Taily: Shard Selection Using the Tail of Score Distributions

Robin Aly, Djoerd Hiemstra
University Twente
The Netherlands
{r.aly,hiemstra}@ewi.utwente.nl

Thomas Demeester
Ghent University - iMinds
Belgium
thomas.demeester@intec.ugent.be

ABSTRACT

Search engines can improve their efficiency by selecting only few promising shards for each query. State-of-the-art shard selection algorithms first query a central index of sampled documents, and their effectiveness is similar to searching all shards. However, the search in the central index also hurts efficiency. Additionally, we show that the effectiveness of these approaches varies substantially with the sampled documents. This paper proposes Taily, a novel shard selection algorithm that models a query's score distribution in each shard as a Gamma distribution and selects shards with highly scored documents in the tail of the distribution. Taily estimates the parameters of score distributions based on the mean and variance of the score function's features in the collections and shards. Because Taily operates on term statistics instead of document samples, it is efficient and has deterministic effectiveness. Experiments on large web collections (Gov2, CluewebA and CluewebB) show that Taily achieves similar effectiveness to sample-based approaches, and improves upon their efficiency by roughly 20% in terms of used resources and response time.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Search process

Keywords

Distributed Retrieval, Database Selection

1. INTRODUCTION

For large collections, search engines have to shard their index to distribute it over multiple machines. Search efficiency can be further increased by shard selection, that is, by querying a small number of promising shards for each query [5]. An important research challenge in this domain is the definition of shard selection algorithms that have to address the following two issues. First, selective shard search should be more efficient than searching all shards, and second, selective shard search should be as effective as searching all shards. In this paper, we investigate the tradeoff between efficiency and effectiveness of sharded search. We propose Taily, a new shard selection algorithm that represents shards by using parametric distributions of the document scores for a query. Taily is much more efficient while showing similar effectiveness, as compared to an exhaustive search.

State-of-the-art shard selection algorithms use a central sample index (CSI) that contains randomly selected documents from each shard [21, 22, 14]. The algorithms use the results of an initial search against the CSI to select the shards to be used in a second, sharded search. In the literature, the efficiency of shard selection algorithms is usually measured by resources used in the sharded search. In this paper, we argue that efficiency measures should also consider the resources spent during the initial search on the CSI, which can be substantial. For example, a common sample size in the literature is four percent [14], which results in a CSI that is bigger than an average shard once there are more than 25 shards. Searching a CSI that is as large as the average shard uses a considerable percentage of the resources required. For the common case that the algorithm selects two shards, the initial search uses roughly one third of the resources required for answering a query. In our experiments we show that our algorithm is more efficient than current sample-based methods, especially when also considering the resources of the initial search, while maintaining similar effectiveness.

Although the query response time is seldom investigated in the shard selection literature, it is often considered more important than the used resources, which are relatively cheap nowadays [9]. Whereas the search in a CSI can use a substantial amount of the total resources, the influence of its execution time on the total query response time can be more severe. In the above example, the initial search would roughly double the response time, assuming the parallel execution of the sharded search. We show that Taily's improvement over the query response time of current shard selection algorithms is particularly strong.

One aspect of sample-based methods that has not been studied so far is the effect of the particular random sample in the CSI on the search effectiveness. One might expect that, if samples are truly random and sufficiently large, different random samples would produce stable effectiveness of the search system in terms of precision or nDCG. We show in this paper that this expectation does not hold in practice. As we show in Section 5, different sample sizes of up to 4%

SIGIR'13, July 28–August 1, 2013, Dublin, Ireland.

of each shard, lead to substantially different effectiveness of the sharded search system. We believe that this variation in effectiveness is a drawback of the sample-based methods tested in this paper.

Like Kulkarni et al. [14] and Arguello et al. [3], Taily is used on clusters of machines in a cooperative search environment. Taily selects one or more shards by estimating the number of documents of a shard that are highly scored in the collection, which we model by the right tail of the score distributions in the collection and the shards. The basic idea of this approach has been proposed by Markov [15]. To estimate these score distributions, we follow the approach by Kanoulas et al. [12], who derive score distributions from statistics of features related to a query's terms (e.g. a term's language model score). Being based on statistics for each term in a vocabulary, Taily belongs to the class of vocabulary-based selection algorithms, which are often biased to larger shards and often show weaker performance than sample-based methods. Our contribution to the shard selection literature are threefold: first, experiments show that its effectiveness is similar or stronger than the effectiveness of sample-based methods, second, compared to a search in a CSI, its processing of feature statistics is more efficient, and results in a lower number of resources used in total and a faster query response time, and finally, compared to sample-based methods, the efficiency and effectiveness do not depend on the size or the sampled documents in a CSI.

The remainder of this paper is structured as follows. Section 2 describes related work on shard selection. Section 3 elaborates on Taily's shard selection algorithm. Section 4 defines the measures that we used to compare our method to state-of-the-art algorithms. Section 5 describes the experiments that we conducted to explore the performance of our search method. Section 6 concludes this paper.

2. RELATED WORK

In this section, we present shard selection algorithms from two classes that are most related to this work. Note that there is also a significant number of works on other approaches to shard selection. Because of space limitations, we refer the interested reader to Kulkarni et al. [14] where a more exhaustive list of related work is presented.

Most of the early shard selection algorithms are vocabulary-based: they represent shards by collection statistics about the terms in the search engine's vocabulary. The popular CORI algorithm by Callan et al. [6] represents a shard by the number of its documents that contain the terms of a vocabulary. The shards are ranked using the INDRI version of the $tf \cdot idf$ score function using the mentioned number as the frequency of each query term. CORI selects a fixed number of shards from the top of this ranking. Xu and Croft [23] build upon this approach by representing them by topical language models. The shards are ranked by the Kullback-Leibner divergence between the language models of the query and the topics. CORI and the algorithm by Xu and Croft characterize a shard by statistics over *all* its documents. We propose that shard selection algorithms should focus on documents with high scores because they contribute most to the system's effectiveness. The gGloss algorithm by Gravano and Garcia-Molina [10] selects shards based on the distribution of the vector space weights, which we refer to as features values. They assume that the distribution of the features is uniform. The algorithm ranks a shard by its expected score value calculated from the expectation of all feature values for a query. In this paper, we also consider the expected score, which we also infer through the expected feature values of the query terms. However, unlike the gGloss algorithm, we assume a gamma distribution of the score instead of a uniform distribution, and focus on the tail of the distribution, which contains the highly scored documents.

Algorithms from the second main class of are called sample-based algorithms because they use a central sample index (CSI) of documents from each shard for shard selection. The REDDE algorithm [21] ranks shards according to the number of the highest ranked documents that belong to this shard, weighted by the ratio between the shard's size and the size of the shard's documents in the CSI. The SUSHI algorithm [22] determine the best fitting function from a set of possible functions that between the estimated ranks of a shard's documents in the central index and their observed scores from the initial search. Using this function, the algorithm estimates the scores of the top-ranked documents of each shard. SUSHI selects shards based on their estimated number of documents among a number of top-ranked documents in the global ranking. REDDE and SUSHI assume that a shard's documents in the CSI are at equidistant ranks in the ranking of the shard. We believe that this assumption may be too strong because the actual ranks vary widely depending on the randomly sampled documents in the CSI, as we found in preliminary experiments. Kulkarni et al. [14] present a family of three algorithms that model shard selection as a process of the CSI documents voting for the shards that they represent. The vote of a shard is the sum of the votes of its documents. The algorithms select shards that have an accumulated vote higher than 0.0001. The three algorithms differ in how they model the strength of a document's vote. In the most effective method Rank-S, the strength of the votes is based on the score from the initial search and decays exponentially according to the rank of the document in this ranking. The base of the exponential decay function is a tuning parameter of the model.

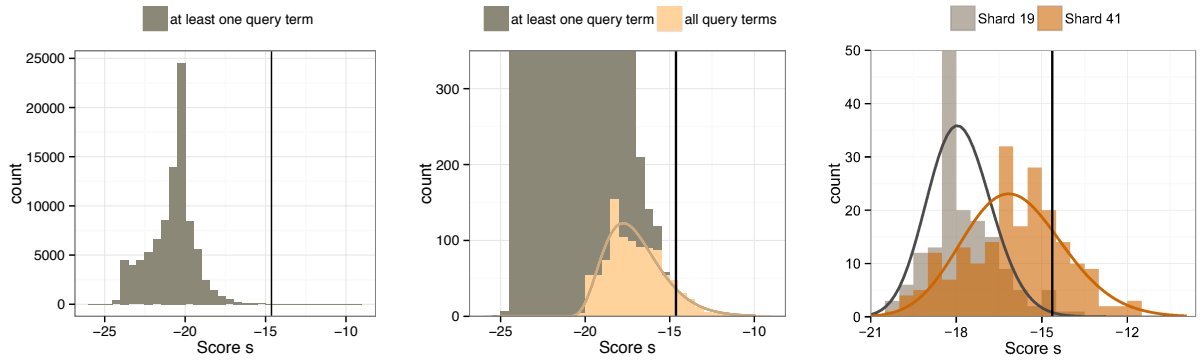
The shard selection algorithm proposed in this paper is vocabulary-based. However, instead of considering all documents in a shard for ranking, it selects shards based on the highly scored documents of a shard similar to the described sample-based methods.

3. SHARD SELECTION USING THE TAIL OF SCORE DISTRIBUTIONS

Before formally introducing our shard selection algorithm in this section, we will explain our reasoning and the intuition behind it.

3.1 Intuition and Reasoning

Abstracting from sharded search, a search engine uses a score function that assigns each document in the collection a score. The documents are then ranked based on that score, and for most effectiveness measures, the top-ranked documents are the most influential. Now, a shard selection algorithm has to identify those shards whose documents can be left out from the complete ranking without hurting the effectiveness. We therefore design our shard selection algorithm to leave out shards with no or only few documents in the top of the complete document ranking. The number of



(a) Score distributions of documents with at least one query term.

(b) Score distributions focusing on the right tail.

(c) Score distribution in two shards of documents with all query terms.

Figure 1: Intuition of the shard selection process for the web-track query 843 *pol pot*. The vertical bar indicates the cut-off score of the $n_c = 100$ highest scored documents. The shown distributions are Gamma distributions fitted using the maximum likelihood and multiplied by the number of documents in the distribution.

considered top-ranked documents can vary depending on the search scenario. Our algorithm therefore considers a number of n_c highly scored documents. Expressed differently, these documents are the right tail of the collection's score distribution in response to the given query, and hence we name our shard selection algorithm Taily. Figure 1a shows the score frequency distribution of the query 843 *pol pot* in the Gov2 collection using language model scores. A search engine may want to preserve, e.g., $n_c = 100$ top-ranked documents. For our example, this corresponds to the documents that are assigned a language model score of -14.6 or higher for this query.

The more accurate our model is in the right tail of the score distribution, the more accurate we can expect our shard selection to be. Score distributions are typically dominated by low scores of documents that contain no or only few of the query terms. We do not expect that the tail of these score distributions can be accurately modeled. Instead, we model the score distribution of documents that contain *all* query terms, which include the top-ranked documents for most queries and empirically leads to a better fit of the right tail, see Figure 1b.

Taily selects shards based on the number of documents with a score above the cut-off score of the top-$n_c$documents. To estimate this number, Taily fits the score distribution in each of the shards, from which the probability of a document in this shard with a score above that cut-off point can be readily calculated. Because shards differ in size and absolute numbers of high-scoring documents, a shard with a low right-tail probability might still have a reasonable number of documents with scores above cut-off. We therefore also estimate the total number of documents that participate in the considered score distribution and select shards based on the expected number of documents that are above the cut-off score. For example, Figure 1c shows the empirical and fitted score distribution[1] of the shards 19 and 41 of topical shards generated by Kulkarni and Callan [13]. Most documents in the selected tail of the collection's score distribution belong to shard 41. Therefore, Taily prefers shard 41 over shard 19 for this query.

A popular way to estimate score distributions is to use scores of document samples from the top of the ranking [1]. However, because we avoid the use of a central sample index, this type of methods is not applicable here. Instead, following Kanoulas et al. [12], we infer the query dependent score distribution from query independent feature distributions that are summed in the score function. The parameters of the feature distributions form Taily's shard representation, which can be calculated offline.

In the following, we develop the Taily algorithm more formally. Section 3.2 introduces the used score function and Section 3.3 describes the statistics that form Taily's shard representation. Section 3.4 shows how these statistics are used to estimate the parameters of the score distributions for the shards, and for the whole collection. Section 3.5 describes how the number of documents with all query terms in the whole collection and per shard can be estimated. Using the estimates for the score distribution and the number of documents, we define Taily's shard selection criterion in Section 3.6.

3.2 Notation and Score Functions

We use the following notation throughout this paper. Queries and documents are denoted by lower case q and d respectively. Sets of documents are denoted by $\mathcal{D}$, and particular set is indicated by a subscript. In particular, let $\mathcal{D}_c$ be the set of documents in the total considered collection, and let $\mathcal{D}_1, ..., \mathcal{D}_N$ be the sets of documents of the N shards of this collection. We often refer to either the set of documents in the collection or the shards, for which we use the subscript i. Terms are denoted by lower case t, the query terms of a query q are denoted by $\vec{q}$. The length of document d is denoted by $dl(d)$, the frequency of term t in document d is written $c(t, d)$, and the number of documents from set $\mathcal{D}_i$ that contain term t at least once is given by $c(t, \mathcal{D}_i)$.

Taily infers a query's score distribution from the distributions of the features that constitute the query's score func-

[1]Note that Fig. 1 displays histograms with absolute frequencies. The fitted lines are the estimated density functions (based on the Gamma distribution), rescaled by the total number of documents included and its bin width, in order to allow visual comparison with the histograms.

tion. In general, our algorithm can be used with any score function that is a weighted sum of term-related feature values. Note that we consider score functions independently from their theoretical motivation. To facilitate experiments, which require a particular score function, we focus in this paper on the query likelihood model, as implemented in the Indri search engine[2]. The query likelihood model uses for a term t in a document d a term feature $f_t(d)$, which is defined as follows:

$$f_t(d) = \log\left(\frac{c(t,d) + \mu P(t|\mathcal{D})}{dl(d) + \mu}\right) \quad (1)$$

where $P(t|\mathcal{D}) = \frac{\sum_d c(t,d)}{\sum_d dl(d)}$ is the collection prior of term t, and μ is the Dirichlet smoothing parameter. Note that the term features in (1) are query independent. The score function $s(d)$ of a document d for a query q is a sum of the features for the query terms:

$$s(d) = \sum_{t \in \vec{q}} f_t(d). \quad (2)$$

In this paper we focus on score functions based on features that are related to a single query term. In future work, we plan to extend Taily to capture multi-term features such as ones used in the full dependency model [16], priors such as PageRank or spam scores, see [19] for a possible integration of these features into score functions.

3.3 Statistical Shard Representation

In order to infer the score distributions in shards and the collection, we represent each of them by the distribution parameters of term features. The main statistics of a feature f_t for term t in document set $\mathcal{D}_i$ are the expected value $E_i[f_t]$ and the variance $\text{var}_i[f_t]$ of the feature, which can be calculated as follows:

$$E_i[f_t] = \frac{\sum_{d \in \mathcal{D}_i} f_t(d)}{c(t, \mathcal{D}_i)} \quad (3)$$

$$E_i[f_t^2] = \frac{\sum_{d \in \mathcal{D}_i} f_t(d)^2}{c(t, \mathcal{D}_i)}$$

$$\text{var}_i[f_t] = E_i[f_t^2] - E_i[f_t]^2 \quad (4)$$

where $E_i[f_t^2]$ is the expected squared feature value. These quantities can be calculated by a single scan through the collection.

The language model score function used in this paper produces negative values. However, the Gamma distribution that we use for the score function is defined for positive values. To be able to shift the score distribution in the next section, we also store for each feature f its minimum value in the collection c:

$$min_c[f] = min\{f(d) | d \in \mathcal{D}_c, c(t,d) > 0\}$$

The expected feature values from (3), the feature variances in (4), and the above minimum values, form the representation used to calculate the score distribution in the shards and the total collection.

3.4 Inferring Score Distributions

Given the shard representation described in the previous section, we derive the distribution parameters of the query specific score distribution. Because the score function used in this paper produces negative scores, we instead consider a score distribution that is shifted by its minimum value, similar to Arampatzis et al. [2]:

$$s^*(d) = s(d) + \sum_{j=1}^{|\vec{f_q}|} min_c[f_j].$$

To keep our notation lean, we continue using s instead of s^* for the score function, keeping in mind that it is now positive defined. For a document set i, the expected score $E_i[s]$ and the score variance $\text{var}_i[s]$ can be derived from the definition of the score function in (2)

$$E_i[s] = \sum_{j=1}^{|\vec{f_q}|} E_i[f_j] + \sum_{j=1}^{|\vec{f_q}|} min_c[f_j] \quad (5)$$

$$\text{var}_i[s] = \sum_{j=1}^{|\vec{f_q}|} \text{var}_i[f_j] \quad (6)$$

where $\vec{f_q}$ is the feature vector of the query terms in (2), and f_j is the jth feature in this vector. Equation 6 uses the simplifying assumption that the sum of covariances is zero. Note that we verified the validity of this assumption by repeating our experiment taking covariances into account, which did not result in a significant increase in effectiveness.

According to Kanoulas et al. [12], the distribution of language model scores is gamma distributed. The parameters of the distribution in document set i can be derived from the expected score and the variance by using the method of moments:

$$k_i = \frac{E_i[s]^2}{\text{var}_i[s]} \quad (7)$$

$$\theta_i = \frac{\text{var}_i[s]}{E_i[s]} \quad (8)$$

where we used the definition of these parameters. Having the parameters k_i and θ_i for a document set i, we can define its cumulative score distribution function, which yields the probability of documents having a score greater than a score s' in a document set i:[3]

$$cdf_i(s') = P_i(s > s') = 1 - \frac{1}{\Gamma(k_i)}\,\gamma\left(k_i,\, \frac{s'}{\theta_i}\right) \quad (9)$$

where Γ is the Gamma function, γ is the incomplete Gamma function, and k_i and θ_i are the distribution parameters defined above. For the case of the whole collection and the example introduced previously, the values of the cumulative distribution function can be visualized as the percentage of documents with a higher score than -14.6 in Figure 1c.

3.5 Documents With All Query Terms

To make the probabilities from the cumulative density functions comparable, Taily uses the number of documents with all query terms in this set. To reduce the strength of assuming independence between the occurrence of query terms [7], we first estimate the number of documents that contain at least one *any* query term Any_i in a document

[2] `http://www.lemurproject.org/indri/`

[3] Note that cumulative distributions are usually defined in terms of the probability in the left tail. We differ from this practice because it simplifies the mathematical formalism used to describe Taily.

set i:

$$Any_i = |\mathcal{D}_i| \left(1 - \prod_{j=1}^{|\vec{q}|} \left(1 - \frac{c(t_j, \mathcal{D}_i)}{|\mathcal{D}_i|}\right)\right)$$

where the term $\prod_{j=1}^{|\vec{q}|}(1 - \frac{c(t,\mathcal{D}_i)}{|\mathcal{D}_i|})$ estimates the number of documents in document set i that have *none* of the query terms. Among the Any_i documents that contain at least one query term, we estimate the number of documents that contain *all* query terms All_i by assuming independence of the term occurrences:

$$All_i = Any_i \prod_{j=1}^{|\vec{q}|} \frac{c(t_j, \mathcal{D}_i)}{Any_i}. \tag{10}$$

where $c(t_j, \mathcal{D}_i)/Any_i$ is the probability that a document with term t_j appears in the documents in $\mathcal{D}_i$ that have at least one of the query term.

Our experiments show that this estimate produces strong and stable results. Important to note here, is that we want an efficient and lightweight algorithm, also during the preprocessing stage. Therefore, even for two-term queries, instead of counting the mutual term occurrences, quadratic in the vocabulary size, we estimate these based on the single-term occurrences.

3.6 Shard Ranking and Selection Criterion

Given the cumulative score distribution cdf_i and estimated number of documents that contain all query terms All_i for both the whole collection and each shard separately, we define Taily's shard selection criteria. Based on our intuition in Figure 1, we first estimate the cut-off score of a fixed number of top-ranked documents in the collection that at least should be in the sharded ranking. Let this number be n_c, which is a parameter of Taily. The probability of a document in the collection to be among the top-ranked documents can be calculated as:

$$p_c = \frac{n_c}{All_c} \tag{11}$$

where All_c is the estimated number of documents in the collection with all query terms. The cut-off score s_c of the top-n_c documents can be estimated using the inverse of the cumulative density function: $s_c = cdf_c^{-1}(p_c)$ where p_c is the probability defined above.

Using the score distribution in a shard i, we can calculate the probability that a document in this shard has a score higher than the cut-off score s_c: $p_i = cdf_i(s_c)$. The number of documents in shard i that have a score above s_c, written n_i, can then be readily estimated[4] by $n_i = All_i\ p_i$. The number of documents in shard i with all query terms is a mere estimation (see (10)), and the sum of estimates All_i for all shards not necessarily equals the overall estimate All_c. Experimentally, this appeared to introduce inaccuracies in the results. As the improvement of score distribution estimates is an ongoing research topic [1], we limit ourselves here to a simple solution. We assume that the estimation of the expected number of documents in the collection n_c is accurate such that (11) holds. A suitably normalized estimate of n_i is hence

$$n_i = All_i\, p_i\ \frac{n_c}{sum_{j=1}^{N} p_j\, All_j} \tag{12}$$

where the term $All_i\, p_i$ is the unnormalized number of documents in $\mathcal{D}_i$ above score s_c and the right term is a normalization constant ensuring that the estimated number of documents above s_c from each shard j add up to the corresponding number of documents in the whole collection, which is n_c.

We are now able to define Taily's shard selection criterion $sel(q)$ for a query q that selects shards with an estimated number of documents in the top-m above a threshold:

$$sel(q) = \left\{ i \middle| i \in 1...N, n_i > v \right\} \tag{13}$$

where i is a shard index, and v is the selection threshold. Note that it can be beneficial for v to be higher than 0 because of the computational costs for including a shard with only very few estimated documents in the top ranks.

4. EFFICIENCY MEASURES

Before we can evaluate Taily, we have to define measures that quantify the efficiency of shard selection algorithms in terms of used resources and response time. To be comparable to related work, we base our measure on the measure by Kulkarni et al. [14], which calculates the resources used by a shard selection algorithm for a query q by the number of documents that the sharded search has to access:

$$\mathrm{C}_R(q) = \sum_{i=1}^{|sel(q)|} \mathcal{D}_i(\vec{q}) \tag{14}$$

where $sel(q)$ are the shards selected by the algorithm and $\mathcal{D}_i(\vec{q})$ is the number of documents in shard i that match at least one of the query terms $\vec{q}$. As discussed in Section 1, the selection algorithm itself can require substantial resources, which should be reflected in the efficiency measure. We therefore extend the above efficiency measure by a component that reflects the costs of executing the selection algorithm. We arrive at our resource efficiency measure $C_{\mathrm{RES}}(q)$ of a query q:

$$C_{\mathrm{RES}}(q) = \mathrm{C}_{SEL}(q) + \mathrm{C}_R(q) \tag{15}$$

where $\mathrm{C}_R(q)$ is measure by Kulkarni et al. from (14), and $\mathrm{C}_{SEL}(q)$ are the costs for executing the selection algorithm for query q. The selection costs C_{SEL} depend on the type of selection algorithm. For sample-based methods, we set the selection costs $\mathrm{C}_{SEL}(q) = CSI(\vec{q})$, where $CSI(\vec{q})$ is the number of documents in the CSI that have at least one query term. For vocabulary based algorithms, we set $\mathrm{C}_{SEL}(q) = N$, where N is the number of shards in the collection, which is the upper bound of the number of entries in the shard representation for any query term.

Additional to the resource usage, the query response time is another important efficiency aspect to consider [18]. In contrast to the used resources, measures for the response time have to take into account that the selected shards are usually processed in parallel, once the selection algorithm has finished. For the search in the selected shards, the costs therefore mainly depend on the shard with the most matching documents. Similar to the methodology in the evaluation of database systems, we measure the response time by the

[4]In fact, we estimate the number of documents that have a score above s_c *and contain all query terms*. This means that we assume that for the shards to be selected, most documents above cut-off contain all query terms. Experimentally, this appears to hold if s_c is reasonably high, see e.g. Figure 1b.

Table 1: Statistics about the collections used in the experiments of this paper (In entries of the form $X \pm Y$, X is the average and Y is the standard deviation. TB=terrabyte track, WT=web track.).

Collection	Documents	Shards	Query set	Query length	Rel. Docs
Gov2	25M	50	TB '04-'06 (701-750)	3.1($\pm$0.97)	180($\pm$148)
CluewebB	50M	100	WT '09+'10 (1-100)	2.1($\pm$1.36)	49($\pm$42)
CluewebA	250M	1000	WT '09+'10 (1-100)	2.1($\pm$1.36)	124($\pm$75)

number of accessed data items on the longest execution path (ignoring implementation dependent aspects):

$$C_{\text{TIME}}(q) = \text{C}_{SEL}(q) + max_{i=1}^{|sel(q)|} \{\mathcal{D}_i(\vec{q})\}\,. \qquad (16)$$

where $1, ..., |sel(q)|$ are the selected shards and the other symbols carry the same definition as in (15). We report average values C_{RES} and C_{TIME} over a considered query set, similar to reporting the mean average precision instead of individual average precision values. Note that the measures in this section consider the number of accessed items, which are documents in shards or CSIs, and shards in vocabulary-based methods. Another choice would have been to consider the number of accessed postings for these items in the inverted files of the query terms.

5. EXPERIMENTS

In this section, we describe the experiments that we conducted to evaluate our shard selection algorithm Taily. We aligned our experiments to the ones from the recent publication by Kulkarni et al. [14], to ensure comparability of our work to the state-of-the-art in shard selection. We proceed as follows: first, we describe the experimental setup, second, we describe each experiment and its results, and finally, we discuss the findings.

5.1 Setup

Table 1 describes the collections and query sets that we used. The collections represent modern retrieval collections of a medium to large size. We used the shards defined by the topical clustering algorithm by Kulkarni and Callan [13]. Due to spam, the effectiveness of the search on the full ClueWebA collection of $500M$ documents was weak. We therefore removed the documents whose spam scores were among the 50% highest scores according to the fusion method by Cormack et al. [8].

We implemented the experiments using the hadoop mapreduce framework, directly answering queries and generating statistics from the full text of the collection; similarly to the approach described in [11]. We used the Krovetz stemmer for both the document text and the queries. We did not use stopwording. For the exhaustive search and the searches in the central sample index (CSI), we used the full dependency model [16] with the parameter setting $(0.8, 0.1, 0.1)$ for single term features, unordered term features and ordered term features respectively as recommended by Metzler et al. [17] and used in Kulkarni et al. [14]. The Dirichlet smoothing parameter was set to $\mu = 2500$.

We chose one baseline from each of the two classes of selection algorithms presented in Section 2. As a vocabulary-based algorithm, we used the popular CORI algorithm [6]. As a sample-based algorithm we chose Rank-S by Kulkarni et al. [14], which showed significantly stronger performance than REDDE [21] and SUSHI [22], two other state-of-the-art shard selection algorithms. Note that we added for Rank-S

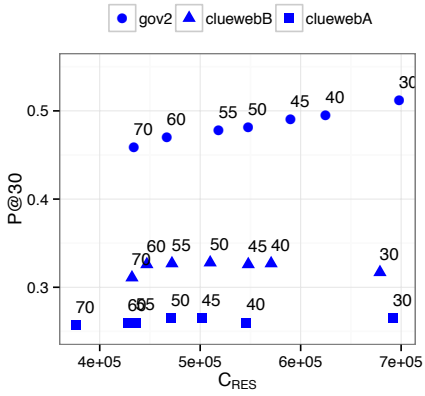
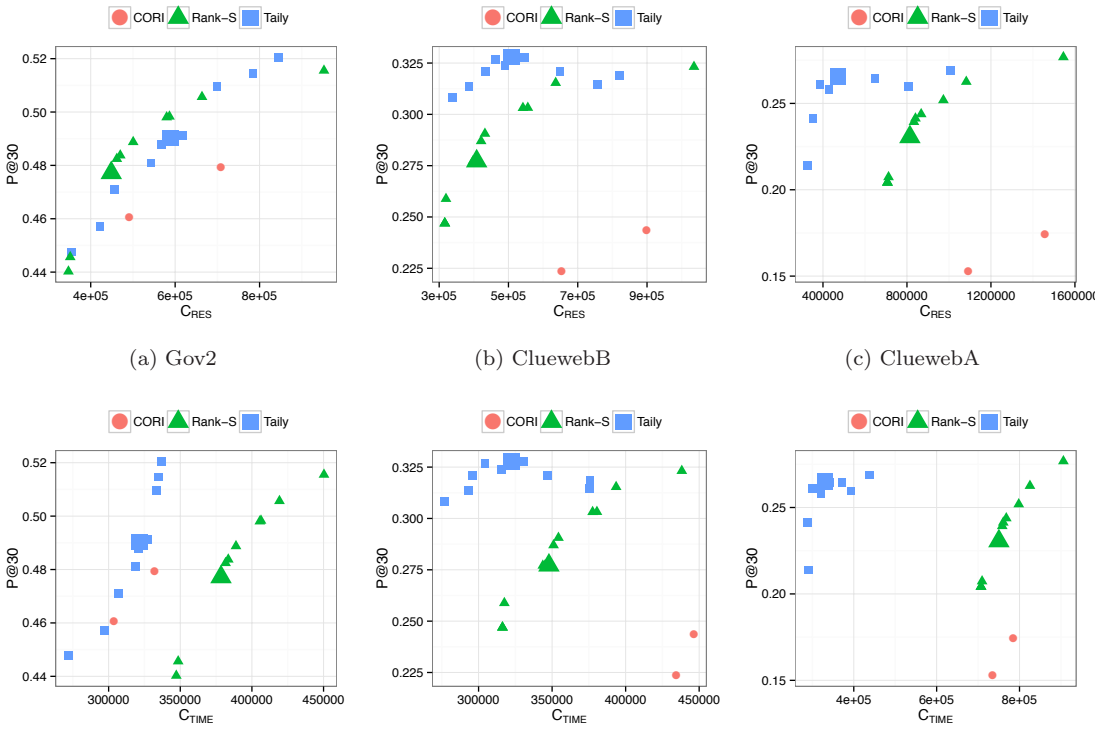


Figure 2: Sensitivity of Taily with n_c=400 according to the threshold parameter v (as indicated in the plot).

the minimum score of the score of the full dependency to make the scores positive. This was not reported by Kulkarni et al. [14] but it was important to achieve results comparable to the ones in the original publication. The documents for the central sample index used by Rank-S were uniformly sampled without replacement from each shard until a percentage P of the shard's size was reached. We ensured that each shard was represented by at least 100 documents. To rule out random effects, we repeated the runs with Rank-S 50 times, producing 50 CSIs with potentially different shards selections. Unless stated otherwise, the reported performance measures are the averages of the 50 repetitions. The average approximates the expected performance for a random CSI of this size. Note that performing statistical significance tests using these expected performance values is mathematically speaking problematic. We still report the results of these tests as an indication of the strength of the performance change. Note that Kulkarni et al. [14] use only a single CSI in their results. Therefore, their numerical results do not necessarily correspond to ours.

The size of the CSI can influence the performance of Rank-S. Unless stated otherwise, we used $P = 0.02, 0.01$ and 0.01 for Gov2, ClueWebB and for ClueWebA respectively. For Gov2 and ClueWebB these settings resulted in a CSI that was roughly as big as an average shard in the respective collection. For ClueWebA we chose $P = 0.01$ because using $P = 0.001$, which corresponds to the size of an average shard, caused poor effectiveness.

We used the effectiveness measures precision at ten, thirty and hundred (P@10, P@30, P@100), mean average precision (map), and ndcg at ten ($ndcg$@10). When we focused on a single effectiveness measure, we chose P@30 because it was more stable than P@10 for all selection algorithms but still reflected a precision oriented search task. To mea-



(a) Gov2 (b) CluewebB (c) CluewebA

(d) Gov2 (e) CluewebB (f) CluewebA

Figure 3: Efficiency-effectiveness tradeoff for CORI, Rank-S, and Taily. The following tradeoff parameter settings of each method were used. CORI: $n \in \{2, 3\}$ (higher values were always outside the limits of the x-axis), Rank-S: $B \in \{2, 5, 10, 30, 50, 70, 100, 200, 500\}$ (lower values caused lower efficiency), Taily: $n_c \in \{200, 250, 300, 350, 400, 600, 800, 1000, 1500, 2000\}$.

sure efficiency, we used the resource efficiency C_{RES} and the response time C_{TIME} described in Section 4.

5.2 Threshold Parameter

The threshold parameter v, defined in Section 3.6, specifies the minimum number of documents in the right tail of the collection's score distribution that a shard should have to be selected. Figure 2 shows a sensitivity analysis of Taily towards changes of v by displaying the resulting P@30 and C_{RES} measures (here, we used a fixed tradeoff parameter of $n_c = 400$ but the results were similar for other values of n_c). The effectiveness of Taily was robust against changes of v, and increased slightly for Gov2. The parameter setting $v = 50$ caused efficiency and effectiveness around the median of the tested values. We chose this parameter setting for the rest of our experiments.

5.3 Efficiency-Effectiveness Comparison

An important characteristic of a shard selection algorithm is the tradeoff it provides between efficiency and effectiveness. A search engine operator may want to invest more resources to ensure high effectiveness, or to make more efficient use of resources and accept worse effectiveness. CORI, Rank-S and Taily each have a parameter that determines the tradeoff between efficiency and effectiveness. For CORI the parameter n states a fixed number of the highest ranked shards that are selected, second, Rank-S uses the parameter B that determines the influence of a CSI document on the selection of the shard it belongs to (see Section 2), finally, Taily uses the parameter n_c that determines the number of top-ranked documents that should be included in the results of the sharded search.

Figure 3 compares the tradeoff that CORI, Rank-S, and Taily provided in the indicated parameter range. The x-axes show the resource usage C_{RES} or the response time C_{TIME}. The y-axes show the effectiveness in terms of precision P@30.

Figure 3a-Figure 3c show the tradeoff between C_{RES} and P@30. For Gov2 the tradeoff is similar for CORI, Rank-S and Taily at low efficiency values. Rank-S and Taily provide a similar tradeoff between effectiveness and efficiency over all parameter settings. For CluewebB the efficiency of CORI was always lower than the one of Rank-S and Taily. The effectiveness of Rank-S was lower than the one of Taily for a resource usage of $C_{\text{RES}} < 60,000$. With higher resource usage, both methods had similar effectiveness. For CluewebA Taily showed a higher effectiveness than Rank-S until $C_{\text{RES}} = 90,000$.

Figure 3d-Figure 3f show the tradeoff between C_{TIME} and P@30. For Gov2 CORI performed similar to Taily. All parameter settings of Rank-S had a larger response time than the ones of Taily. To achieve comparable effectiveness, the response time of Rank-S was roughly 15% larger than the one from Taily. For CluewebB CORI's performance was low. Taily showed higher effectiveness than Rank-S until

Table 2: Effectiveness and efficiency comparison between Taily and Rank-S for the indicated parameter settings. (▲ and ▼ indicate statistically significant improvement or regression respectively compared to the Exhaustive search, using a two-sided t-test with $p{-}value{<}0.05$. Percentages state the efficiency change compared to the Rank-S method.)

Method	P@10	P@30	P@100	*map*	*ndcg*@10	Shards	C_{RES}		C_{TIME}	
Exhaustive	0.58	0.52	0.42	0.34	0.49	50.0	4.92M		0.51M	
CORI (n=3)	0.57	▼0.48	▼0.36	▼0.25	0.48	3.0	0.71M		0.33M	
Rank-S (B=50 P=0.02)	▼0.55	▼0.48	▼0.37	▼0.24	▼0.45	1.5	0.45M		0.38M	
Taily ($n_c = 400, v = 50$)	0.56	▼0.48	▼0.38	▼0.27	▼0.46	2.6	0.55M	22.1%	0.32M	−15.7%

(a) Gov2

Method	P@10	P@30	P@100	*map*	*ndcg*@10	Shards	C_{RES}		C_{TIME}	
Exhaustive	0.29	0.32	0.22	0.20	0.24	100.0	10.15M		0.47M	
CORI (n=3)	0.25	▼0.24	▼0.16	▼0.13	0.21	3.0	0.90M		0.45M	
Rank-S (B=50 P=0.01)	0.31	▼0.28	▼0.18	▼0.15	▲0.27	1.5	0.40M		0.34M	
Taily ($n_c = 400, v = 50$)	0.31	0.33	0.22	▼0.18	▲0.27	2.7	0.51M	26.1%	0.32M	−6.5%

(b) CluewebB

Method	P@10	P@30	P@100	*map*	*ndcg*@10	Shards	C_{RES}		C_{TIME}	
Exhaustive	0.29	0.27	0.18	0.11	0.19	1000.0	48.78M		1.02M	
CORI (n=3)	▼0.20	▼0.17	▼0.11	▼0.06	▼0.14	3.0	1.46M		0.78M	
Rank-S (B=50 P=0.01)	0.29	▼0.24	▼0.15	▼0.08	0.19	2.2	0.90M		0.77M	
Taily ($n_c = 400, v = 50$)	0.30	0.27	0.17	▼0.09	0.20	2.5	0.47M	−47.6%	0.33M	−57.3%

(c) ClewebA

roughly $C_{\text{TIME}} = 40,0000$. For ClewebA CORI's performance is again low. Compared with ClewebB the difference of Rank-S and Taily in terms of C_{TIME} is larger.

The results in Figure 3 show that the effectiveness of Rank-S varies with different settings of B, unlike the results by Kulkarni et al. [14] (their Figures 2 and 3) that suggest that the effectiveness of Rank-S is almost unaffected by the parameter. Because the particular sampled documents in the central indices used by Kulkarni et al. were not available, we did not further investigate these differences. They might be explained by two important differences in the experimental setup: first, we use a smaller CSI size ($P = 0.02$ and $P = 0.01$ vs. $P = 0.04$ by Kulkarni et al.), and second, we report the average performance over random samples, instead of results on a single sample.

5.4 Detailed Effectiveness Analysis

We also investigated the effectiveness in terms of multiple measures at a fixed parameter settings. For CORI we chose a shard cut-off value of $n = 3$. Although larger values sometimes produced better effectiveness, the efficiency was too low to be comparable to Rank-S and Taily. We set the Rank-S parameter $B = 50$ for all three collections because Kulkarni et al. also used this value for ClewebB. For Taily we set $n_c = 400$ and $v = 50$. The efficiency and effectiveness for these parameters is shown as larger points in Figure 3. We also display the performance for the exhaustive search, as a reference. For the displayed efficiency of the exhaustive search, we assumed a parallel search of all shards with zero costs for the selection ($C_{SEL} = 0$ in (15) and (16)).

Table 2a shows the results for the Gov2 collection. CORI shows similar performance than Rank-S and Taily in terms of P@10. The resource efficiency was the lowest among the three methods. The response time C_{TIME} was comparable to the one of Taily. Rank-S showed comparable effectiveness to Taily with only *map* being lower. The resource usage of Rank-S was the lowest among the three runs. The response time was the largest. On average, Taily and Rank-S selected 2.6 and 1.5 shards on average respectively.

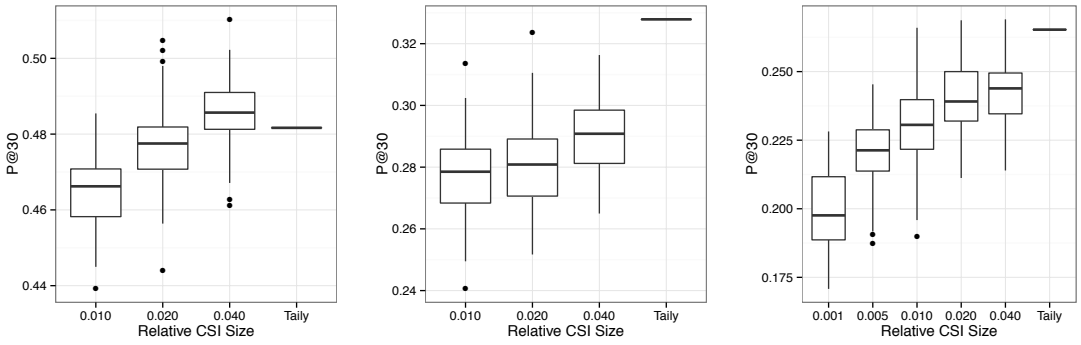
Table 2b shows the results for the ClewebB collection. CORI showed weaker efficiency and effectiveness than Rank-S and Taily. Taily's effectiveness was stronger for all efficiency measures. In terms of C_{RES}, Taily used 26.1% more resources than Rank-S, while the response time improved by 6.5%. Note that Rank-S and Taily improved significantly upon exhaustive search for *ndcg*@10. This is possible if top-ranked non-relevant documents in the exhaustive ranking are from shards, which are ignored by the shard selection algorithm. We consider this phenomenon as a property of the data and therefore did not make any further investigations.

Table 2c shows the results for the ClewebA collection. CORI showed weak effectiveness and efficiency. Rank-S performed significantly worse than exhaustive search for P@30, P@100 and *map*. Taily showed similar or better effectiveness compared to Rank-S. Compared to Rank-S, the efficiency improved 47.6% and 57.3% upon Rank-S in terms of C_{RES} and C_{TIME} respectively.

Note that although Taily used more resources in Gov2 and ClewebB than Rank-S for the indicated parameter settings, there exist other parameter settings where this was not the case, see Figure 3. Furthermore, the used effectiveness measures depend on a high coverage of judgments in the top-ranked documents. For ClewebA only roughly 50% of the top-30 documents were judged. This coverage was similar for all investigated shard selection algorithms. As a result, the absolute effectiveness values are not exact. However, we believe that the relative effectiveness comparison among the shard selection algorithms would be similar under complete coverage because the selective rankings are derived from the exhaustive ranking and only differ by few added or removed shards per query, which is likely to average out.

5.5 Influence of Document Sample on Rank-S

The performance of Rank-S depends on the CSI it uses



(a) Gov2 (b) CluewebB (c) CluewebA

Figure 4: Influence of the document sample on the effectiveness of Rank-S. (Rank-S used $B = 50$. Taily does not use of a CSI and his parameters were $n_c = 400$ and $v = 50$).

for the initial search in two ways: first, the number of documents, assuming that a larger CSI also causes a more accurate selection, and second, exactly which documents are sampled. Variation in performance can influence the comparison of Rank-S and Taily. Therefore, we investigated the performance variability of Rank-S along these axes.

Figure 4 summarizes the results of this experiment, using the P@30 measure. As a reference, we also plot the effectiveness of Taily. Figure 4a shows the results for Gov2. The median effectiveness increased by roughly 5% between $P = 0.01$ and $P = 0.04$. The lowest achieved performance was roughly 8% weaker than the median performance. Taily showed similar effectiveness as the median performance of Rank-S with $P = 0.04$.

Figure 4b shows the results for CluewebB. The median effectiveness was more stable when changing the CSI size than with Gov2. The lowest achieved search performance was roughly 10% lower than the median performance. Taily showed better effectiveness than all measurements for Rank-S.

Figure 4c shows the results for CluewebA. The median search performance increases by roughly 18% between $P = 0.001$ and $P = 0.04$. The lowest achieved performance was roughly 15% lower than the median performance. Taily's effectiveness was en par with the best-measured effectiveness of Rank-S with $P = 0.02$ and $P = 0.04$.

5.6 Discussion

We discuss the results presented in this section. Taily depends on the following two parameters: the tradeoff parameter n_c that determines the number of high scored documents of the collection, and the threshold parameter v that determines minimum of this estimate for are a shard to be selected, which was introduced to suppress estimation errors. Section 5.2 and Section 5.3 showed that both parameters can be set reliably within a range of values, resulting in strong performance. Nevertheless, the parameter values are unintuitive. For example, the setting of $n_c = 400$ and $v = 50$, used in Section 5.4, means according to the definition of the parameters that shards with 50 documents in the top 400 documents should not be selected. A likely reason for these unintuitive estimates is the crude normalization of the expected values in (12). Therefore, for future work we propose the research of more accurate normalization methods, which possibly improves the performance of Taily further.

The results of the comparison of CORI, Rank-S and Taily yielded a number of findings. First, the performance of the vocabulary-based algorithms shows opposite trends depending on the collection size. While CORI's effectiveness decreases, Taily effectiveness increases. A likely explanation for CORI's performance decrease is its known bias towards bigger shards, which do not necessarily contain many relevant documents, which causes efficiency and effectiveness to drop. Therefore, Taily's approach to model the tail of the score distribution in each shard selects substantially smaller shards with more relevant documents than CORI's approach to model *all* documents. Second, for each parameter setting of Rank-S there was a parameter setting of Taily with higher efficiency and similar or higher effectiveness compared to the former's expected performance. This was in particular true for the response time. This shows that the selection costs C_{SEL} for executing Taily were substantially lower than the ones for Rank-S and Taily's vocabulary-based approach can select shards with a comparable number of relevant documents as Rank-S. Finally, because of the shape of the achieved efficiency - effectiveness tradeoff combinations, we propose that finding tradeoff settings for a given efficiency is easier to achieve for Taily than for Rank-S. An exception is the response time of Taily in Gov2, which increases at a high rate, causing small differences in efficiency to cause large changes in effectiveness.

The effectiveness of Rank-S is strongly correlated with the CSI size. For example, for CluewebA an increase of the CSI size from $P = 0.001$ to $P = 0.01$ improved the median effectiveness by 16%. At the same time, larger CSI sizes consume more storage space, which also makes them less efficient. Therefore, it is likely that Rank-S does not scale to collections lager than CluewebA.

6. CONCLUSIONS AND FUTURE WORK

We introduced Taily, a novel shard selection algorithm that is based on highly scored documents in the tail of the collection's score distribution. The scores are assumed to be Gamma distributed and estimated from statistics about features related to the query terms of the language model score function. Taily is therefore a member of vocabulary-based shard selection algorithms that represent shards by statistics of terms in the vocabulary.

We evaluated Taily on three large web collections (Gov2,

ClewebB and ClewebA) using topically clustered shards defined by Kulkarni and Callan [13]. Compared to the popular vocabulary-based method CORI [6], Taily showed better efficiency and effectiveness. Compared to Rank-S [14], a state-of-the-art, sample-based shard selection algorithm, Taily achieved similar or better effectiveness using less resources. Especially for larger collections and CSIs, the vocabulary-based Taily used less resources than the sample-based Rank-S although it selected on average more shards. The improvement of the response time compared to Rank-S was larger than the improvement of the resource usage. Taily showed his highest effectiveness with a shorter response time than the shortest response time measured for Rank-S over a wide range of parameters. For ClewebA, Taily achieved its best performance in roughly 50% of the response time of Rank-S.

Taily does not use document samples and therefore does not need to answer some of the questions that sample-based methods need to answer, such as what samples to take, and what would be a reasonable size of the central sample index (CSI). We investigated possible answers to these questions for Rank-S, and found that the effectiveness of Rank-S decreases with the CSI size. For example, the effectiveness decreased by 20% between using a commonly used CSI size of 4% and 0.1% for the ClewebA collection, where 0.1% corresponded to the size of an average shard. We also found that, for a given CSI size, the lowest effectiveness of Rank-S in 50 CSIs consisting of different document samples was roughly 10% lower than the median performance. The dependence of effectiveness on CSI size and CSI sample is a weakness of sample-based methods. This weakness can also be seen as an advantage of vocabulary-based methods, like Taily, which do not depend on a CSI.

This work focused on shard selection in a cooperative environment using topically clustered shards. We believe that the basic ideas behind Taily can also be applied to database selection in a uncooperative, federated search scenario [4, 20]. In future work we plan to investigate methods to gather the statistics required by Taily in a federal search scenario, to evaluate whether its performance also applies to this setting.

Acknowledgments

The work reported in this paper was funded by the EU Project AXES (FP7-269980), the Netherlands Organisation for Scientifc Research (NWO) under project 639.022.809, the University of Twente in The Netherlands, Ghent University in Belgium, and iMinds (Interdisciplinary institute for Technology), a research institute founded by the Flemish Government. This work is part of the programme of BiG Grid, the Dutch e-Science Grid, which is financially supported by the Nederlandse Organisatie voor Wetenschappelijk Onderzoek (Netherlands Organisation for Scientific Research, NWO). The authors want to thank the Big Grid team for their outstanding technical support, as well as Jamie Callan and the anonymous reviewers for their fruitful input.

Bibliography

[1] A. Arampatzis and S. E. Robertson. Modeling score distributions in information retrieval. *Information Retrieval*, 14:1–21, 2010. doi: 10.1007/s10791-010-9145-5.

[2] A. Arampatzis, N. Nussbaum, and J. Kamps. Where to stop reading a ranked list? In *TREC'08*, 2008.

[3] J. Arguello, J. Callan, and F. Diaz. Classification-based resource selection. In *CIKM'09*, pages 1277–1286. ACM, 2009. doi: 10.1145/1645953.1646115.

[4] R. Baeza-Yates, C. Castillo, F. Junqueira, V. Plachouras, and F. Silvestri. Challenges on distributed web retrieval. In *ICDE'07*, pages 6–20, 2007. doi: 10.1109/ICDE.2007.367846.

[5] R. Baeza-Yates, V. Murdock, and C. Hauff. Efficiency trade-offs in two-tier web search systems. In *SIGIR'09*, pages 163–170. ACM, 2009. doi: 10.1145/1571941.1571971.

[6] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *SIGIR '95*, pages 21–28. ACM, 1995. doi: 10.1145/215206.215328.

[7] W. S. Cooper. Some inconsistencies and misidentified modeling assumptions in probabilistic information retrieval. *ACM Trans. Inf. Syst.*, 13(1):100–111, 1995. doi: 10.1145/195705.195735.

[8] G. V. Cormack, M. D. Smucker, and C. L. A. Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *CoRR*, abs/1004.5168, 2010.

[9] S. Ganguly, W. Hasan, and R. Krishnamurthy. Query optimization for parallel execution. In *SIGMOD'92*, pages 9–18, NY, USA, 1992. ACM. doi: 10.1145/130283.130291.

[10] L. Gravano and H. Garcia-Molina. Generalizing gloss to vector-space databases and broker hierarchies. In *VLDB'95*, pages 78–89. Morgan Kaufmann Publishers Inc., 1995. ISBN 1-55860-379-4.

[11] D. Hiemstra and C. Hauff. Mapreduce for information retrieval evaluation: "let's quickly test this on 12 tb of data". In *Multilingual and Multimodal Information Access Evaluation.* Springer Verlag, 2010.

[12] E. Kanoulas, K. Dai, V. Pavlu, and J. A. Aslam. Score distribution models: assumptions, intuition, and robustness to score manipulation. In *SIGIR'10*, pages 242–249. ACM, 2010. doi: 10.1145/1835449.1835491.

[13] A. Kulkarni and J. Callan. Document allocation policies for selective searching of distributed indexes. In *CIKM '10*, pages 449–458. ACM, 2010. doi: 10.1145/1871437.1871497.

[14] A. Kulkarni, A. S. Tigelaar, D. Hiemstra, and J. Callan. Shard ranking and cutoff estimation for topically partitioned collections. In *CIKM'12*. ACM, 2012. doi: 10.1007/s10791-006-9014-4.

[15] I. Markov. Modeling document scores for distributed information retrieval. In *SIGIR'11*, pages 1321–1322. ACM, 2011. doi: 10.1145/2009916.2010180.

[16] D. Metzler and W. B. Croft. A markov random field model for term dependencies. In *SIGIR'05*, pages 472–479. ACM, 2005. doi: 10.1145/1076034.1076115.

[17] D. Metzler, T. Strohman, H. Turtle, and W. Croft. Indri at trec 2004: Terabyte track. In *TREC 2004*, 2004.

[18] A. Moffat, W. Webber, J. Zobel, and R. Baeza-Yates. A pipelined architecture for distributed text query evaluation. In *Information Retrieval*, volume 10, pages 205–231. Kluwer Academic Publishers, 2007. doi: 10.1007/s10791-006-9014-4.

[19] D. Nguyen and J. Callan. Combination of evidence for effective web search. In *TREC 2010*, 2010.

[20] M. Shokouhi and L. Si. *Federated search*, volume 5. Now Publishers Inc, 2011. doi: 10.1561/1500000010.

[21] L. Si and J. Callan. Relevant document distribution estimation method for resource selection. In *SIGIR'03*, pages 298–305. ACM, 2003. doi: 10.1145/860435.860490.

[22] P. Thomas and M. Shokouhi. Sushi: scoring scaled samples for server selection. In *SIGIR'09*, pages 419–426. ACM, 2009. doi: 10.1145/1571941.1572014.

[23] J. Xu and W. Croft. Cluster-based language models for distributed retrieval. In *SIGIR'99*, pages 254–261. ACM, 1999.