Central-Rank-Based Collection Selection in Uncooperative Distributed Information Retrieval

Milad Shokouhi

School of Computer Science and Information Technology
RMIT University, Melbourne 3001, Australia
milad@cs.rmit.edu.au

Abstract. Collection selection is one of the key problems in distributed information retrieval. Due to resource constraints it is not usually feasible to search all collections in response to a query. Therefore, the central component (broker) selects a limited number of collections to be searched for the submitted queries. During the past decade, several collection selection algorithms have been introduced. However, their performance varies on different testbeds. We propose a new collection-selection method based on the ranking of downloaded sample documents. We test our method on six testbeds and show that our technique can significantly outperform other state-of-the-art algorithms in most cases. We also introduce a new testbed based on the TREC GOV2 documents.

1 Introduction

Distributed information retrieval (DIR) has attracted considerable research interest during recent years. Centralized search engines are not capable of indexing the *hidden web* [Raghavan and Garcia-Molina, 2001]. In addition, it is not feasible to crawl and index the web documents with the same rate that they change. DIR has been introduced as a solution to these deficiencies. DIR techniques provide a search service over non-crawlable pages. They also return the recent version of webpages without consuming costly resources for crawling. Distributed search can be divided into three major steps; Firstly suitable collections are selected for a query. Secondly, the query is sent to the selected collections and they search their documents for suitable answers. Finally, the results from selected collections are returned to the broker that then merges them for presentation to the user.

In this paper, we focus on the collection selection stage. In DIR systems, each collection is represented by a set of documents and vocabularies usually known as collection summaries or representation sets. In *cooperative* environments, collections provide the broker with their term statistics and summaries [Gravano et al., 1997]. However, in real-life environments such as the web, collections may be *uncooperative*. In this case, collections are not willing to share their information and the broker should gather small summary sets for each collection by sampling [Callan and Connell, 2001]. Random queries are sent to each collection and results are downloaded and stored as collection representation sets.

G. Amati, C. Carpineto, and G. Romano (Eds.): ECIR 2007, LNCS 4425, pp. 160–172, 2007.

We propose a novel collection selection algorithm that can be used in both cooperative and uncooperative environments. However, we focus on the latter scenario as it is more similar to the practical situations. For each entered query, our method ranks collections according to the ranking of their centrally held sampled documents. The sampled documents from all collections are gathered in a single central index. Each query is executed on this index and the collection weights are calculated according to the ranking of their sampled documents. In the next sections we show that this simple idea can outperform the state-of-the-art methods on many testbeds including our new testbed created from the TREC GOV2 documents.

2 Collection Selection

For a given query, the broker ranks available collections based on the computed similarity values for their representation sets. It is not usually feasible to search all collections for a query. Therefore, the broker selects a few collections that are more likely to return relevant documents. For this purpose, the broker evaluates the similarity of the entered query with the collection summaries. These summaries vary from term statistics in cooperative situations [Callan et al., 1995; Gravano et al., 1997] to a limited number of documents downloaded by sampling for uncooperative environments [Callan and Connell, 2001; Craswell et al., 2000]. The broker chooses those collections that have a summary similar to the query.

Introduced by Gravano et al. [1999], GLOSS uses the document frequency and weight of each term to select suitable collections. However, GLOSS uses unrealistic assumptions such as uniform term weight distribution across all collections. In CVV [Yuwono and Lee, 1997], the fraction of documents inside each collection that contain the query terms is used for collection selection.

CORI [Callan et al., 1995] applies inference networks for collection selection. It has been reported as the most effective method in many papers [Craswell et al., 2000; Powell and French, 2003], but there are question marks over its effectiveness [D'Souza et al., 2004b].

Nottelmann and Fuhr [2003] suggest a decision-theoretic framework (DTF) that selects the best collections while reducing the overall costs such as time and money. Despite having a theoretical foundation, the reported performance of DTF is worse than CORI for short queries.

During recent years, new collection selection algorithms have been claimed to produce better results than CORI [D'Souza etal.,2004a; Si et al., 2002: Si and Callan, 2003a: 2004]. Si et al. [2002] developed a language modeling framework for distributed information retrieval. Their system slightly outperforms CORI in some cases. REDDE [Si and Callan, 2003a] ranks the collections based on the estimated number of relevant documents they contain. REDDE has been shown to be very effective on some testbeds. However, as we show in the next sections, it produces poor results for some other testbeds.

Si and Callan [2004] presented their Unified Utility Maximization (UUM) framework for collection selection. UUM performs slightly better than REDDE on some testbeds. However, it uses training data and logistic models that require human relevance judgments.

We use CORI and REDDE as the benchmarks of our experiments because they are the two well-known algorithms that do not require training data for collection selection, and they both can work on uncooperative environments. In the next section, we introduce a novel approach for selecting collections in DIR systems.

3 Central-Rank-Based Collection Selection

In uncooperative environments, collection summaries usually consist of a limited number of documents downloaded by query-based sampling [Callan and Connell, 2001]. The query is compared to each of these summaries and collections are selected based on the similarity of their summaries with the query [Callan et al., 1995] or according to the estimated number of relevant documents they contain [Si and Callan, 2003a]. The ranking of sampled documents contains useful information that could be applied for selecting suitable collections. The set of all collection summaries together approximates a global index of documents in all collections and the ranking of summary documents could be seen as the result of a query against this index.

We propose *central-rank-based collection selection* (CRCS) as a new method that ranks collections according to the ranking of their summary documents and show that it can effectively select suitable collections.

In CRCS, an effective retrieval model is applied to the index of all sampled documents from collections. We refer to this model as the *central sample search engine* (CSSE). For each query, CSSE ranks the downloaded documents from collections. Then, the weight of each collection is calculated based on the ranks of its sampled documents that are in the top γ results. We have arbitrarily set γ to 50 for our experiments. How to identify an optimum value for this number is left as future work. The documents ranked after γ are less likely to be relevant and should have no impact on collection weights.

The top γ documents are ranked according to their probabilities of relevance. Intuitively, the weight of a collection with a sampled document at rank one should be incremented more than another collection with a sampled document at rank say 40.

The impact of a sampled document D on the weight of its original collection c is computed according to the position of that document in the top γ results. In the simplest form, this can be computed linearly as below:

$$R(D_j) = \begin{cases} \gamma - j & \text{if } j < \gamma \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $R(D_j)$ represents the impact of document D at the jth rank of results returned by CSSE. The impact of documents decreases linearly according to their ranks. However, previous studies [Joachims et al., 2005; Manmatha et al., 2001]

suggest that the importance of documents for users and their probabilities of relevance have a negative exponential relation with the document ranks. Therefore, it might be more suitable to assign document scores in a negative exponential manner as follows:

$$R(D_j) = \alpha \exp(-\beta \times j) \qquad if\ D_j \in S_i \tag{2}$$

Here, j is the rank of document D. Coefficient parameters α and β are two constants respectively set to 1.2 and 2.8 in our experiments according to the suggested figures by Joachims et al. [2005]. We use CRCS(l) when the impact values are computed linearly and CRCS(e) when Eq. (2) is applied.

The size of a collection is also important for calculating its final weight. If two collections contribute the same number of documents in the top-ranked results, the larger collection is likely to contain more relevant documents due to its greater size. Collection size is an important factor that has to be considered during collection selection. KL-divergence [Si et al., 2002], REDDE [Si and Callan, 2003a] and UUM [Si and Callan, 2004] all consider a collection size parameter in their calculations. Since the exact values for collection sizes are not usually available in uncooperative environments, we recommend that the size of each collection be estimated using the capture-history method [Shokouhi et al., 2006b].

In our experiments, we assume that the size of all collection summaries is the same (300 documents). In practice, collection summaries might be different in size. Larger summaries are more likely to contain the query terms, which means that collections with larger summaries are more likely to be selected for any given query. To overcome this bias, CRCS divides the weight of each collection by the size of its representation set.

Putting this together, CRCS calculates the weight of each collection as below:

$$C_i = \frac{CH_i}{CH_{max} \times |S_i|} \times \sum_{D \in S_i} R(D_j) \tag{3}$$

where, CH_i is the size of collection i estimated by the capture-history method [Shokouhi et al., 2006b]. We normalize the collection sizes by dividing the size of each collection by the size of the largest collection involved (CH_{max}). $|S_i|$ is the size of the representation set for collection i that is the number of documents downloaded by query-based sampling [Callan and Connell, 2001] from that collection. The weight of each collection is calculated by summing up the impact values for its summary documents. In summary, CRCS computes the final weights of collections as below:

- CSSE runs the query and ranks the sampled documents on the broker.
- The top γ documents returned by CSSE are selected and their impact values on the weights of their corresponding collections are calculated
- Collections are ranked according to the impact values of their sampled documents and their estimated sizes.

A similar technique is suggested by Craswell et al. [2000] for collection selection. In their approach, the broker sends a number of training multi-term probe

queries to collections. The top results from each collection are downloaded and gathered together in a single index. Then, the broker applies an effective retrieval model to rank the downloaded documents for the initial training queries. The search effectiveness of collection are computed according to their contribution to the top n (they suggested $n = 20$) results when the query is executed on the downloaded documents. Our approach is different to their technique in several ways; they calculate an effectiveness score for each collection according to its performance for the training queries. The final weight of a collection is computed by adding its effectiveness factor to the score calculated by a standard collection selection algorithms such as CORI. Unlike our approach, their suggested technique has not been used independently without relying on other collection selection methods. Also, while CRCS can select effective collections online, the suggested method by [Craswell et al., 2000] cannot capture this without a sufficient number of offline training queries.

REDDE [Si and Callan, 2003a] also uses a similar strategy for collection selection. However, it ignores the rank difference of documents in the central sample index and concentrates on selecting collections with the highest number of relevant documents (high recall). In contrast, our novel CRCS method focuses on selecting collections with high-quality documents (high precision).

In the following sections we compare the performance of CRCS with the other state-of-the-art methods on different testbeds.

4 Experimental Testbeds and Evaluation Metrics

Several testbeds have been developed for DIR experiments. These testbeds may be useful for evaluating DIR methods for specific applications such as enterprise search. However, most of them are not suitable for evaluating DIR techniques on the web because:

- The proposed testbeds are usually much smaller than the web collections and cannot be considered as good instances of the current web servers. The largest testbed reported so far is about 18 GB used by Hawking and Thomas [2005]. The other common testbeds are at most one sixth of this size[Powell and French, 2003: Si and Callan, 2003b:a].
- Collections are generated artificially by allocating each document to a collection according to a predefined criteria. This might be the author name or the year of publication[Powell and French, 2003; Si and Callan, 2003b;a].
- Collections usually only contain documents from the TREC newswire data. Considering the diversity of topics on the web, testbeds containing only news-related documents are not sufficient for unbiased evaluations.

In addition to the testbed deficiencies, there is a common defect with related work in this area; proposed algorithms are usually only tested on a few testbeds and their robustness has not been evaluated on different test data. In Section 5, we show that the performance of DIR methods vary substantially on different testbeds. This is consistent with previous work by D'Souza et al.

Table 1. Testbed statistics

		Number of Documents ×1000			Size (MB)		
Testbed	Size (GB)	Min	Avg	Max	Min	Avg	Max
trec123-100col-bysource	3.2	0.7	10.8	39.7	28	32	42
trec4-kmeans	2.0	0.3	5.7	82.7	4	20	249
100-col-GOV2	110.0	32.6	155.0	717.3	105	1126	3891

[2004a] that investigates the impact of testbed characteristics on collection selection algorithms. An algorithm that produces the best results on one dataset does not necessarily perform as well on another. We also introduce a new testbed based on the TREC GOV2 data. Our new testbed is more than six times larger than the largest DIR testbed reported so far; it is about 36 times larger than trec123-100col-bysource [Powell and French, 2003; Si and Callan, 2003b;a; 2004; Si et al., 2002], and 55 times larger than trec4-kmeans [Si and Callan, 2003b; Xu and Croft, 1999]. We test our method on six testbeds, so that we can fairly compare its performance and robustness with other available approaches. Table 1 includes information about the three major testbeds that have been used in our experiments. The other three testbeds are all generated from trec123-100col-bysource, thus we do not present them in the table.

- **trec4-kmeans (trec4)**: One hundred collections created from the TREC4 data. A k-means clustering algorithm have been used to organize the collections by topic [Xu and Croft, 1999]. TREC topics 201–250 (description) and their relevance judgments are used for performance evaluations. Collections in this testbed are small and the average query length is 7.2 words.
- **trec123-100col-bysource (uniform)**: One hundred collections are created by organizing the documents from the TREC disks 1, 2, and 3 by source and publication date. It is one of the most popular DIR testbeds and has been used in many papers [Powell and French, 2003; Si and Callan, 2003b;a; 2004; Si et al., 2002]. TREC topics 51–100 (title) are used as queries with the average length of 3 words per query.
- **100-col-GOV2 testbed (GOV2)**: In this new testbed, documents from the largest 100 servers in the TREC GOV2 data—in terms of the number of crawled pages—have been extracted and located in one hundred separate collections. TREC topics 701–750 (title) were used as queries. On average there are 3.2 words per query. The documents in all collections are from crawled webpages and the size of this testbed is many times larger than the current alternatives.

The other three testbeds are generated artificially from the uniform testbed [Powell and French, 2003; Si and Callan, 2003a; 2004].

- **trec123-AP-WSJ-60col (relevant)**: 24 Associate Press collections in the uniform testbed are collapsed into a single large APress collection. The same process is applied to 16 Wall Street Journal collections and they create a large

WSJournal collection. The other collections remain unchanged. Two large collections have higher density of relevant documents for the TREC queries.
- **trec123-2ldb-60col (representative)**: Collections in the uniform testbed are sorted by their name. Every fifth collection starting with the first is merged into a large "representative" collection. The same process is applied to every fifth collection starting from the second collection and they form another large representative collection. The other collections are unchanged.
- **trec123-FR-DOE-81col (nonrelevant)**: The 13 Federal Register and 6 Department of Energy collections in the uniform testbed are collapsed in two large collections respectively called FR and DOE. The rest of collections remain as were before. The larger collections have a lower density of relevant documents.

Collection selection algorithms are often compared using a recall metric R_k [Powell and French, 2003; Si and Callan, 2003a; 2004]:

$$R_k = \frac{\sum_{i=1}^{k} E_i}{\sum_{i=1}^{k} B_i} \tag{4}$$

Here, E is the collection selection ranking (CORI, REDDE, CRCS). B is the baseline ranking that is the relevance based ranking (RBR) [Powell and French, 2003] in our experiments. E_i and B_i are respectively the number of relevant documents in the ith ranked collection of E and B.

Choosing collections with a greater number of relevant documents (high recall) does not always lead to higher search effectiveness in DIR experiments [Si and Callan, 2003a; 2004]. Therefore, we also assess the effectiveness of algorithms on each testbed using relevance judgments. In all testbeds, we download 300 documents by query-based sampling[Callan and Connell, 2001] for each collection. Although it has been argued that using static summary sizes for collections is not always recommended [Baillie et al., 2006; Shokouhi et al., 2006a], we use this number to make our results comparable to other published work in this area [Callan and Connell, 2001; Craswell et al., 2000; Si and Callan, 2003a; 2004]. Each collection returns at most 100 answers to the broker for the entered query. We used SSL [Si and Callan, 2003b] algorithm to merge the returned results from the selected collections. The next section discusses the experimental results.

5 Results

Figure 1 depicts the performance of different collection selection algorithms on the trec4 and uniform testbeds. The horizontal axis in these figures shows the cutoff values, which are the number of collections that are selected for each query. The goal of collection selection methods is to select a few collections that contain the best answers. Therefore, we only show the R_k values for cutoffs smaller than 20. This number is consistent with DIR experiments that are reported elsewhere [Si and Callan, 2003a; 2004]. On the trec4 testbed, all methods produce almost

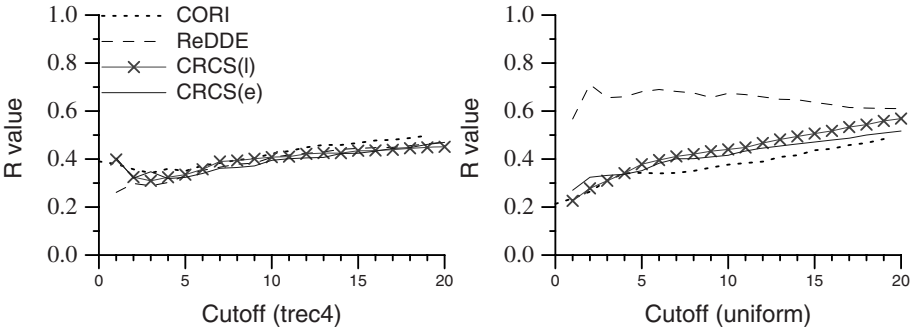

Fig. 1. R values for the CORI, REDDE and CRCS algorithms on the trec4-kmeans (left) and trec123-100col-bysource (right) testbeds

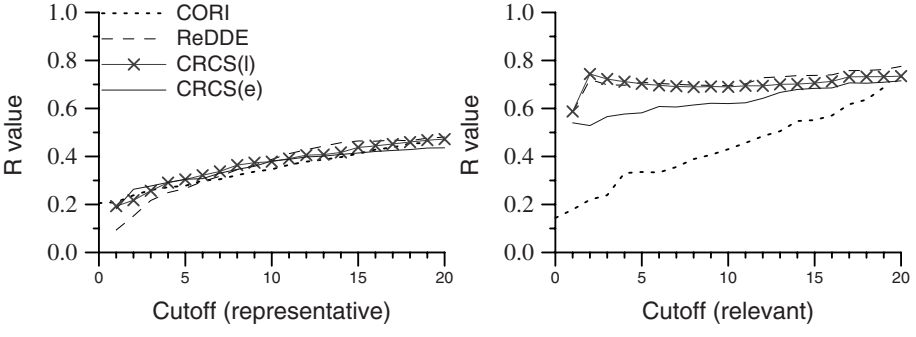

Fig. 2. R values for the CORI, REDDE and CRCS algorithms on the trec123-2ldb-60col (left) and trec123-AP-WSJ-60col (right) testbeds

the same R_k values for different cutoff points. For the uniform testbed, however, REDDE has a clear advantage while other methods have similar performance. We show later in this section that the advantage in recall does not have any significant impact on the final search effectiveness.

The R_k values for the representative and relevant testbeds are illustrated in Fig. 2. On the representative testbed, the difference between methods is negligible. On the relevant testbed, CORI produces far worse results than the other approaches. REDDE and CRCS(l) show similar outputs and they both work better than CRCS(e) for smaller cutoff points.

Figure 3 shows the R_k values produced by different methods on the GOV2 and non-relevant testbeds. The difference between methods on the non-relevant dataset is negligible for all cutoff values. On the GOV2 testbed, CORI selects collections with more relevant documents for smaller cutoff points. For larger cutoffs, all methods show similar outputs.

Higher recall values in the collection selection stage do not always lead to high precision in the final results [Si and Callan, 2003a; 2004]. Therefore, we

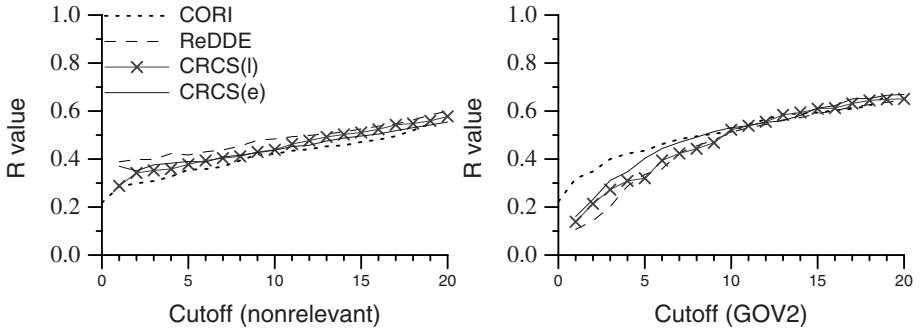

Fig. 3. R values for the CORI, REDDE and CRCS algorithms on the trec123-FR-DOE-81col (left) and 100-col-GOV2 (right) testbeds

Table 2. Performance of different methods for the Trec4 (trec4-kmeans) testbed. TREC topics 201–250 (long) were used as queries.

	Cutoff=1				Cutoff=5			
	P@5	**P@10**	**P@15**	**P@20**	**P@5**	**P@10**	**P@15**	**P@20**
CORI	0.3000	0.2380	0.2133†	0.1910†	0.3480	0.2980	0.2587	0.2380
ReDDE	0.2160	0.1620	0.1373	0.1210	0.3480	0.2860	0.2467	0.2190
CRCS(l)	0.2960	0.2260	0.2013	0.1810†	0.3520	0.2920	0.2533	0.2310
CRCS(e)	0.3080	0.2400	0.2173†	0.1910†	0.3880	0.3160	0.2680	0.2510

evaluate the search effectiveness of algorithms using the TREC queries and relevance judgments. We only report the results for cutoff=1 and cutoff=5. The former shows the system outputs when only the best collection is selected while for the latter, the best five collections are chosen to get searched for the query. We do not report the results for larger cutoff values because cutoff=5 has shown to be a reasonable threshold for DIR experiments on the real web collections [Avrahami et al., 2006]. The $P@x$ values show the calculated precision on the top x results.

We select REDDE as the baseline as it does not require training queries and its effectiveness is found to be higher than CORI and older alternatives [Si and callan, 2003a]. The following tables compare the performance of discussed methods on different testbeds. We used the t-test to calculate the statistical significance of difference between approaches. For each table, † and ‡ respectively indicate significant difference at the 99% and 99.9% confidence intervals between the performance of REDDE and other methods.

Results in Table 2 show that on the trec4 testbed, methods produce similar precision values when five collections are selected per query. The numbers also suggest that REDDE is not successful in selecting the best collection. It produces poorer results than the other methods and the difference is usually significant for P@15 and P@20.

Table 3. Performance of collection selection methods for the uniform (trec123-100col-bysource) testbed. TREC topics 51–100 (short) were used as queries.

	Cutoff=1				Cutoff=5			
	P@5	**P@10**	**P@15**	**P@20**	**P@5**	**P@10**	**P@15**	**P@20**
CORI	0.2520	0.2140	0.1960	0.1710	0.3080	0.3060	0.2867	0.2730
ReDDE	0.1920	0.1660	0.1413	0.1280	0.2960	0.2820	0.2653	0.2510
CRCS(l)	0.2120	0.1760	0.1520	0.1330	0.3440	0.3240	0.3067	0.2860
CRCS(e)	0.3800‡	0.3060‡	0.2613‡	0.2260†	0.3960	0.3700†	0.3480†	0.3310†

Table 4. Performance of collection selection methods for the representative (trec123-2ldb-60col) testbed. TREC topics 51–100 (short) were used as queries.

	Cutoff=1				Cutoff=5			
	P@5	**P@10**	**P@15**	**P@20**	**P@5**	**P@10**	**P@15**	**P@20**
CORI	*0.2160*	*0.2040*	*0.1773*	*0.1730*	0.3520	0.3500	0.3347	0.3070
ReDDE	0.3320	0.3080	0.2960	0.2850	0.3480	0.3220	0.3147	0.3010
CRCS(l)	0.3160	0.2980	0.2867	0.2740	0.3160	0.3160	0.2973	0.2810
CRCS(e)	0.2960	0.2760	0.2467	0.2340	0.3400	0.3500	0.3333	0.3090

On the uniform testbed (Table 3), CRCS(e) significantly outperforms the other alternatives for both cutoff values. CORI, REDDE, and CRCS(l) show similar performance on this testbed. Comparing the results with the R_k values in Figure 1 confirms our previous statement that selecting collections with a high number of relevant documents does not necessarily lead to an effective retrieval.

For the representative testbed as shown in Table 4, there is no significant difference between methods for cutoff=5. CRCS(l), CRCS(e) and REDDE produce comparable performance when only the best collection is selected. The precision values for CORI when cutoff=1 are shown in italic to indicate that they are significantly worse than REDDE at the 99% confidence interval.

On the relevant testbed (Table 5), all precision values for CORI are significantly inferior to that of REDDE for both cutoff values. REDDE in general produces higher precision values than CRCS methods. However, none of the gaps are detected statistically significant by the t-test at the 99% confidence interval.

Table 5. Performance of collection selection methods for the relevant (trec123-AP-WSJ-60col) testbed. TREC topics 51–100 (short) were used as queries.

	Cutoff=1				Cutoff=5			
	P@5	**P@10**	**P@15**	**P@20**	**P@5**	**P@10**	**P@15**	**P@20**
CORI	*0.1440*	*0.1280*	*0.1160*	*0.1090*	*0.2440*	*0.2340*	*0.2333*	*0.2210*
ReDDE	0.3960	0.3660	0.3360	0.3270	0.3920	0.3900	0.3640	0.3490
CRCS(l)	0.3840	0.3580	0.3293	0.3120	0.3800	0.3640	0.3467	0.3250
CRCS(e)	0.3080	0.2860	0.2813	0.2680	0.3480	0.3420	0.3280	0.3170

Table 6. Performance of collection selection methods for the non-relevant (trec123-FR-DOE-81col) testbed. TREC topics 51–100 (short) were used as queries.

	Cutoff=1				Cutoff=5			
	P@5	**P@10**	**P@15**	**P@20**	**P@5**	**P@10**	**P@15**	**P@20**
CORI	0.2520	0.2100	0.1867	0.1690	0.3200	0.2980	0.2707	0.2670
ReDDE	0.2240	0.1900	0.1813	0.1750	0.3480	0.2980	0.2707	0.2610
CRCS(l)	0.2040	0.1860	0.1813	0.1800	0.3360	0.3220	0.2973	0.2860
CRCS(e)	0.3200†	0.2820†	0.2533†	0.2240	0.3880	0.3600	0.3387†	0.3210†

Table 7. Performance of collection selection methods for the GOV2 (100-col-GOV2) testbed. TREC topics 701–750 (short) were used as queries.

	Cutoff=1				Cutoff=5			
	P@5	**P@10**	**P@15**	**P@20**	**P@5**	**P@10**	**P@15**	**P@20**
CORI	0.1592†	0.1347†	0.1143†	0.0969†	0.2735	0.2347	0.2041	0.1827
ReDDE	0.0490	0.0327	0.0286	0.0235	0.2163	0.1837	0.1687	0.1551
CRCS(l)	0.0980	0.0755	0.0667	0.0531	0.1959	0.1510	0.1442	0.1286
CRCS(e)	0.0857	0.0714	0.0748	0.0643	0.2776	0.2469	0.2272	0.2122

The results for CRCS(l), REDDE and CORI are comparable on the non-relevant testbed (Table 6). CRCS(e) significantly outperforms the other methods in most cases. On the GOV2 testbed (Table 7), CORI produces the best results when cutoff=1 while in the other scenarios there is no significant difference between the methods.

Overall, we can conclude that CRCS(e) selects better collections and its high performance remains robust. In none of the reported experiments, the precision values for CRCS(e) were significantly poorer than any other method at the 99% confidence interval. However, in many cases, the performance of CRCS(e) was significantly better than the second best method at the 99% or 99.9% confidence intervals. CORI and REDDE showed variable performance on different testbeds, each outperforming the other on some datasets.

6 Conclusions

We have introduced a new collection selection method for uncooperative DIR environments. We have shown that our proposed CRCS method can outperform the current state-of-the-art techniques. We investigated the robustness of different collection selection algorithms and showed that the performance of REDDE and CORI changes significantly on different testbeds while CRCS produces robust results. We also introduced a new testbed for DIR experiments based on the TREC GOV2 dataset. Our proposed testbed is about 36 times larger than the most well-known DIR testbeds and more than 6 times larger than the largest DIR testbed ever reported. Moreover, unlike traditional DIR testbeds that docu-

ments are assigned artificially into collections, collections in this testbed contain downloaded web documents arranged by server.

Experiments reported in this paper are based on the assumption that all collections are using the same retrieval model with equal effectiveness. However, in practice, collections often use different retrieval models and have different effectiveness. We plan to extend our experiments on collections with different retrieval models. Finally, the most proper values for α, β and γ have not been investigated and will be explored in our future research.

Acknowledgment

I am grateful to Justin Zobel, for his valuable comments on this work.

References

T. Avrahami, L. Yau, Luo Si, and Jamie Callan. The FedLemur: federated search in the real world. *Journal of the American Society for Information Science and Technology*, 57(3):347–358, 2006.

M. Baillie, L. Azzopardi, and F. Crestani. Adaptive query-based sampling of distributed collections. In *SPIRE String Processing and Information Retrieval Symposium*, pages 316–328, Glasgow, UK, 2006.

J. Callan and M. Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems*, 19(2):97–130, 2001.

J. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *Proc. ACM SIGIR Conf.*, pages 21–28, Seattle, Washington, 1995.

N. Craswell, P. Bailey, and D. Hawking. Server selection on the World Wide Web. In *Proc. ACM Conf. on Digital Libraries*, pages 37–46, San Antonio, Texas, 2000.

D. D'Souza, J. Thom, and J. Zobel. Collection selection for managed distributed document databases. *Information Processing and Management*, 40(3):527–546, 2004a.

D. D'Souza, J. Zobel, and J. Thom. Is CORI effective for collection selection? an exploration of parameters, queries, and data. In *Proc. Australian Document Computing Symposium*, pages 41–46, Melbourne, Australia, 2004b.

L. Gravano, C. K. Chang, H. Garcia-Molina, and A. Paepcke. STARTS: Stanford proposal for Internet meta-searching. In *Proc. ACM SIGMOD Conf.*, pages 207–218, Tucson, Arizona, 1997.

L. Gravano, H. Garcia-Molina, and A. Tomasic. GlOSS: text-source discovery over the Internet. *ACM Transactions on Database Systems*, 24(2):229–264, 1999.

D. Hawking and P. Thomas. Server selection methods in hybrid portal search. In *Proc. ACM SIGIR Conf.*, pages 75–82, Salvador, Brazil, 2005.

T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proc. ACM SIGIR Conf.*, pages 154–161, Salvador, Brazil, 2005.

R. Manmatha, T. Rath, and F. Feng. Modeling score distributions for combining the outputs of search engines. In *Proc. ACM SIGIR Conf.*, pages 267–275, New Orleans, Louisiana, 2001.

H. Nottelmann and N. Fuhr. Evaluating different methods of estimating retrieval quality for resource selection. In *Proc. ACM SIGIR Conf.*, pages 290–297, Toronto, Canada, 2003.

A. L. Powell and J. French. Comparing the performance of collection selection algorithms. *ACM Transactions on Information Systems*, 21(4):412–456, 2003.

S. Raghavan and H. Garcia-Molina. Crawling the hidden web. In *Proc. 27th Int. Conf. on Very Large Data Bases*, pages 129–138, Roma, Italy, 2001. Morgan Kaufmann Publishers Inc.

M. Shokouhi, F. Scholer, and J. Zobel. Sample sizes for query probing in uncooperative distributed information retrieval. In *Proc. Asia Pacific Web Conf.*, pages 63–75, Harbin, China, 2006a.

M. Shokouhi, J. Zobel, F. Scholer, and S.M.M. Tahaghoghi. Capturing collection size for distributed non-cooperative retrieval. In *Proc. ACM SIGIR Conf.*, pages 316–323, Seattle, Washington, 2006b.

L. Si and J. Callan. Unified utility maximization framework for resource selection. In *Proc. ACM CIKM Conf.*, pages 32–41, Washington, 2004.

L. Si and J. Callan. Relevant document distribution estimation method for resource selection. In *Proc. ACM SIGIR Conf.*, pages 298–305, Toronto, Canada, 2003a.

L. Si and J. Callan. A semisupervised learning method to merge search engine results. *ACM Transactions on Information Systems*, 21(4):457–491, 2003b.

L. Si, R. Jin, J. Callan, and P. Ogilvie. A language modeling framework for resource selection and results merging. In *Proc. ACM CIKM Conf.*, pages 391–397, McLean, Virginia, 2002.

J. Xu and B. Croft. Cluster-based language models for distributed retrieval. In *Proc. ACM SIGIR Conf.*, pages 254–261, Berkeley, California, United States, 1999.

B. Yuwono and D. L. Lee. Server ranking for distributed text retrieval systems on the Internet. In *Proc. Conf. on Database Systems for Advanced Applications*, pages 41–50, Melbourne, Australia, 1997.