Mirex and Taily at TREC 2013

Robin Aly[1], Djoerd Hiemstra[2], Dolf Trieschnigg[1], Thomas Demeester[3]

[1]Human Media Interaction Group University of Twente, [2]Database Group University of Twente, [3]Ghent University - iMinds

***Abstract*— We describe the participation of the Lowlands at the Web Track and the FedWeb track of TREC 2013. For the Web Track we used the Mirex Map-Reduce library with out-of-the-box approaches and for the FedWeb Track we adapted our shard selection method Taily for resource selection. Here, our results were above median and close to the maximum performance achieved.**

I. Introduction

In this paper we describe the contribution from University of Twente at Text Retrieval Evaluation Conference 2013. We participate in the Web Track and the Federated Web Track, which we also helped organizing. The remainder of this paper is structured as follows. Section II describes our participation in the web track and Section III describes our participation in the Federated Web Track.

II. Web Track Participation

Our experiments are run by MIREX [6][1] (MapReduce Information Retrieval Experiments), a library of MapReduce programs to extract data and sequentially scan document representations. Built on Hadoop, sequential scanning becomes a viable approach. MIREX allows researchers to easily experiment with different retrieval models, because the framework is easy to extend.

Run	P@5	P@10
ut22xact	0.404	0.384
ut22base	0.380	0.398
ut22spam	0.348	0.358

TABLE I: Precision at 5 and 10 (50 queries)

Table I shows the precision at 5 and 10 results of the three official Web Track runs. All runs use anchor texts as document representations, which we made available for download.[2] The first run, tagged `ut22xact`, matches the exact query string to the anchors, and ranks the documents by the number of exact matches found. This run finds exact matches for 41 out of 50 queries. We appended the results from the second run, i.e. those documents that were not already found by exact matches, to the run as the final result. The second run, tagged `ut22base`, uses a simple unigram language model with linear interpolation smoothing and $\lambda = 0.95$. A run without smoothing (or $\lambda = 1$) retrieves the exact same top 10 documents for 47 out of 50 queries, and therefore also achieves the same precision at 5 and 10 documents. The third run, tagged `ut22spam`, uses the same ranking as the second run but removes the 50 % most spammiest documents from the Waterloo spam rankings [5].[3] The experimental results show that `ut22xact`, exact matching of the full query string, outperforms the other runs for precicion at 5 documents retrieved, whereas `ut22base`, the language model, performs best at precision at 10. Although, removing the 50 % spammiest documents helps on various ClueWeb09 test collections, in this case it hurts our results.

Run	P@5	P@10
ut22xact	0.196	0.172
ut22base	0.170	0.168
ut22spam	0.132	0.119

TABLE II: High Relevance Precision (47 queries)

Table II shows the ability of the systems to retrieve documents judged as *highly relevant*, *key*, or *navigational*. So, documents judged as *relevant* were not considered in this evaluation. The results show that clearly, the exact query string matching favours highly relevant documents for 5 and 10 documents retrieved.

Run	J@10	J@20	J@30
ut22xact	1.000	0.894	0.745
ut22base	1.000	0.918	0.782
ut22spam	0.998	0.838	0.629

TABLE III: Fraction of judged documents (50 queries)

Different topics were pooled to different depths because the original depth (20) resulted in too many documents to be judged in the allotted amount of assessing time. Tables III show the effects of the pool depth of the fraction of judged documents for each run. Although the run `ut22spam` was not part of the pool that was judged, it almost has all documents judged in the top 10. Of the runs that did contribute to the pool, at 20 documents retrieved, about 10 % of the documents is not judged. At 30 documents retrieved, this drops to about 22 % to 25 %.

Qrels	total	per query
all judged	14474	290
highly rel. (> 1)	1106	22
relevant (> 0)	4150	83
irrelevant ($= 0$)	10090	202
spam ($= -2$)	234	5

TABLE IV: Number of documents judged

[1]http://mirex.sourceforge.net

[2]http://www.cs.utwente.nl/∼hiemstra/2013/anchor-text-for-clueweb12.html

[3]http://www.mansci.uwaterloo.ca/∼msmucker/cw12spam/

Table IV shows general statistics of the TREC 2013 Web Track collection. Of the total number of documents that are judged, almost 29 % were judged relevant, so it is likely that many more relevant documents would have been found if more resources would have been available for judging.

We tried simple, out-of-the-box approaches to this year's Web track. It is amazing to see that very simple methods, such as counting the number of exact query string matching in anchor texts, provides relatively powerful retrieval results. Search becomes easy if you have a lot of data.

III. FedWeb Track Participation

The Federated Web Track models a distributed search scenario where users send requests to a broker which forwards the requests to a set of search engines that are likely to produce relevant results. The track consists of two tasks: 1) the resource selection task, which requires selecting resources based on resource descriptions a search request and 2) the result merging task, which requires the fusion of the results being returned from search engines. This year we only participated in the resource selection task.

The track provided sample texts and snippets form documents sampled from each search engine. Prior to resource selection, these documents have to be transformed into a resource description. Currently, resource descriptions based directly on the sampled documents in a central sample index are the most popular. However, this approach also requires substantial storage space and administrative overhead when selecting resources.

Our approach is to adapt Taily [1], which we recently proposed for shard selection in centralized search, for federated web search. Instead of using a centralized sample index, Taily uses vocabulary-based resource descriptions based on statistics of term related features in each shard that are used in ranking functions. Compared to this centralized setting, the full collection is only represented by a sample and and the ranking function of each individual search engine is unknown. Therefore our main contributions in this paper is to adapt the Taily method to a setting where only samples of documents are available and the ranking function is unknown.

Taily assumes a gamma distribution for scores of a query, which is inferred from the feature statistic. As we only have only few document sample instead of the full collection, some samples can overestimate the variance of the collection. As the variance strongly influences the cumulative distribution for gamma distributions, we also experiment with normal distributions.

In the following we first present an extended intuition of the Taily algorithm in Section III-A. Section III-B introduces the used score function and Section III-C describes the statistics that form Taily's resource representation. Section III-D shows how these statistics are used to estimate the parameters of the score distributions for the search engines, and for the whole collection. Section III-E describes how the number of documents with all query terms in the whole collection and per search engine can be estimated. Using the estimates for the score distribution and the number of documents, we define Taily's search engine selection criterion in Section III-F. before we adapt the algorithm to federate web search.

A. Intuition and Reasoning

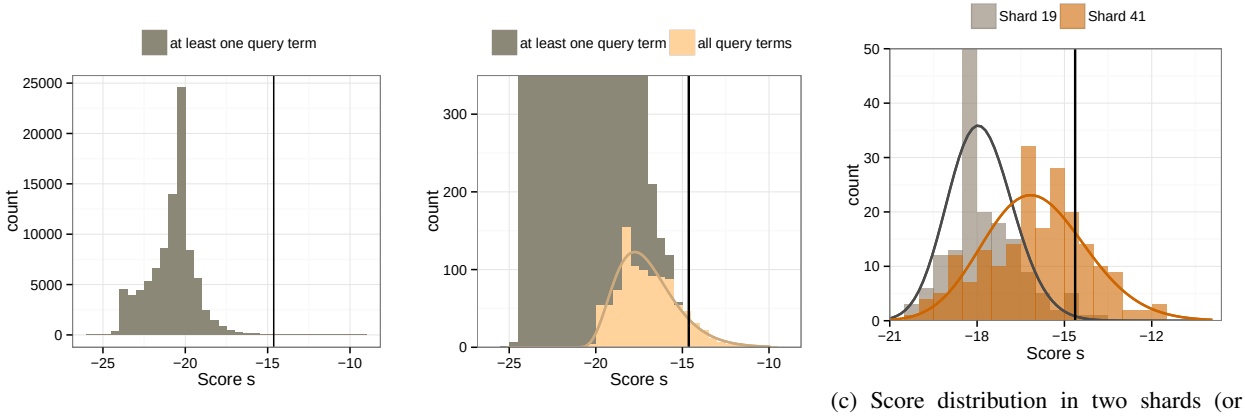
We assume that each search engine uses the same score function that assigns each document a score based on its term frequencies. The broker aims to select those resources that contain the highest ranked documents because they are the most influential for most effectiveness measures. We therefore want Taily to leave out search engines with no or only few documents in the top of the complete document ranking. The number of top-ranked documents that should be considered 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. For example, Figure 1a shows the score frequency distribution of the query 843 *pol pot* in the Gov2 collection using language model scores. A broker may want to preserve, e.g., the $n_c = 100$ top-ranked documents. This corresponds to the documents with a language model score of -14.6 or higher for this query. The broker should therefore only select search engines that index documents with score higher than -14.6.

The more accurate our model is in the right tail of the score distribution, the more accurate we can expect our resource selection to be. Score distributions are typically dominated by low scores of documents that contain no or only few of the query terms. We expect that it is difficult to model the tail of this distribution. 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 resources 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 search engines, from which the probability of a document in this shard with a score above that cut-off point can be readily calculated. Because the resources search engines index differ in size and high-scoring documents, a search engine with a low right-tail probability might still have a reasonable number of documents with scores higher than the 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[4] of the shards 19 and 41 of topical shards generated by Kulkarni and Callan [8]. 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 [2].

[4]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.



(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 (or search engines) of documents with all query terms.

Fig. 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.

However, because we avoid the use of a central sample index, this type of estimation methods is not applicable here. Instead, following Kanoulas et al. [7], 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 resource representation, which can be calculated offline.

B. Notation

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 a particular set is indicated by a subscript. In particular, let $\mathcal{D}_c$ be the set of documents in the total considered collection (the union of documents in all search engines), and let N be the number search engines and $\mathcal{D}_1, ..., \mathcal{D}_N$ be the sets of documents in the respective resource. We often refer to either the set of documents in the collection or the resources, 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 function. In general, our algorithm can be used with any score function that is a weighted sum of term-related feature values.[5] 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[6]. The query likelihood model uses for a term t in a document d a term feature $f_t(d)$, which is defined as follows:

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

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:

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

where the term-related features f are defined in (1).

C. Statistical Shard Representation

In order to infer the score distributions in search engines and the collection, we represent 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, if all documents of the search engine i, $d \in \mathcal{D}_i$ are available:

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

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

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

where $E_i[f_t^2]$ is the expected squared feature value. These quantities can be calculated by a single scan through the collection. In federated web search we usually do not have all documents of a resource $\mathcal{D}_i$ but only a sample of those documents $\mathcal{D}_{i,s} \subseteq \mathcal{D}_i$. Therefore, we approximate these the

[5] Note that we consider score functions independently from their theoretical motivation.

[6] http://www.lemurproject.org/indri/

expectations above through the values in the sample $\mathcal{D}_i$:

$$E_i[f_t] \simeq \frac{\sum_{d \in \mathcal{D}_{i,s}} f_t(d)}{c(t, \mathcal{D}_{i,s})} \quad (5)$$

$$E_i[f_t^2] \simeq \frac{\sum_{d \in \mathcal{D}_{i,s}} f_t(d)^2}{c(t, \mathcal{D}_{i,s})} \quad (6)$$

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.

D. 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. [3]:

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

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)

$$E_i[s] = \sum_{j=1}^{|\vec{f}_q|} E_i[f_j] + \sum_{j=1}^{|\vec{f}_q|} min_c[f_j] \quad (7)$$

$$\text{var}_i[s] = \sum_{j=1}^{|\vec{f}_q|} \text{var}_i[f_j] \quad (8)$$

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 8 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. [7], 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:

$$k_i = \frac{E_i[s]^2}{\text{var}_i[s]} \quad (9)$$

$$\theta_i = \frac{\text{var}_i[s]}{E_i[s]} \quad (10)$$

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:[7]

$$cdf_i(s') = P_i(s > s') = 1 - \frac{1}{\Gamma(k_i)} \gamma\left(k_i, \frac{s'}{\theta_i}\right) \quad (11)$$

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.

E. The Number of 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 [4], we estimate the number of documents that contain at least one, or *any* query term Any_i in document set i. Because we cannot access all documents of search engine's resource, we assume that the sample of search engine i, $\mathcal{D}_{i,s}$ is the *complete* set of documents.

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

where the term $\prod_{j=1}^{|\vec{q}|} (1 - \frac{c(t,\mathcal{D}_{i,s})}{|\mathcal{D}_{i,s}|})$ estimates the number of documents in sample from 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:

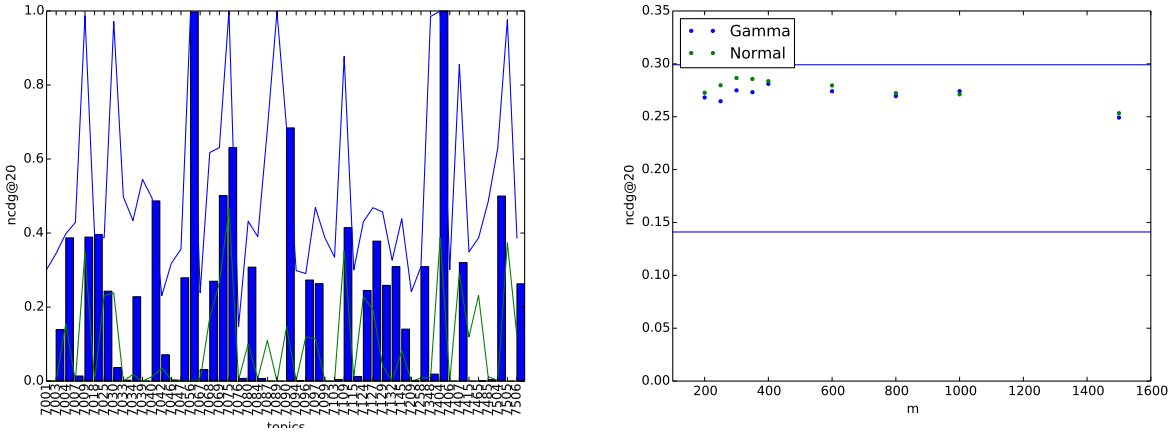
$$All_i = Any_i \prod_{j=1}^{|\vec{q}|} \frac{c(t_j, \mathcal{D}_{i,s})}{Any_i}. \quad (12)$$

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, which requires storage quadratic in the vocabulary size, we estimate these based on the single-term occurrences. Furthermore, assuming that $\mathcal{D}_{i,s} = \mathcal{D}_i$ is a strong assumption. In particular, this estimate will not reflect the different sizes of resources behind each search engine.

F. Search Engine 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 search engine separately,

[7]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.



(a) Performance per query

(b) Parameter sweep over n_c.

Fig. 2: Unofficial experiments FedWeb Track

we define Taily's search engine 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 ranking of the selected search engine. This number n_c is a parameter of Taily. The probability of a document in the collection to be among the top-ranked documents can be calculated as:

$$p_c = \frac{n_c}{All_c} \tag{13}$$

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 search engine i, we can calculate the probability that a document in this search engine has a score higher than the cut-off score s_c: $p_i = cdf_i(s_c)$. The number of documents in search engine i that have a score above s_c, written n_i, can then be readily estimated[8] by $n_i = All_i\ p_i$. The number of documents in search engine i with all query terms is a mere estimation (see 12), and the sum of estimates All_i for all search engines 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 [2], we limit ourselves here to a simple solution. We assume that the estimation of the expected number of documents in the collection, All_c, is accurate, such that 13 holds. A suitably normalized estimate of n_i is hence

$$n_i = All_i\, p_i\, \frac{n_c}{sum_{j=1}^{N} p_j\ All_j}. \tag{14}$$

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

$$sel(q) = \left\{ i \middle| i \in 1...N, n_i > v \right\} \tag{15}$$

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.

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

G. Results

We present the results obtained based on the evaluation data. Table V shows the official overall evaluation scores of the runs `utTailyM400`, which uses Gamma distributions, and `utTailyNormM400`, which uses Normal distributions. Both runs used the parameter setting $n_c = 400$. (We refer to the parameter n_c as M in run names for legacy reasons).

Run	ndcg@20	err@20
Median Performance	0.141	0.008
utTailyM400	0.216	0.010
utTailyNormM400	0.214	0.010
Maximum Performance	0.299	0.020

TABLE V: Official FedWeb Result. Median and maximum performance shown for comparison.

We also performed additional analysis of our method. Figure 2 shows the results. In Figure 2a we investigate the per query performance of `utTailyM400` in terms of $ndcg@20$. The run shows low performance for a number of queries and significantly stronger performance for roughly the other half of the queries. Figure 2b shows the results of varying the parameter n_c. We see that both Taily variants achieve stable performance close to the best achieved performance performance.

H. Conclusions

We presented an adaption of the Taily shard selection algorithm to resource selection in the Federated Web search

scenario. As the expected feature value and the feature variance over the whole collection, which were used for shard selection, can be approximated by the expectation and the variance from a random sample the mathematical formalism was very similar to the original shard selection approach. There are the following areas of future work: 1) we used the number of sampled documents from each search engine as its size, which is clearly an over simplified estimate. In future work we propose to replace this estimate by existing estimates for the size of a database based on a sample. 2) The number of sample documents in Federated Web Search are small compared to the actual size of the search engines, which clearly affects the accuracy of the estimates. We expect that the case where no documents with a query term were sampled can lead to particularly large mistakes. We propose to make estimates more robust by smoothing techniques, which have improved performance performance in retrieval models.

ACKNOWLEDGEMENTS

This work was carried within and funded by several projects. Robin Aly was supported by the EU Project AXES (FP7-269980), carried out at the University of Twente, The Netherlands. Djoerd Hiemstra was supported by the Dutch national program COMMIT. Dolf Trieschnigg was supported by the Folktales as Classifiable Texts (FACT) project, which is part of the CATCH programme funded by the Netherlands Organisation for Scientific Research (NWO). Thomas Demeester was supported by the Ghent University - iMinds in Flanders. The experiments were conducted on the Hadoop Cluster from Sara, the Dutch center for super computing (www.sara.nl).

REFERENCES

[1] R. Aly, D. Hiemstra, and T. Demeester, "Taily: shard selection using the tail of score distributions," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '13. New York, NY, USA: ACM, 2013, pp. 673–682. [Online]. Available: http://doi.acm.org/10.1145/2484028.2484033

[2] A. Arampatzis and S. E. Robertson, "Modeling score distributions in information retrieval," *Information Retrieval*, vol. 14, pp. 1–21, 2010.

[3] A. Arampatzis, N. Nussbaum, and J. Kamps, "Where to stop reading a ranked list?" in *TREC 08*, 2008.

[4] W. S. Cooper, "Some inconsistencies and misidentified modeling assumptions in probabilistic information retrieval," *ACM Trans. Inf. Syst.*, vol. 13, no. 1, pp. 100–111, 1995.

[5] G. Cormack, M. Smucker, and C. Clarke, "Efficient and effective spam filtering and re-ranking for large web datasets," in *arXiv:1004.5168*, 2010.

[6] 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*, ser. Lecture Notes in Computer Science 6360. Springer Verlag, 2010, pp. 64–69.

[7] E. Kanoulas, K. Dai, V. Pavlu, and J. A. Aslam, "Score distribution models: assumptions, intuition, and robustness to score manipulation," in *Proceeding of the 33rd international ACM conference on Research and development in information retrieval*, ser. SIGIR. USA: ACM, 2010, pp. 242–249.

[8] A. Kulkarni and J. Callan, "Document allocation policies for selective searching of distributed indexes," in *Proceedings of the 19th ACM international conference on Information and knowledge management*, ser. CIKM '10. ACM, 2010, pp. 449–458.