**	**70.96**	**74.44**	**74.44**
CombSum	23.30	41.38$^{\downarrow}$	53.14$^{\downarrow}$	61.38$^{\downarrow}$	67.20$^{\downarrow}$	70.12$^{\downarrow}$	**74.44**	**74.44**
CombMNZ	**23.34**	41.40$^{\downarrow}$	53.72$^{\downarrow}$	61.82	67.62	70.54	**74.44**	**74.44**

TREC7

	Number of judgments						
	100	300	500	700	900	1100	all
DocId	5.34$^{\downarrow}$	17.1$^{\downarrow}$	29.52$^{\downarrow}$	41.94$^{\downarrow}$	53.76$^{\downarrow}$	62.16$^{\downarrow}$	**70.28**
Rank	21.84$^{\downarrow}$	40.68$^{\downarrow}$	51.48$^{\downarrow}$	58.5$^{\downarrow}$	63.36$^{\downarrow}$	66.64	**70.28**
Moffat	23.5$^{\downarrow}$	41.1$^{\downarrow}$	51.5$^{\downarrow}$	58.58$^{\downarrow}$	63.42$^{\downarrow}$	66.74	**70.28**
Borda	**26.4**	**47.7**	**58.5**	**64.68**	**68**	**69.26**	**70.28**
CombSum	26.20	46.34$^{\downarrow}$	56.88$^{\downarrow}$	63.30$^{\downarrow}$	67.16$^{\downarrow}$	68.90	**70.28**
CombMNZ	25.78$^{\downarrow}$	46.22$^{\downarrow}$	57.72$^{\downarrow}$	64.02$^{\downarrow}$	67.72	69.18	**74.44**

TREC8

	Number of judgments							
	100	300	500	700	900	1100	2000	all
DocId	6.96$^{\downarrow}$	17.88$^{\downarrow}$	28.86$^{\downarrow}$	39.8$^{\downarrow}$	50.46$^{\downarrow}$	59.62$^{\downarrow}$	73.92	**75.68**
Rank	21.1$^{\downarrow}$	40.2$^{\downarrow}$	51.34$^{\downarrow}$	59.98$^{\downarrow}$	65.42$^{\downarrow}$	69.42$^{\downarrow}$	75.04	**75.68**
Moffat	23.68$^{\downarrow}$	40.58$^{\downarrow}$	51.8$^{\downarrow}$	60.18$^{\downarrow}$	65.64$^{\downarrow}$	69.48$^{\downarrow}$	75.08	**75.68**
Borda	26.94	**49.40**	**59.36**	65.24	**68.98**	71.38	**75.48**	**75.68**
CombSum	**27.54**	49.00	59.04	**65.28**	68.64$^{\downarrow}$	71.14	75.16	**75.68**
CombMNZ	27.06$^{\downarrow}$	48.62$^{\downarrow}$	58.98	65.18	68.68$^{\downarrow}$	**71.40**	75.36	**75.68**

Table 2: Static Pooling Baselines. Average number of relevant documents found at different number of judgments performed. For each judgment level and collection, the highest average is bolded. The symbol $^{\downarrow}$ indicates a significant decrease over the corresponding best average (two-tailed paired t-test, $\alpha = .05$).

We implemented these six static methods and experimented with them against the four test collections described above. The results are shown in Table 2. DocId is the worst performing approach. This result is not surprising, as DocId does not try to order the pooled documents by estimated relevance. Rank and Moffat's method perform roughly the same, and both of them are inferior to Borda, CombSum and CombMNZ. These three methods, Borda, CombSum and CombMNZ, are the best performing approaches. However, the statistical tests suggest that we should prefer Borda. Borda often yields the highest average counts of relevant documents. Furthermore, when any other method surpasses Borda the difference is not statistically significant (e.g. TREC5 at 100 judgements, CombMNZ gives 23.06 relevant documens on average while Borda gives 22.82, but the difference is statistically insignificant). In contrast, CombSum/CombMNZ are often inferior to Borda and the decrease is often statistically significant (e.g. TREC5 at 300 judgments). We therefore chose Borda as our reference static baseline.

TREC5

	Number of judgments						
	100	300	500	700	900	1100	all
Borda	22.82	**44.64**	52.58	**59.36**	62.66	**65.48**	**71.78**
SD (no pseudo-qrels)	22.8$^{\downarrow}$	42.78$^{\downarrow}$	52.14	57.74$^{\downarrow}$	60.98$^{\downarrow}$	63.78$^{\downarrow}$	**71.78**
SD + pseudo-qrels	**24.14**	44.04	**53.16**	58.96	**62.76**	65.08	**71.78**

TREC6

	Number of judgments							
	100	300	500	700	900	1100	2000	all
Borda	23.2$^{\downarrow}$	42.26$^{\downarrow}$	54.62	**62.22**	**67.94**	**70.96**	**74.44**	**74.44**
SD (no pseudo-qrels)	23.58$^{\downarrow}$	41.58$^{\downarrow}$	53.9	61.32$^{\downarrow}$	66.58$^{\downarrow}$	69.72	**74.44**	**74.44**
SD + pseudo-qrels	**24.76**	**44.34**	**55.1**	62.02	67.18	70.38	74.42	**74.44**

TREC7

	Number of judgments						
	100	300	500	700	900	1100	all
Borda	26.4	47.7	58.5	64.68	**68**	69.26	**70.28**
SD (no pseudo-qrels)	26.34$^{\downarrow}$	46.94	57.88$^{\downarrow}$	64.08	67.36	69.1	**70.28**
SD + pseudo-qrels	**27.72**	**48.7**	**59.44**	**65.1**	67.9	**69.36**	**70.28**

TREC8

	Number of judgments							
	100	300	500	700	900	1100	2000	all
Borda	26.94$^{\downarrow}$	49.4$^{\downarrow}$	59.36	65.24	68.98	**71.38**	**75.48**	**75.68**
SD (no pseudo-qrels)	27.18$^{\downarrow}$	48.32$^{\downarrow}$	58.26$^{\downarrow}$	64.12$^{\downarrow}$	67.78	70.4	74.76	**75.68**
SD + pseudo-qrels	**28.38**	**50.64**	**60.3**	**66.2**	**69.32**	**71.38**	75.26	**75.68**

Table 3: Static Score Distribution Methods. Average number of relevant documents found at different number of judgments performed. For each judgment level and collection, the highest average is bolded. The symbol $^{\downarrow}$ indicates a significant decrease over the corresponding best average (two-tailed paired t-test, $\alpha = .05$).

The comparison of Borda against the static methods based on Score Distributions is shown in Table 3. The score distribution model with pseudo-qrels is often the best performing method and, sometimes (e.g. TREC6/8 at 100/300 judgments), the improvement over the two other methods is statistically significant. Although Borda is occasionally the best performing method, the improvement over SD + pseudo-qrels achieved by Borda is never statistically significant. This empirical evidence supports the claim that we should prefer the model based on score distribution with pseudo-qrels. If we can afford 500 or more judgments then any of the methods can be chosen. But if we can only afford a few hundred judgments then we should go for the score distribution model with pseudo-qrels. Another interesting finding from our experiments derives from the behavior of the SD model with and without pseudo-qrels. The evaluation shows that incorporating pseudo-qrels into the initialization of the SD distributions leads to improved models. Our innovative proposal of injecting pseudo-relevance estimations into the SD models gives substantial profits.

4.2. Dynamic methods

We considered the following dynamic methods that have been proposed in existing studies:

- MoveToFront (MTF): MTF is a dynamic adjudication approach proposed by Cormack and his colleagues [15]. It maintains a set of priorities associated with the retrieval systems. Initially, all priorities are set to the same value. A retrieval system is randomly chosen from the top priority systems and its top-ranked document is judged for relevance. If the document assessed is relevant then documents from the same ranking continue to be assessed – in a top-down fashion– until a non-relevant document is extracted. When a non-relevant document is found, the priority of the current system is reduced and the judgment process jumps to another maximum priority system.
- Bayesian Bandits (BB): BB is a recent pooling adjudication method based on multi-armed bandits [30]. The multi-armed bandit framework [38] models the following reinforcement learning problem. A gambler is repeatedly faced with a choice among n bandits (or slot machines). Each bandit has an unknown probability of distributing a prize. After each play, the gambler gets a reward that depends on the selected bandit. His primary goal is to maximize the expected total reward over some period. A number of bandit allocation algorithms have been proposed in the past. These algorithms approach the classical exploration versus exploitation dilemma in different ways. This fundamental dilemma is between choosing the bandit that currently looks optimal (exploit our current knowledge) and choosing another suboptimal bandit and learn more about it (explore). We constrain our discussion to the formal Bayesian method (BB) that was adapted for pooling in [30]. The experiments reported by Losada and colleagues demonstrated that BB is superior to many other adjudication strategies [30]. Each retrieval system is assigned a probability of *winning*. In our case, this value can be thought as the probability of supplying a relevant document. Under a Bayesian framework, this probability is regarded as a parameter endowed with a prior distribution. Initially, all systems are assigned a uniform prior, $\mathcal{U}(0,1)$, because we have no knowledge on the quality of the rankings. Observe that $\mathcal{U}(0,1)$ is equivalent to $Beta(1,1)$ (the Uniform distribution is a particular case of the Beta distribution, $Beta(\alpha,\beta)$, when both α and β are set

to 1). Every time that a document is judged for relevance we get evidence that can be effectively used to update our beliefs about the probability of winning of the systems that retrieved this document. We work with binary relevance and, therefore, the pieces of evidence are events $O \in \{0, 1\}$. These outcomes are modeled as Bernoulli random variables or, equivalently, Binomials with a single trial. This is convenient because the Beta distribution is the conjugate prior distribution for the Binomial distribution, which leads to a closed-form expression for the posterior distribution, $Beta(\alpha + O, \beta + 1 - O)$. As the judgment process runs, we keep updating the Beta distributions associated with the retrieval systems. BB is a non-stationary variant of this Bayesian update method that selects the next system by choosing the system that has the highest mean of the posterior distributions. A full description of BB can be found in [30].

- Hedge: Hedge is an effective online learning method that was adapted for metasearch and pooling in [6]. Hedge starts selecting a document that is highly ranked by many systems. If the document is judged as relevant then we gain confidence on the retrieval systems that ranked the document at high positions and we lose confidence on retrieval systems that ranked the document at low positions. Conversely, if the document is non-relevant then we lose confidence on retrieval systems that ranked the document at high positions and we gain confidence on retrieval systems that ranked the document at low positions. In subsequent rounds, Hedge will likely select documents that are ranked at high positions by systems on which we have confidence. The algorithm maintains a set of weights that represent its confidence on the performance of each system. The algorithm begins with uniform weights and selects documents based on the systems' weights and the ranks of the documents. After each judgment, the relevance outcome and the position of the assessed document in each ranked list determine the confidence loss assigned to each system. A learning rate parameter determines how fast or slow we react to new judgments[8]. A full description of the Hedge algorithm can be found in [6].

[8]Following [6], the learning rate was fixed to 0.1 in all our experiments.

TREC5

	Number of judgments						
	100	300	500	700	900	1100	all
MoveToFront	25.96$^{\downarrow}$	46.14$^{\downarrow}$	56.64$^{\downarrow}$	62.4$^{\downarrow}$	65.66	67.94	**71.78**
BB	26.14$^{\downarrow}$	46.98$^{\downarrow}$	57$^{\downarrow}$	62.32$^{\downarrow}$	65.66	68.28	**71.78**
Hedge	**31.42**	**54.16**	**62.96**	**66.32**	**68.04**	**69.04**	**71.78**

TREC6

	Number of judgments							
	100	300	500	700	900	1100	2000	all
MoveToFront	26.42$^{\downarrow}$	46.4$^{\downarrow}$	56.88$^{\downarrow}$	63.56$^{\downarrow}$	68.38	70.92	74.3	**74.44**
BB	26.06$^{\downarrow}$	46.42$^{\downarrow}$	56.82$^{\downarrow}$	64$^{\downarrow}$	68.76	71.42	**74.42**	**74.44**
Hedge	**31.54**	**53.82**	**63.04**	**67.74**	**70.32**	**71.86**	74.34	**74.44**

TREC7

	Number of judgments						
	100	300	500	700	900	1100	all
MoveToFront	29.04$^{\downarrow}$	47.18$^{\downarrow}$	57.22$^{\downarrow}$	63.62	67.02	69.02	**70.28**
BB	28.68$^{\downarrow}$	49.64$^{\downarrow}$	59.88	65.32	68	69.24	**70.28**
Hedge	**33.42**	**54.92**	**62.1**	**66.12**	**68.34**	**69.46**	**70.28**

TREC8

	Number of judgments							
	100	300	500	700	900	1100	2000	all
MoveToFront	29.7$^{\downarrow}$	49.02$^{\downarrow}$	59.54$^{\downarrow}$	66.68$^{\downarrow}$	70.82	72.96	75.4	**75.68**
BB	29.42$^{\downarrow}$	52.44$^{\downarrow}$	63.16	68.6	71.72	73.52	75.46	**75.68**
Hedge	**35.12**	**58.3**	**65.66**	**69.3**	**71.98**	**73.52**	**75.54**	**75.68**

Table 4: Dynamic Pooling Baselines. Average number of relevant documents found at different number of judgments performed. For each judgment level and collection, the highest average is bolded. The symbol $^{\downarrow}$ indicates a significant decrease over the corresponding best average (two-tailed paired t-test, $\alpha = .05$).

Table 4 reports the performance of the three dynamic baselines. Hedge is the best performing method and MoveToFront is the worst performing method. BB is weaker than Hedge but, after 500/700 judgments, the difference between BB and Hedge is often statistically not significant. We therefore selected both, Hedge and BB, as our reference dynamic baselines for further comparisons.

TREC5

	Number of judgments						
	100	300	500	700	900	1100	all
BB	26.14$^{\downarrow}$	46.98$^{\downarrow}$	57$^{\downarrow}$	62.32$^{\downarrow}$	65.66	68.28	**71.78**
Hedge	**31.42**	**54.16**	**62.96**	**66.32**	**68.04**	69.04	**71.78**
SD + pseudo-qrels	25.06$^{\downarrow}$	48.22$^{\downarrow}$	58.04$^{\downarrow}$	63.2	66.5	**69.1**	**71.78**

TREC6

	Number of judgments							
	100	300	500	700	900	1100	2000	all
BB	26.06$^{\downarrow}$	46.42$^{\downarrow}$	56.82$^{\downarrow}$	64$^{\downarrow}$	68.76$^{\downarrow}$	71.42$^{\downarrow}$	74.42	**74.44**
Hedge	**31.54**	**53.82**	**63.04**	**67.74**	70.32	71.86$^{\downarrow}$	74.34	**74.44**
SD + pseudo-qrels	26.14$^{\downarrow}$	48.5$^{\downarrow}$	59.86$^{\downarrow}$	67.04	**70.64**	**72.96**	**74.44**	**74.44**

TREC7

	Number of judgments						
	100	300	500	700	900	1100	all
BB	28.68$^{\downarrow}$	49.64$^{\downarrow}$	59.88$^{\downarrow}$	65.32$^{\downarrow}$	68$^{\downarrow}$	69.24	**70.28**
Hedge	**33.42**	**54.92**	62.1$^{\downarrow}$	66.12	68.34$^{\downarrow}$	69.46$^{\downarrow}$	**70.28**
SD + pseudo-qrels	28.9$^{\downarrow}$	52.12$^{\downarrow}$	**62.5**	**67.16**	**69.1**	**69.84**	**70.28**

TREC8

	Number of judgments							
	100	300	500	700	900	1100	2000	all
BB	29.42$^{\downarrow}$	52.44$^{\downarrow}$	63.16	68.6	71.72	73.52	75.46	**75.68**
Hedge	**35.12**	**58.3**	**65.66**	**69.3**	71.98	73.52	75.54	**75.68**
SD + pseudo-qrels	29.4$^{\downarrow}$	53.6	63.72	69.1	**72.34**	**74**	**75.6**	**75.68**

Table 5: Dynamic Score Distribution Methods. Average number of relevant documents found at different number of judgments performed. For each judgment level and collection, the highest average is bolded. The symbol $^{\downarrow}$ indicates a significant decrease over the corresponding best average (two-tailed paired t-test, $\alpha = .05$).

Table 5 compares our dynamic method based on SD against BB and Hedge. If we only judge a small set of documents then Hedge is a clear winner. For example, after 100 or 300 judgments, the counts of relevant documents found by Hedge are significantly superior to the counts of relevant documents found by either BB or SD+pseudo-qrels. However, if we go for a deep judgment process – for instance, 1100 judgments– then we should prefer SD+pseudo-qrels. The reason is that SD+pseudo-qrels often leads to statistically significant improvements over Hedge/BB in TREC6 and TREC7 (in TREC7 even with only 500 judgments); and, additionally, in TREC5 and TREC8, SD+pseudo-qrels does not differ from Hedge in a statistically significant way.

At the beginning of the process, the SD+pseudo-qrels distributions work with many estimated judgments and Hedge, which does not make any apriori assumption about the relevance of the documents, is superior to the SD model. As we obtain more human judgments, the SD model overcomes Hedge. This result suggests that the constant update of the models leads to improved fittings of the score distributions.

5. Discussion

The experiments performed allow us to draw some definitive conclusions. With static adjudication, SD+pseudo-qrels is the most consisting approach. At shallow judgment levels, it is superior –in a statistically significant way– to the competing methods. At deep judgment levels, SD+pseudo-qrels and Borda behave roughly the same and, thus, any of them is a safe bet. Score distribution+pseudo-qrels are boosted by pseudo-relevance estimations, which supply an initial push to early identify relevant items. But static adjudication methods do not react to the outcome of the assessments and, therefore, the quality of SD+pseudo-qrels estimations loses steam as we go down in the assessment lists.

With dynamic adjudication, Hedge is highly effective at shallow judgment levels; and, with deep judgments, SD+pseudo-qrels becomes the preferable choice. Hedge is an online learning method that quickly rewards or penalizes the search systems based on performance. These rewards or penalties work very well at the beginning of the process (top ranks), when the population of relevant items is high, and it is crucial to avoid poorly performing systems. SD+pseudo-qrels is somehow smoother. Rather than working with individual rewards/penalties, it updates the distributions of scores for the full ranking and this leads to higher payoffs in the long run.

In practice, the selection among these methods will be governed by factors such as the viability of implementing static or dynamic assessment processes, the budget available for doing judgments, and the choice between judging deep pools of a small number of queries or judging shallow pools of a larger number of queries.

Another important issue is how many judgments we need to create a robust test collection. Our experiments demonstrate that some adjudication strategies are effective at early identifying relevant documents and, therefore, we may want to employ them to create relevance judgments with reduced human effort. But judging only a subset of the pooled documents introduces a *bias* when compared to judging the full pool. A standard approach to measure this bias is to use the official TREC judgments as the basis for comparison. The official TREC judgments contain relevance assessments for all documents in the pool. These full-pool judgments are utilized for ranking the retrieval systems that participated in the search challenge. This ranking is produced by ordering the participating systems using an official effectiveness measure (for instance, Mean Average Precision). With any given subset of assessments (e.g. with the first 100 assessments generated by Hedge), we can proceed in the same way. The reduced set of assessments can be employed to rank the participating systems, and we can compute a measure of agreement between the official ranking and the ranking under the subsetting strategy. This bias analysis is a common way to analyze pooling prioritization methods and helps to estimate the judgment effort needed to have a sufficiently high agreement with the official ranking.

Kendall's τ correlation [25] is a well-known measure of agreement between a pair of rankings. Given two rankings of the same items, Kendall's τ coefficient is defined as:

$$\tau(ranking1, ranking2) = \frac{P - Q}{P + Q} \tag{9}$$

where P is the number of pairs of items that are concordant (have the same order in both rankings) and Q i the number of pairs of items that are discordant (have the reverse order). τ takes values in the range $[-1, 1]$ (1 means that the rankings are identical while -1 means that one ranking is the reverse of the other). If τ equals 0 then half of the pairs of items are concordant and half of the pairs of items are discordant. In IR, Kendall's τ is a widely adopted metric to compare pairs of rankings.

Elzinga and colleagues [21, 28] proposed a set of axioms for measures of concordance, and they thoroughly studied classical measures, such as Kendall's τ. They showed that Kendall's τ is not very sensitive to small perturbations of the rankings involved and argued that it is hard to generalize Kendall's τ to compute agreement with more than two rankings. This study led to a new class of measures of concordance. A full description of these measures can be found in [21]. Essentially, these new measures represent the rankings as strings of characters (e.g. "abcde" vs "baced") and compute all distinct common subsequences of strings. Following [21], we adopted the following measure of concordance (γ):

$$\gamma(ranking1, ranking2) = \frac{\log_2(\phi(ranking1, ranking2) - (n + 1))}{\log_2(2^n - (n + 1))} \tag{10}$$

where $\phi(ranking1, ranking2)$ is the number of all distinct common subsequences between $ranking1$ and $ranking2$, and n is the size of the rankings. γ takes values in the range $[0, 1]$.

Figures 3 and 4 plot the evolution of Kendall correlation of the static and dynamic methods against the number of assessments done. Figures 5 and 6 plot similar graphs for concordance (γ). Figures 3 and 5 confirm the superiority of SD+pseudo-qrels (static version) over Borda. The concordance plot is less smooth than the correlation plot. This is because γ, when compared with τ, is much more sensitive to perturbations of the orderings [21]. The dynamic plots, Figs. 4 and 6, clearly show that SD+pseudo-qrels (dynamic version) needs a few hundred assessments to get to correlation or concordance levels that are similar or superior to those obtained by the two other dynamic methods. Another interesting finding is that all these methods quickly get to high levels of correlation or concordance with the official ranking. Evaluation methods that produce Kendall correlations higher than .9 are considered robust and reliable [46]. Our analysis confirms that these adjudication methods can reduce the judgment effort and still yield a set of assessments that are effective for comparing retrieval algorithms.

In Tables 6 and 7, we provide additional quantitative evidence on the assessment effort required by these strategies. We report the number of judgments needed by every adjudication method in order to get to levels of correlation or concordance higher than .9, .95 and .99. Under the static setting, the reduction of SD+pseudo-qrels over Borda is quite substantive. In some cases, SD+pseudo-qrels saves up several hundreds of assessments. With dynamic adjudication, SD+pseudo-qrels can save up some assessments, but only if we want to go to deep judgment levels and ensure levels of correlation higher than .99.

			TREC5	
		$\tau \geq .9$	$\tau \geq .95$	$\tau \geq .99$
Static Methods	Borda	213	460	742
	SD + pseudo-qrels	**152**	**341**	**725**
Dynamic Methods	BB	45	168	**492**
	Hedge	**31**	**97**	549
	SD + pseudo-qrels	131	261	534
			TREC6	
		$\tau \geq .9$	$\tau \geq .95$	$\tau \geq .99$
Static Methods	Borda	335	518	**1080**
	SD + pseudo-qrels	**197**	**433**	1204
Dynamic Methods	BB	113	**243**	1091
	Hedge	**102**	269	1006
	SD + pseudo-qrels	169	364	**956**
			TREC7	
		$\tau \geq .9$	$\tau \geq .95$	$\tau \geq .99$
Static Methods	Borda	168	354	822
	SD + pseudo-qrels	**137**	**312**	**668**
Dynamic Methods	BB	**104**	**222**	757
	Hedge	116	302	1022
	SD + pseudo-qrels	127	247	**746**
			TREC8	
		$\tau \geq .9$	$\tau \geq .95$	$\tau \geq .99$
Static Methods	Borda	229	417	938
	SD + pseudo-qrels	**154**	**316**	**834**
Dynamic Methods	BB	**109**	**234**	**682**
	Hedge	135	343	789
	SD + pseudo-qrels	145	253	685

Table 6: Number of judgments required to achieve .9, .95 or .99 Kendall's τ correlation with the ranking of systems done with the full pool. For each group of methods, correlation level and collection, the lowest number of judgments is bolded.

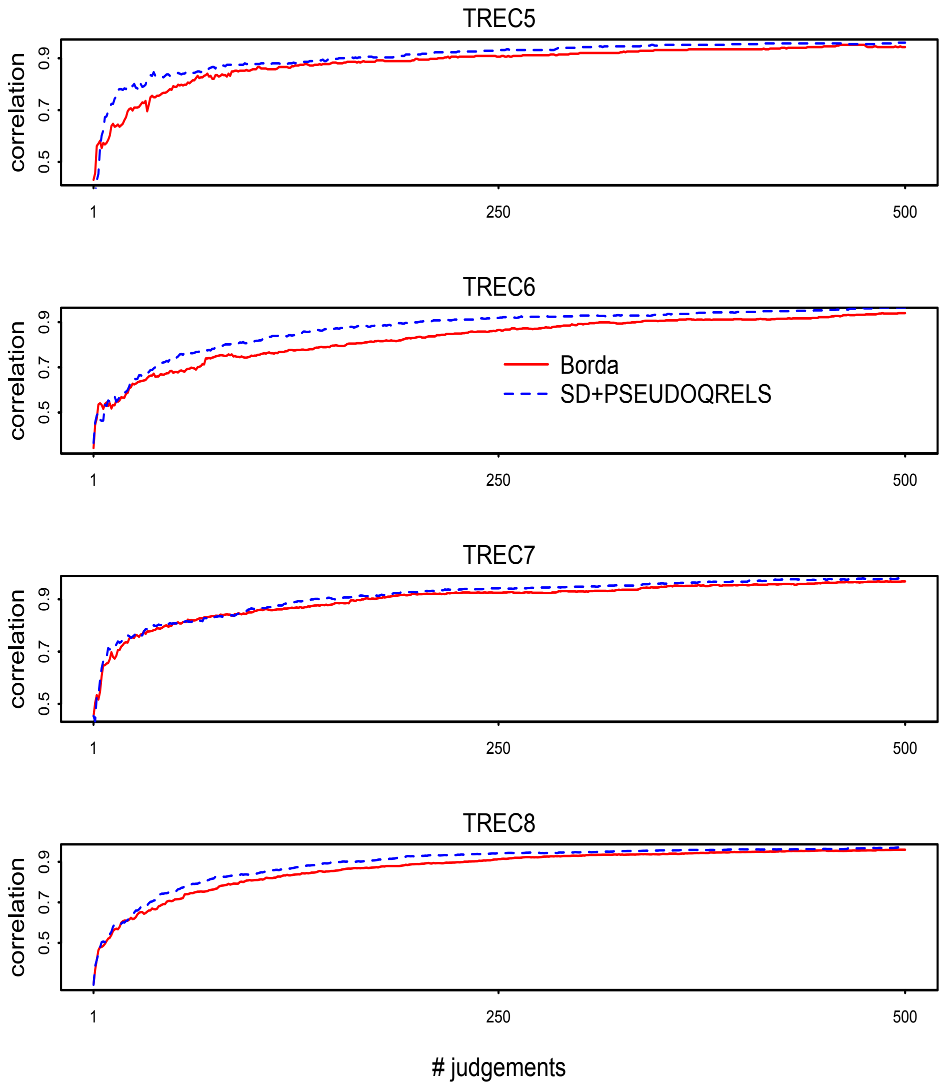

Figure 3: Kendall's τ correlation between the ranking of systems using the full pool and the ranking of systems built by the two static subset pooling strategies. The plots show the evolution of the correlation at different number of documents judged (up to 500 judgments).

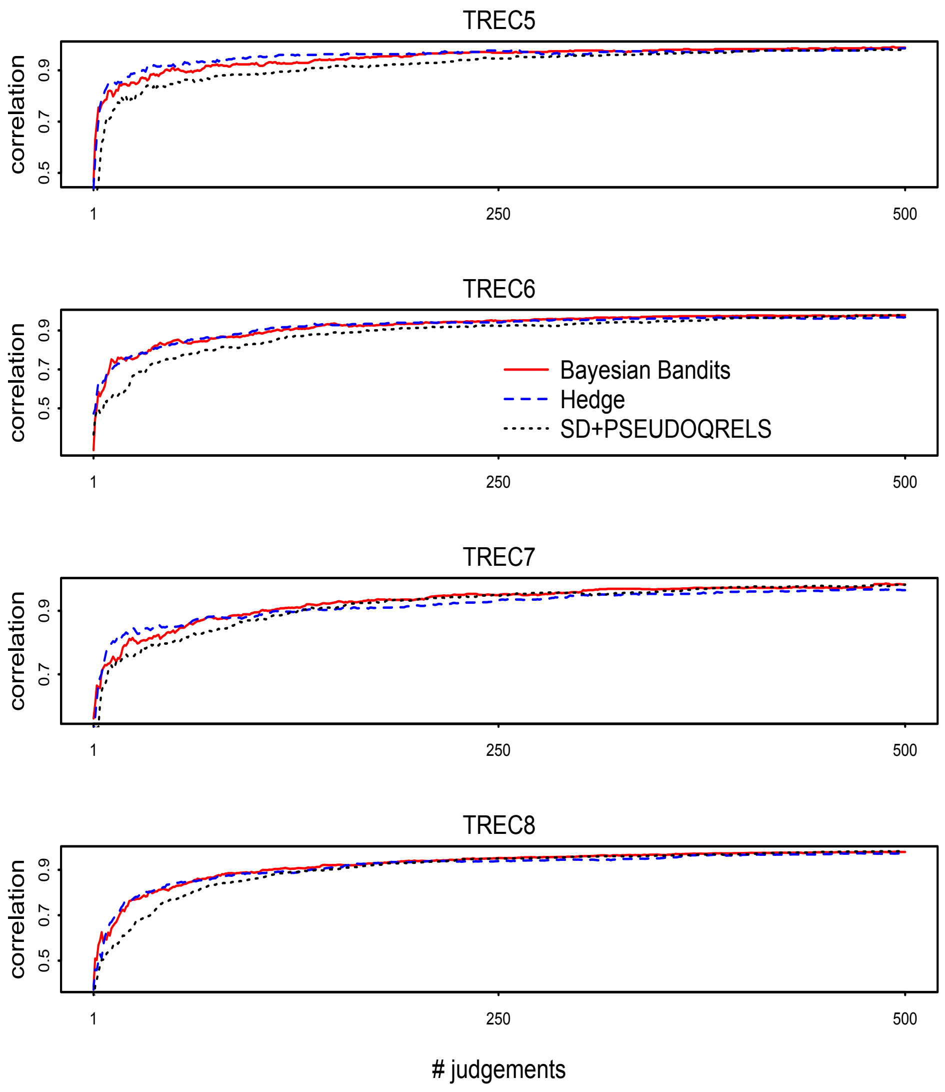

Figure 4: Kendall's τ correlation between the ranking of systems using the full pool and the ranking of systems built by the three dynamic subset pooling strategies. The plots show the evolution of the correlation at different number of documents judged (up to 500 judgments).

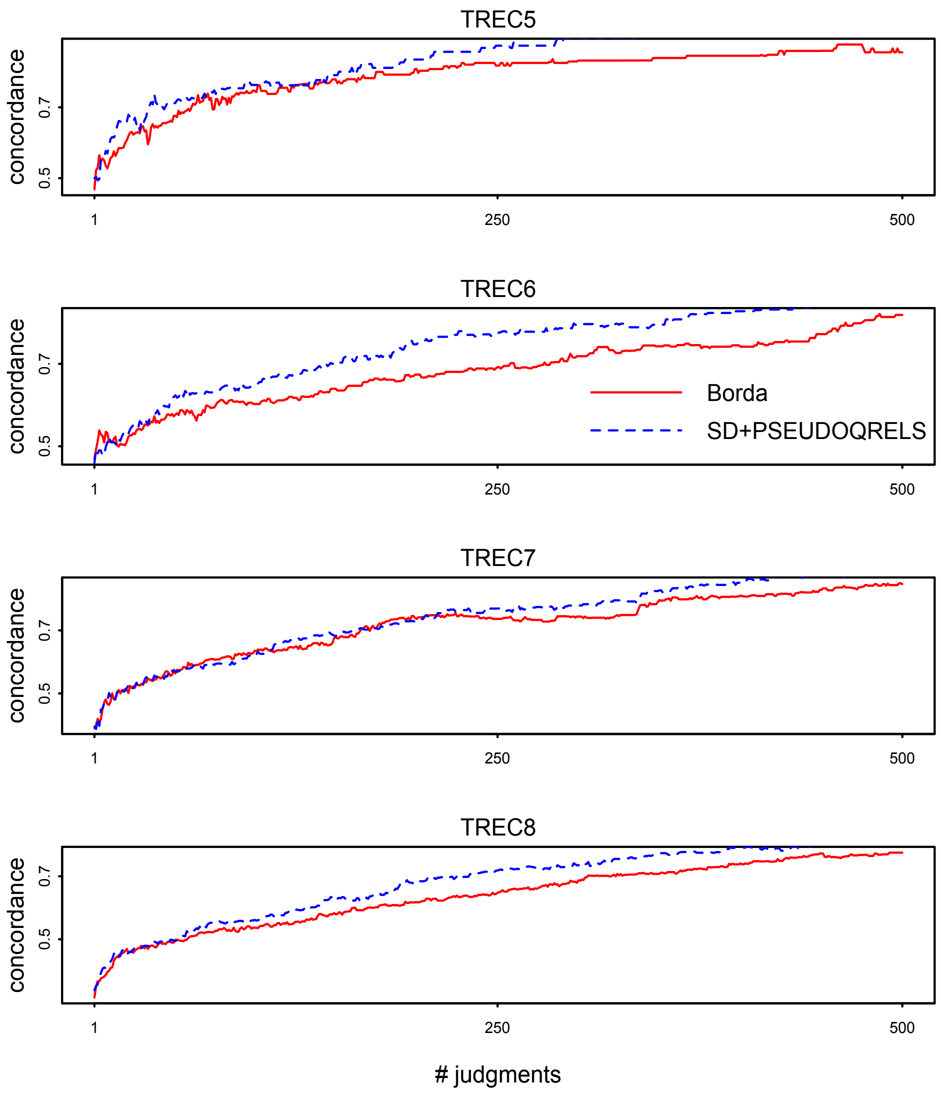

Figure 5: Concordance (γ) between the ranking of systems using the full pool and the ranking of systems built by the two static subset pooling strategies. The plots show the evolution of the concordance at different number of documents judged (up to 500 judgments).

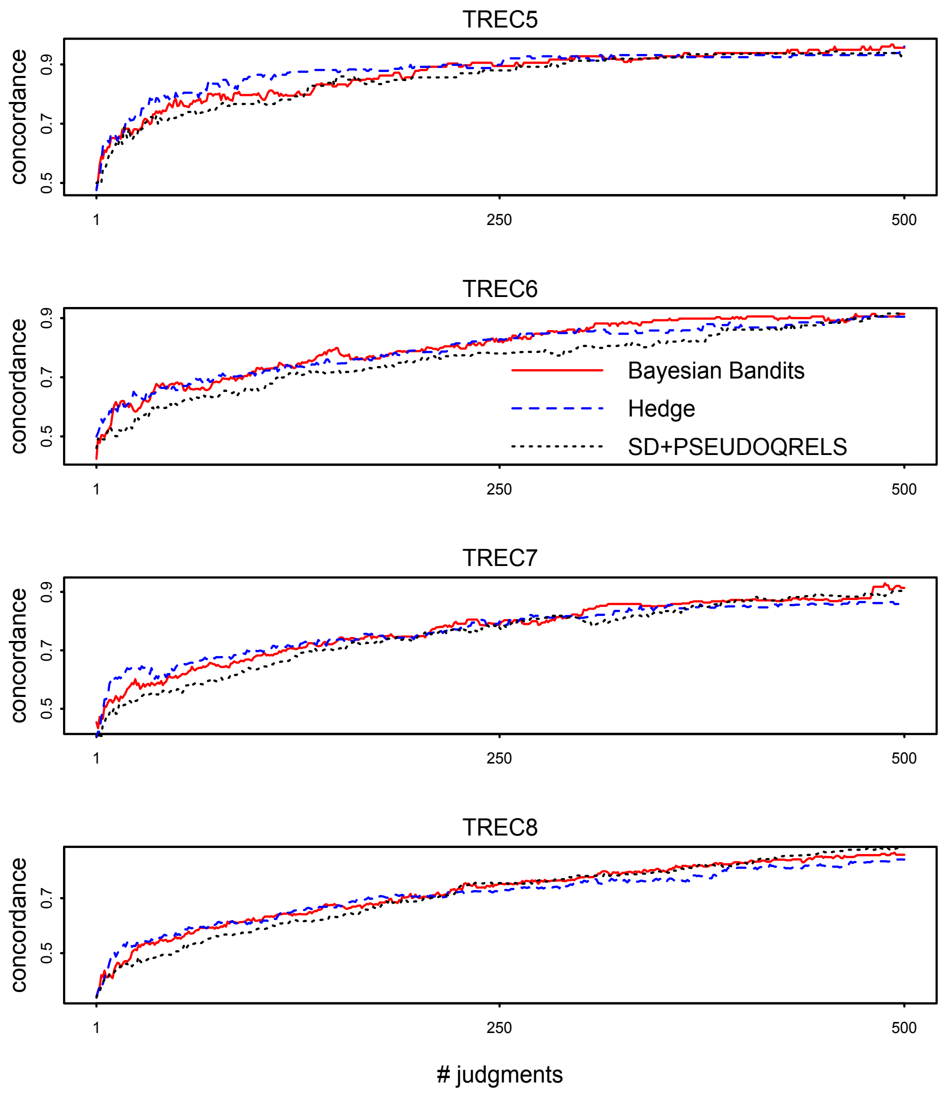

Figure 6: Concordance (γ) between the ranking of systems using the full pool and the ranking of systems built by the three dynamic subset pooling strategies. The plots show the evolution of the concordance at different number of documents judged (up to 500 judgments).

			TREC5	
		$\gamma \geq .9$	$\gamma \geq .95$	$\gamma \geq .99$
Static Methods	Borda	561	685	**1843**
	SD + pseudo-qrels	**340**	**674**	1866
Dynamic Methods	BB	**218**	**494**	1756
	Hedge	220	498	1937
	SD + pseudo-qrels	261	531	**1717**
			TREC6	
		$\gamma \geq .9$	$\gamma \geq .95$	$\gamma \geq .99$
Static Methods	Borda	755	**1080**	**1314**
	SD + pseudo-qrels	**672**	1119	1414
Dynamic Methods	BB	**384**	924	1358
	Hedge	470	**652**	1476
	SD + pseudo-qrels	481	911	**1017**
			TREC7	
		$\gamma \geq .9$	$\gamma \geq .95$	$\gamma \geq .99$
Static Methods	Borda	629	881	**1276**
	SD + pseudo-qrels	**505**	**695**	1310
Dynamic Methods	BB	**481**	951	1362
	Hedge	714	1035	1461
	SD + pseudo-qrels	493	**916**	**1355**
			TREC8	
		$\gamma \geq .9$	$\gamma \geq .95$	$\gamma \geq .99$
Static Methods	Borda	**777**	1016	2230
	SD + pseudo-qrels	788	**936**	**2228**
Dynamic Methods	BB	588	**795**	2318
	Hedge	687	966	**1286**
	SD + pseudo-qrels	**584**	820	1337

Table 7: Number of judgments required to achieve .9, .95 or .99 concordance (γ) with the ranking of systems done with the full pool. For each group of methods, concordance level and collection, the lowest number of judgments is bolded.

6. Related Work

6.1. Score Distributions

Score Distribution models have been employed for supporting different Information Access tasks, including meta-search [31], threshold optimization in different search problems [1, 2], query performance prediction [17], image retrieval [4] and pseudo-relevance feedback [36].

The work by Manmatha and colleagues [31], which combined the outputs of multiple search engines, is perhaps the most related previous paper. We have adopted their method, based on posterior probabilities (eq. 4), to estimate the importance assigned to each retrieved document by each retrieval method. However, our work extends Manmatha's study in a number of ways. First, we employ the SD models in a different task, aimed at prioritizing documents for creating an IR benchmark. Second, Manmatha's method is inherently *static*: the SD models are built once and then used for building the output of a meta-search service. Our dynamic models, instead, are adaptive: the SD models are constantly updated as judgments come in. The dynamic nature of the evaluation task is an environment that fits well with the formal SD models. As a matter of fact, the flow of human judgments permits to build SD models from estimations that are increasingly more reliable. Third, we have built static and dynamic models from pseudo-relevance judgments. This innovative proposal makes an intelligent initialization of the SD models, and we have shown that this initialization is superior to the standard estimation of SDs (using the scores alone).

Liang and colleagues [27] adopted a data fusion perspective for search result diversification. Given multiple rankings, they aimed at optimising for diversity. To meet this aim, they worked with retrieval scores and followed a topic modeling approach to extract latent topics. Liang and colleagues modeled each topic as a Log-Normal distribution and used the top-ranked documents to estimate the parameters of the Log-Normal distributions. This estimation is a form of injecting pseudo-relevance evidence into the estimation of Score Distribution models. However, our paper differs from Liang's paper in a number of ways. First, our method to obtain pseudo-relevance data is different to Liang et al's method. We do not take the top documents and assume that they are relevant. Instead, we studied and compared existing methods –by Soboroff and colleagues [44], by Sakai & Lin [40], and by Nuray & Can [34]– to extract pseudo-qrels from multiple rankings. Our analysis concluded that Soboroff's strategy was the most consistent approach to extract pseudo-relevance data. Second, we model the distribution of scores in rankings of documents ordered by estimated relevance, while Liang's paper is concerned with how the distribution of document scores relates to different topics. These differing needs result in models that are different: our Score Distribution Models are mixtures of two Log-Normal distributions (one for relevant documents and another one for non-relevant documents), while Liang's models have one Log-Normal distribution for each topic. Third, the task we address is intrinsically different to search result diversification. We aim at fusing multiple rankings for reducing the effort of creating a set of relevance assessments. Fourth, as the assessment process runs, our dynamic models iteratively update (and improve) the pseudo-relevance data and the Score Distribution models. This adaptative strategy is not a feature of Liang's models.

Wilkins and colleagues [50] also employed Score Distributions for fusion in multimedia retrieval. For a given query, they produced multiple rankings using different features (e.g. textual features and visual features). Their fusion approach, which is also quite different to ours, combined these rankings using Dempster-Shaper theory.

6.2. Pooling

The general idea of pooling, as a way to aggregate evidence, transcends search technologies. For instance, pooling methods have been employed in Image Fusion evaluation [37]. And prioritizing pooled documents can also be seen as an instance of a more general problem: ordering-based decision making. Ordering-based decision making consists of finding the most suitable method for evaluating candidates based on multiple criteria. A survey on ordering-based decision making can be found in [11].

Pooling has been a core component in IR evaluation for decades. Pooling-based collections have been created in many evaluation campaigns, like TREC [48] or CLEF [22]. In IR, given a query and multiple ranked lists of documents, judging deeply the pools is usually prohibitive. Many studies have investigated on *subset pooling models*. The aim is to efficiently scan pools, trying to identify a sufficient number of relevant documents as quickly as possible. These strategies can substantially reduce the assessment effort and lead to high quality and reusable test collections [32, 7, 9, 14]. There are two main classes of subset pooling methods: static and dynamic. Static methods build a fused ranking of pooled documents and this ranked list, which remains unchanged during the assessment process, is used to prioritize judgments. In our study, we considered several existing baseline methods that are static, namely Borda [5], the method by Moffat and colleagues [32], a rank-based prioritization, CombSum [42], CombMNZ [42] and a näive baseline based on document identifiers. Our results show that our static method based on SDs is superior to all of them. The second class of subset pooling methods are dynamic. In this case, the priority of pooled documents changes as we obtain relevance assessments. Dynamic methods can quickly adapt to the outcome of the assessments done so far and, thus, they improve our chances of early finding relevant documents. However, dynamic methods put an additional burden on the creation of the qrels [29]. The assessments need to be coordinated (the assessment of the next document cannot start until the previous assessment has finished) and, therefore, standard speed-up strategies, based on parallel distribution across multiple assessors, cannot be implemented. We have considered several dynamic baselines, including MoveToFront [15], Bayesian Bandits [30] and Hedge [6], and compared them against our SD models. Our results show that, if we want to judge a small set of documents, Hedge should be the method of choice. Still, the dynamic SD-based solution is preferable when we need to create qrels with several hundred assessed documents.

In this work, we did not study the consistency of assessor judgments and inter-assessor agreement. However, with multiple assessors inspecting the same query-document pair, our approach would still be applicable. In such case, every assessment would be obtained by aggregating the assessments of several human judges.

This aggregation can be done by employing current standards in IR evaluation (e.g., majority voting) [41], or by studying formal ways to combine expert's opinions [23].

In this paper, we have explored rank fusion in the context of pooling-based evaluation of IR systems. Many other researchers have applied different forms of fusion to several search tasks. For instance, Chenlo and colleagues [13] combined different types of evidence for blog distillation search, and a complete survey of Information Fusion for web search can be found in [51].

7. Conclusions and Future Work

Our study has shown that SD-based models lead to effective solutions that fuse multiple ranked list for prioritizing relevance assessments in IR evaluation. Under a static prioritization setting, SD models are superior to all the methods included in our comparison. Under a dynamic prioritization setting, the SD models are inferior to the Hedge algorithm, but only if we do shallow assessments. If we need hundreds of judgments (e.g. for a high recall information seeking task) then the SD models can build the set of relevance assessments with reduced effort (when compared with Hedge).

Furthermore, we have proposed an innovative way to build SD models from pseudo-relevance data. We provided empirical evidence on the positive impact of pseudo-relevance evidence on estimating SD models. To the best of our knowledge, this is the first study that injects pseudo-relevance data into SDs and our promising results in IR evaluation could apply more generally. We have only started exploring pseudo-relevance and SDs, and only begin to understand how they can be combined and put to fruitful use. We plan to keep exploring this combination and we will study other tasks that can benefit from these improved SD models.

Another line of future work is to apply pooling methods in Machine Learning and Data Mining. More specifically, we will study pool-based ways to prioritize unlabeled items. These prioritization strategies can be helpful in building robust and effective ground-truth data.

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) "Consellería de Cultura, Educación e Ordenación Universitaria" (accreditation 2016-2019, ED431G/08) and the European Regional Development Fund (ERDF).

References

[1] A. Arampatzis, J. Beney, C. H. A. Koster, and T. P. van der Weide. Incrementality, Half-life, and Threshold Optimization for Adaptive Document Filtering. In *Proceedings of the 9th Text REtrieval Conference, TREC 2000*. National Institute for Science and Technology (NIST), 2000.

[2] A. Arampatzis, J. Kamps, and S. Robertson. Where to stop reading a ranked list?: Threshold optimization using truncated score distributions. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 524–531, New York, NY, USA, 2009. ACM.

[3] A. Arampatzis and S. Robertson. Modeling score distributions in information retrieval. *Information Retrieval*, 14(1):26–46, 2011.

[4] A. Arampatzis, K. Zagoris, and S. A. Chatzichristofis. Dynamic two-stage image retrieval from large multimedia databases. *Information Processing & Management*, 49(1):274–285, January 2013.

[5] 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.

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

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

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

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

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

[11] S. Chen, J. Liu, H. Wang, and J.C. Augusto. Ordering based decision making – a survey. *Information Fusion*, 14(4):521 – 531, 2013.

[12] Yuxin Chen and Changho Suh. Spectral MLE: top-k rank aggregation from pairwise comparisons. *CoRR*, abs/1504.07218, 2015.

[13] J.M. Chenlo, J. Parapar, D. Losada, and J. Santos. Finding a needle in the blogosphere: An information fusion approach for blog distillation search. *Information Fusion*, 23:58 – 68, 2015.

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

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

[16] R. Cummins. Measuring the ability of score distributions to model relevance. In *Proceedings of the 7th Asia Conference on Information Retrieval Technology*, AIRS'11, pages 25–36, Berlin, Heidelberg, 2011. Springer-Verlag.

[17] R. Cummins. Document score distribution models for query performance inference and prediction. *ACM Trans. Inf. Syst.*, 32(1):1–28, January 2014.

[18] R. Cummins and C. O'Riordan. On theoretically valid score distributions in information retrieval. In *Proceedings of the 34th European Conference on Advances in Information Retrieval*, ECIR'12, pages 451–454, Berlin, Heidelberg, 2012. Springer-Verlag.

[19] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[20] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 613–622, New York, NY, USA, 2001. ACM.

[21] C.H. Elzinga, H. Wang, Z. Lin, and Y. Kumar. Concordance and consensus. *Information Sciences*, 181(12):2529–2549, June 2011.

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

[23] E. Herrera-Viedma, F. J. Cabrerizo, J. Kacprzyk, and W. Pedrycz. A review of soft consensus models in a fuzzy environment. *Information Fusion*, 17:4 – 13, 2014. Special Issue: Information fusion in consensus and decision making.

[24] G. Kazai, N. Gövert, M. Lalmas, and N. Fuhr. The inex evaluation initiative. In Henk Blanken, Torsten Grabs, Hans-Jörg Schek, Ralf Schenkel, and Gerhard Weikum, editors, *Intelligent Search on XML Data: Applications, Languages, Models, Implementations, and Benchmarks*, pages 279–293, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[25] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[26] A. Klementiev, D. Roth, and K. Small. *An Unsupervised Learning Algorithm for Rank Aggregation*, pages 616–623. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[27] S. Liang, Z. Ren, and M. de Rijke. Fusion helps diversification. In *Proceedings of the 37th International ACM SIGIR Conference on Research &#38; Development in Information Retrieval*, SIGIR '14, pages 303–312, New York, NY, USA, 2014. ACM.

[28] Z. Lin, H. Wang, and C.H. Elzinga. Concordance and the smallest covering set of preference orderings. *CoRR*, abs/1609.04722, 2016.

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

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

[31] R. Manmatha, T. Rath, and F. Feng. Modeling score distributions for combining the outputs of search engines. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 267–275, New York, NY, USA, 2001. ACM.

[32] 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.

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

[34] R. Nuray and F. Can. Automatic ranking of information retrieval systems using data fusion. *Inf. Process. Manage.*, 42(3):595–614, 2006.

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

[36] J. Parapar, M. A. Presedo-Quindimil, and A. Barreiro. Score distributions for pseudo relevance feedback. *Information Sciences*, 273:171 – 181, 2014.

[37] V. Petrović and V. Dimitrijević. Focused pooling for image fusion evaluation. *Information Fusion*, 22:119 – 126, 2015.

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

[39] S. Robertson. On score distributions and relevance. In G. Amati, C. Carpineto, and G. Romano, editors, *Advances in Information Retrieval: 29th European Conference on IR Research, ECIR 2007, Rome, Italy, April 2-5, 2007. Proceedings*, pages 40–51, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[40] T. Sakai and C. Lin. Ranking retrieval systems without relevance assessments – revisited. In *Proceedings of the 3rd International Workshop on Evaluating Information Access*, EVIA 2010, Tokyo, Japan, 2010.

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

[42] J. A. Shaw and E. A. Fox. Combination of multiple searches. In *The Second Text Retrieval Conference (TREC-2)*, pages 243–252, 1994.

[43] D. Sheldon, M. Shokouhi, M. Szummer, and N. Craswell. Lambdamerge: Merging the results of query reformulations. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 795–804, New York, NY, USA, 2011. ACM.

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

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

[46] E. Voorhees. Evaluation by highly relevant documents. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 74–82, New York, NY, USA, 2001. ACM.

[47] 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.

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

[49] E. Voorhees and D. Harman. *TREC: Experiment and Evaluation in Information Retrieval (Digital Libraries and Electronic Publishing)*. The MIT Press, 2005.

[50] P. Wilkins, P. Ferguson, and A. F. Smeaton. Using score distributions for query-time fusion in multimedia retrieval. In *Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, MIR '06, pages 51–60, New York, NY, USA, 2006. ACM.

[51] J. Yao, V. Raghavan, and Z. Wu. Web information fusion: A review of the state of the art. *Information Fusion*, 9(4):446 – 449, 2008. Special Issue on Web Information Fusion.