Approaches to Collection Selection and Results Merging for Distributed Information Retrieval

Yves Rasolofo
Institut interfacultaire d'informatique,
Université de Neuchâtel
Pierre à Mazel, 7
2000 Neuchâtel, Switzerland
Yves.Rasolofo@unine.ch

Faïza Abbaci
Ecole nationale supérieure des Mines de Saint-Étienne
158 Cours Fauriel
42023 Saint-Étienne, France
Abbaci@emse.fr

Jacques Savoy
Institut interfacultaire d'informatique,
Université de Neuchâtel
Pierre à Mazel, 7
2000 Neuchâtel, Switzerland
Jacques.Savoy@unine.ch

ABSTRACT

We have investigated two major issues in Distributed Information Retrieval (DIR), namely: collection selection and search results merging. While most published works on these two issues are based on pre-stored metadata, the approaches described in this paper involve extracting the required information at the time the query is processed. In order to predict the relevance of collections to a given query, we analyse a limited number of full documents (e.g., the top five documents) retrieved from each collection and then consider term proximity within them. On the other hand, our merging technique is rather simple since input only requires document scores and lengths of results lists. Our experiments evaluate the retrieval effectiveness of these approaches and compare them with centralised indexing and various other DIR techniques (e.g., CORI [2][3][23]).

We conducted our experiments using two testbeds: one containing news articles extracted from four different sources (2 GB) and another containing 10 GB of Web pages. Our evaluations demonstrate that the retrieval effectiveness of our simple approaches is worth considering.

Keywords

Distributed information retrieval, collection selection, search results merging, evaluation.

1. INTRODUCTION

Conventional retrieval systems based on a single centralised index are subject to several limitations [11]. Among these is the very critical limitation due to the exponential growth in available information on the Internet. Even though search engines have currently increased their coverage of the content of the Web, They are still a long way from full coverage of the entire Web.

Moreover, other limitations inherent to centralised approach might surface such as insufficient bandwidth server overloading and failures. Given these facts, it thus seems more appropriate to turn to the DIR approach for storage and search processing.

A simple DIR system is made up of collection servers and a broker. Typically, a user submits a request to the broker, which then forwards the query to a carefully selected subset of collection servers likely to contain relevant documents to the query (e.g., based on query terms, query language or a subset of servers pre-selected by the user). Each selected collection server processes the query and returns a ranked list to the broker. Finally, the broker merges the results lists received into a single list and forwards it to the user.

Some recent studies have reported that retrieval effectiveness in DIR systems may potentially be as effective as a single centralised IR system [17][24]. In our research, we have investigated how to select a subset of collection servers which are most likely to be relevant to a given query, and then how to merge the results lists in order to obtain improved retrieval effectiveness.

In this vein, the collection selection system we propose is different in two main aspects from various other selection methods. Firstly, our approach does not use any pre-stored metadata to predict the collection's relevance to the query. Secondly, it does not require collection ranking: each collection is selected independently from the others. On the other hand, our merging technique involves a rather simple recalculation of each document score.

We carried out our evaluations using a first testbed made up of news articles (about 2 GB from TREC8) and a second testbed containing pages extracted from the Web (more than 10 GB from TREC9).

In Sections 2 and 3, we will describe some well-known techniques for collection selection and results merging, and then describe our approaches in more details. Section 4 will discuss the details of evaluations we carried out on two testbeds (TREC8 and TREC9), and comparisons between performances of our strategies with those of other approaches. The final section will comment on experimental results and on our work in progress.

2. COLLECTION SELECTION

Collection selection refers to automatic or manual selection of a subset of collections (or servers) most likely to contain relevant documents for a given query. Obviously, ignoring this step and

sending the query to all known collections is one possible solution, but this method is proving to be very expensive in terms of resources, and it can increase user latency. Thus, the goal of collection selection is to reduce the number of collections searched as much as possible, without decreasing retrieval effectiveness [9][11].

2.1 Related Works

In order to select automatically a subset of servers, most collection selection techniques compute a score for each of the collections, based on their usefulness to the submitted query. The collections are then ranked according to these scores, thus the system could select either the N highest ranking collections or collections with scores exceeding a certain threshold. This type of selection requires the global collection of information in order to calculate the collection scores. Thus, the approaches differ in the nature of this information or in the manner in which it is acquired. Previous works consider collection descriptions [4] or collection statistics (frequency, co-occurrence, etc.) [10][16][25] in order to perform collection selection, but these techniques require collection cooperation.

Xu et al. [24] suggested that documents could be gathered according to their topics, and a language model associated with each topic. Callan et al. [2][3][23] have presented a Collection Retrieval Inference network (CORI), which considers each collection as a single gigantic document. Ranking the collections is similar to document ranking methods used in conventional information retrieval system. Several methods were developed by Zobel [25] to calculate collection scores. Moffat et al. [16] suggested to decompose each collection into blocks of documents, with the blocks being indexed by the broker. This index is then used to find those blocks having high-ranking scores with respect to the query, and the collections matching these blocks are then selected. The GlOSS system [10] ranks collections according to their goodness for the submitted query, and to do so it estimates the number of documents in each collection having similarities to the query greater than a predetined threshold. It then sums these similarities in order to obtain the collection score. At query time, Hawking et al. [1 1] proposed to broadcast a probe query (containing one to three terms) to all available collections, each of which responds with term information that is used to calculate collection score. Towel1 et al. [21] developed a learning based method for reducing search costs, whereby they determined the optimum number of documents to be retrieved from each collection rather than defining the number of collections to be searched. In their calculation of collection scores Craswell et al. [7] included the search engine's retrieval effectiveness for each collection.

We believe CORI to be a good representative of the above strategies, we will describe its selection procedure in more details and evaluate it in our experiments. This approach uses an inference network to rank collections. For the ith collection and for a given query Q, the collection score is computed as:

$$s_i = \frac{1}{m}\sum_{j=1}^{m} s(t_j \quad C_i)$$

Where $s(t_j / C_i)$ indicates the contribution of the search term t_j in the score of collection C_i calculated as follows:

$$s(t_j \mid C_i) = defB + (1 - defB) \cdot \frac{df_i}{df_i + K} \cdot \frac{\log\left(\frac{|C| + 0.5}{cf_j}\right)}{\log(|C| + 1.0)},$$

where:

$$K = k \cdot \left[(1-b) + b \cdot \frac{lc_i}{avlc}\right],$$

m is the number of terms included in Q,

$|C|$ is the number of collections,

df_i is the number of documents in collection C_i containing the jth query term,

cf_j is the number of collections containing the query term t_j,

lc_i is the number of indexing terms in C_i ,

$avlc$ is the average number of indexing terms in each collection, and

$defB$, b and k are constants and, as suggested by Xu and Callan [23], were set at the following values: $defB$ = 0.4, k = 200 and b = 0.75.

After ranking the collections according to their scores, one possibility is to select the N top ranked collections, where N is determined by the user. Another possibility is to use an algorithm to cluster the collection scores and the collections in the top clusters are then selected [2]. We have evaluated the latter case using a cluster difference threshold α=0.0002.

2.2 Our Selection Procedure

We have denoted our selection approach as TRD-CS, for 'using Top Ranked Documents for Collection Selection' and it differs from previous ones in that it does not assign scores to each collection. It bears a slight resemblance to the approach developed by Hawking et al. [11], both approaches assuming that no information is available a ***priori*** to perform the selection. The information needed is derived while processing the query.

In our approach, the broker broadcasts the query to all available collections ($|C|$ collections), with each collection returning nb_doc highly ranked documents to the broker. The broker then calculates the score for each document received ($nb\_doc * |C|$), and sorts them according to their scores, with the collections matching the n_first documents being selected.

In order to compute document scores, we assume that the following are good relevance indicators: the number of query terms included in each document surrogate, the distance between query terms and their frequencies. From this perspective and inspired by [14], we calculate the document score as follows:

$$score(D,Q) = (c_1 \cdot nb_q) + \left(c_2 \cdot dis\_ind(D,Q)\right) + \frac{nb\_occ}{c_3}$$

where for each document D:

nb_q is the number of query terms present in D,

nb_occ is the total number of occurrences of query terms in D,

c_1 , c_2 , c_3 are constants and set to c_1 =lOO, c_2 =lOOO, c_3 =lOOO in our experiments,

dis-ind is the indicator of distance between query terms in *D*. This function returns a real value greater or equal to zero.

According to the formula introduced by Clarke et al. [5] and assuming the two first query terms are the most important search keywords, we compute *dis-ind* for these two terms as follows.

$$dis\_ind(D, Q) = \sum dis(k,l)_i$$

where:

k and *l* are query term positions within the document *D* delimiting the ith block,

$dis(k,l)_i$ is the score for this block in the document *D*, which satisfies the query Q (i.e. the block contains the first two query terms in our case), and which does not include any other block satisfying the query (the block having the smallest size is selected). For example, we consider a given query consisting of two terms t_i and t_j. If t_i appears in the 5th and 25" positions and t_j in the 27" position, we may find the first block ($k=5$ and $l=27$) and the second block ($k=25$ and $l=27$). As the first block contains the second, this first block is ignored and *dist_ind* is therefore reduced to $dist\_ind(D,Q) = dis(25, 27)=0.5$.

More formally:

$$dis(k,l)_i = \begin{cases} \dfrac{1}{|(k,l)_i|}, & if\ |(k,l)_i| > 1 \\ 1, & if\ |(k,l)_i| \le 1 \end{cases}$$

In the case of a mono-term query, and according to [14], *dis-ind* represents the inverse of the distance from the start of the document to the first occurrence of this search term. Finally, the document score is assigned a zero value if it does not contain any query terms.

3. RESULTS MERGING STRATEGIES

After defining the set of selected collections, results lists returned from collections must be combined into a final single ranked list. To resolve this problem, various merging strategies have been suggested.

3.1 Related Works

One of the best-known approaches assumes that each collection contains approximately the same number of relevant items and that they are equally distributed within results lists taken from the collections [22]. Under this assumption, we can set up a final result list in a round robin manner, a merging strategy used by earlier meta-search engines on the Web [19]. Our previous evaluation [20] demonstrated that this method did not perform well and thus is not included in this paper for further evaluation.

When collections are indexed by the same search model, we may assume that scores attributed to documents are comparable across collections [12]. The document scores are then used to merge the documents from collections into a single list. This strategy is called "Raw Score Merging" (RSM). However, Dumais [8] mentioned that various statistics may be collection dependant (e.g., the *idf* value used to weight documents and/or queries) and these values may vary widely across collections. Therefore, this phenomenon may invalidate the raw score merging hypothesis.

One variant of the RSM strategy is to assume that absolute document scores would not be compared. We could normalise the document scores based on the maximum document score for each collection.

The CORI [2][3][13][23] approach proposes a fourth merging strategy, and we will compare its performance with that of our merging approach. As shown previously, CORI computes a score for each collection (Section 2.1). Based on this collection score denoted as s_i, the weight w_i for ith collection is:

$$w_i = 1 + c \mid \cdot [(s_i - \bar{s})/\bar{s}]$$

where:

s_i is the collection's score of the *ith* collection,

$\bar{s}$ is the mean of collection scores, and

$|C|$ is the number of collections as defined above.

The resulting weight w_i will be used to modify the score attached to each document. Instead of using document scores directly (as in RSM), each document score is multiplied by the value w_i of the corresponding collection and the broker merges the results lists according to these new scores.

3.2 Our Merging Strategy

Our merging strategy is denoted LMS for 'using result Length to calculate Merging Score', and it calculates a score for each collection. The underlying idea is to use weights in order to increase document scores from collections having scores greater than the average score, and to decrease those from any collections having scores less than the average score. Our approach has the advantage of being simple, since as input it only uses document scores and result lengths. Since collection statistics are not required, a broker using our approach does not need to store collection information. By contrast, when collections statistics are required within a dynamic environment such as the Web, they need to be updated frequently, and this is not possible without establishing some cooperation between the broker and collection servers. Thus, our approach is more practical.

Our merging strategy consists of calculating a collection score according to the proportion of documents retrieved (result length) by each collection. This score is based on our intuition that a collection would contain more relevant documents for a given query, if its collection server found more documents. The score for the ith collection is determined by:

$$s_i = \log\left(1 + \frac{l_i \cdot K}{\sum_{j=1}^{|C|} l_j}\right)$$

where:

K is a constant (set to 600 in our evaluations),

l_i is the number of documents retrieved by the ith collection, and

$|C|$ is the number of collections.

Our model uses a constant K in order to normalize the collection score as well as the natural logarithm, an order-preserving transformation used in similar contexts [15]. Based on this collection score, our merging algorithm calculates the collection weight denoted w_i for the ith collection as follows:

$$w_i = 1 + [(s_i - \bar{s})/\bar{s}]$$

where:

s_i is the ith collection score, and

$\bar{s}$ is the mean collection score.

As in the CORI approach, the final document score is the product of w_i and the document score as computed by the server for this document.

4. EXPERIMENTS

4.1 The Testbeds

Our first testbed consisted of documents written in English and used for the eighth TREC conference, called TREC8. It contains 528,155 documents (about 1,904 MB) extracted from four sources: *Financial Times* (FT, 2 10,158 documents), *Federal Register* (FR, 55,630 documents), *Foreign Broadcast Information Service* (FBIS, 130,471 documents) and *Los Angeles Times* (LA Times, 13 1,896 documents).

Collection	Size (MB)	# docs	# Topic-Rel.	# Topic-Ret.
FT	564	210,158	49	50
FR	395	55,630	18	50
FBIS	470	130.471	43	50
LA Times	475	131,896	45	50
TREC8	1904	528,155	50	50

Table 1. TREC8 statistics

An assessed set of 50 topics was provided, covering a rather broad range of subjects, including for example "Estonia, economy," "suicides," "airport security," "osteoporosis" and "cosmic events." We used only topic titles having two words on average (standard deviation: 0.96) in order to simulate typical queries sent by search engine users. We noted that these query words are ambiguous and occurres frequently within the documents.

The second testbed is the test collection used for the Web track of TREC9 conference, which differs from TREC8 in that it contains only Web pages from sites around the world. (1,692,096 Web pages with a total size of 11,033 MB). Thus, it is roughly six times greater than TREC8 and its contents are quite different, including a larger number of spelling mistakes. As with the first testbed, we used only topic titles based on real-life queries sent by users to the Excite search engine. They originated from various fields (e.g., "Parkinson's disease," "hunger," "Baltimore," "how e-mail benefits businesses" and "Mexican food culture"). Query terms are also ambiguous, and their average length is 2.4 words (standard deviation of 0.6).

We decided to split the TREC8 testbed into four collections according to their sources (FT, FR, FBIS, LA Times), with the number of documents and the collection size varying from one collection to another (see Table 1). This configuration more closely reflects a digital library environment, made up of several information sources (or servers) and using the same search model. We divided the TREC9 testbed into eight collections, each having roughly the same number of documents and the same size (see Table 2).

Collection	Size (MB)	# docs	Topic-Rel.	# Topic-Ret.
TREC9.1	1,325	207,485	28	48
TREC9.2	1,474	207,429	44	48
TREC9.3	1,438	221,916	38	48
TREC9.4	1,316	202,049	21	48
TREC9.5	1,309	203,073	22	48
TFUK9.6	1,311	215,451	24	48
TREC9.7	1,336	200,146	25	47
TREC9.8	1,524	234,547	43	48
TREC9	11,033	1,692,096	50	48

Table 2. TREC9 statistics

Tables 1 and 2 contain various statistics including for each collection the size, number of documents, number of topics having at least one relevant item, and number of topics having at least one document returned. For some topics there were no relevant documents returned by the collections. For example in TREC9, query topics #464 and #487 did not return any documents from any of the eight collections, due to spelling errors (topic #464: "nativityscenes" and topic #487: "angioplast7").

Merging / Selection	Single (baseline) Av. Prec.	RSM Av. Prec.	RSM Diff.	CORI Av. Prec.	CORI Diff	LMS Av. Prec.	LMS Diff
TREC8-NS	0.2566	0.2397	-6.59 %	0.2416	-5.85 %	0.2462	-4.05 %
TREC8-CORI	0.2566	0.2005	-21.86 %	0.2033	-20.77 %	0.1997	-22.17 %
TREC8-TRD-CS	0.2566	0.2440	-4.91 %	0.2453	-4.40 %	0.2462	-4.05 %
TREC8-OPT	0.2566	0.2543	-0.90 %	0.2533	-1.29 %	0.2480	-3.35 %
TREC9-NS	0.1986	0.1832	-7.75 %	0.1847	-7.00 %	0.1932	-2.72 %
TREC9-CORI	0.1986	0.1862	-6.24 %	0.1893	-4.68 %	0.1922	-3.22 %
TREC9-TRD-CS	0.1986	0.1828	-7.96 %	0.1867	-5.99 %	0.1944	-2.11 %
TREC9-OPT	0.1986	0.2097	5.59 %	0.2142	7.85 %	0.2144	7.96 %

Table 3. Average precision achieved by various selection and merging strategies

Merging / Selection	RSM vs. CORI	RSM vs. LMS	CORI vs. LMS
TREC8-NS	RSM < CORI	RSM < LMS	CORI < LMS
TREC8-CORI	RSM = CORI	RSM = LMS	CORI = LMS
TRECI-TRD-CS	RSM = CORI	RSM = LMS	CORI = LMS
TREC8-OPT	RSM = CORI	RSM = LMS	CORI = LMS
TREC9-NS	RSM < CORI	RSM < LMS	CORI < LMS
TREC9-CORI	RSM = CORI	RSM < LMS	CORI < LMS
TREC9-TRD-CS	RSM < CORI	RSM < LMS	CORI < LMS
TREC9-OPT	RSM = CORI	RSM = LMS	CORI = LMS

Table 4. Results of Sign tests for various selection and merging strategies

Selection/Merging / Collection	NS/RSM vs. CORIKORI	NS/RSM vs. TRD-CS/LMS	CORIKORI vs. TRD-CS/LMS
TREC8	NS/RSM = CORI/CORI	NS/RSM < TRD-CS/LMS	CORI/CORI < TRD-CS/LMS
TREC9	NS/RSM = CORI/CORI	NS/RSM < TRD-CS/LMS	CORIKORI < TRD-CS/LMS

Table 5. Results of Sign tests for NS/RSM vs. CORI/CORI vs. TRD-CS/LMS

All collections were indexed by the SMART system [1], using the OKAPI [18] probabilistic search model (Appendix 1).

4.2 Our Baselines

In the evaluations below, we will refer to a number of baselines, defined as follows:

- the optimal collection selection: given knowledge about the complete set of relevant documents supplied with the TREC data, we select collections that have at least one relevant document (labelled 'TREC8-OPT' and 'TREC9-OPT' in our tables);
- the centralised approach: for each of the testbeds, all documents are located in a single database (labelled 'Single' in our tables);
- no selection (NS): all collections are searched (we do not apply any selection procedure in evaluations labelled 'TREC8-NS' and 'TREC9-NS').

4.3 Evaluation

Given the three collection selection approaches, namely TRD-CS, CORI using collection scores clustering, and optimal selection (OPT), and the three merging strategies LMS, CORI and RSM, our experiments combined each collection selection approach with each merging strategy. The results reported below were obtained by using the following parameter values:

Our selection approach (TRD-CS) used nb_doc = 5 (the number of top documents returned by each collection to be inspected); n_first = 11 for TREC8 testbeds and n_first = 22 for TREC9 testbed.

We used two different methodologies to evaluate the various approaches. Firstly for a given testbed and a combination of a selection approach with a merging approach, we used the TREC_EVAL program to compute average precision and precisions achieved after retrieving 5, 10, 15, 20, 30, 100, 200, 500 and 1000 documents. Table 3 shows the average precision values achieved by various selection/merging combinations and compares them with those of the centralised approach (column labelled 'Single'). Appendix 2 (Figures 1 to 6) depicts differences between precisions achieved by different combinations, according to the number of retrieved documents by means of curves.

Secondly, in order to decide whether a retrieval strategy was statistically better than another was, we used the Sign test [6], with a significance level α = 0.05. The decisions based on this test are reported in Tables 4 and 5.

4.4 Discussion

In Table 3, the baseline (labelled 'Single') represents the average precision achieved by the centralised approach. The optimal selection ('TREC8-OPT' and 'TREC9-OPT') can be viewed as an ideal selection'. This optimal selection procedure produced the best retrieval effectiveness, no matter which merging strategy was used. With the TREC9 corpus and for three merging strategies, this scheme even improved the retrieval effectiveness over the baseline. The second best selection procedure seems to be our approach ('TRECI-TRD-CS' and 'TREC9-TRD-CS'). Ignoring the selection procedure ('TREK&NS' and 'TREC9-NS') decreases the retrieval effectiveness from 6% to 8% over the centralised approach for RSM and CORI merging strategies. However, this degradation is lower when using our merging strategy LMS (4% for TREC8 and 3% for TREC9). Finally when combining our selection procedure (TDR-CS) with our merging strategy (LMS), it is evident that the achieved retrieval effectiveness is very close to that of the centralised approach for TREC9 corpus (-2.11%) and very satisfactory for TREC8 corpus *(-4.05%).*

[1] We performed such an ideal selection by using relevance judgement information. For a given query, a collection is selected only if it contains at least one relevant document.

In order to analyse the relative significance of differences found in Table 3, we applied the Sign tests. The results shown in Table 4 indicate that raw score merging (RSM) attains a retrieval performance that could be considered either equal or less effective than either of the CORI and our suggested LMS merging strategies. Also worth noting is that the last column indicates that our merging strategy (LMS) produces better or at least equal retrieval effectiveness compared to that of CORI.

Finally, it seems more appropriate to directly compare various combinations of good selection and good merging strategies. In this vein, we also compared NS/RSM, CORI/CORI and TRD-CS/LMS. The results as reported in Table 5 indicate that our solution outperforms both that of the CORI/CORI and NS/RSM strategies.

5. CONCLUSION

This paper describes effective selection collection and results merging strategies useful in DIR. Our approaches do not require the creation of any pre-stored metadata, and as such, do not need any up-dates to reflect changes in collection content. Moreover, our experiments were conducted using very short queries, similar to those submitted to search engines and therefore viewed as "Web realistic." Our evaluations show that:

- a combination of our two strategies works better than other collection-selection/results merging combinations,
- our selection method works well even with RSM or CORI merging strategies, and our merging approach works also well when no selection is performed,
- however, our selection strategy requires more transfer traffic for the downloading of the first *nb_doc* (*nb_doc=5* in our evaluations) documents per collection. Thus, response delay may increase slightly.

The investigation described in this paper used the same search engine on several collections, a context that corresponds to a digital library environment where all sources are managed by the same search engine. Our current work will also consider the use of several collections indexed and searched by different search engines, without requiring a learning phase as described in [15].

6. ACKNOWLEDGEMENTS

This material is based on work supported in part by SNSF (Swiss National Science Foundation, under grant #21-58 813.99, J. Savoy and Y. Rasolofo) and by Region Rhône-Alpes (Eurodoc grant from F. Abbaci).

7. REFERENCES

[1] Buckley C.: Implementation of SMART Information Retrieval System. Technical Report 85-686, Computer Science Department, Cornell University, Ithaca, May 1985.

[2] Callan J. P., Lu Z., Croft W. B.: Searching Distributed Collections with Inference Networks. Proceedings of the ACM-SIGIR'95, 1995, pp. 21-28.

[3] Callan J.: Distributed Information Retrieval. In W. B. Croft (Ed.), Advances in Information Retrieval. Kluwer Academic Publishers, 2000 pp. 127-150.

[4] Chakravarthy A. S., Haase K. B.: NetSerf: Using Semantic Knowledge to Find Internet Information Archives. Proceeding of the ACM-SIGIR'95, 1995, pp. 4-l 1.

[5] Clarke C. L. A., Cormack G. V., Burkowski F. J.: Shortest Substring Ranking (MultiText Experiments for TREC-4). Proceedings of TREC-4, 1995, pp. 295-304.

[6] Conover W.J.: Practical Nonparametric Statistics (2nd ed.). John Wiley & Sons, 1980, pp. 122-129.

[7] Craswell N., Bailey P., Hawking D.: Server Selection in the World Wide Web. Proceedings of The Fifth ACM Conference on Digital Libraries, 2000, pp. 37-46.

[8] Dumais S. T.: Latent Semantic Indexing (LSI) and TREC-2. Proceedings of TREC-2, 1994, pp. 105-l 15.

[9] French J. C., Powell A.L., Callan J., Viles C. L., Emmitt T., Prey K. J, Mou Y.: Comparing the Performance of Database Selection Algorithms. Proceedings of ACM-SIGIR'99, 1999, pp. 238-245.

[10] Gravano L., Garcia-Molina H., Tomasic A.: GlOSS: Text-Source Discovery Over the Internet. ACM Transactions on Database Systems, 24(2), 1999, pp. 229-264.

[11] Hawking D., Thistlewaite P.: Methods for Information Server Selection. ACM Transactions on Information Systems, 17(1), 1999, pp. 40-76.

[12] Kwok K. L., Gmnfeld L., Lewis D. D.: TREC-3 Ad-hoc, Routing Retrieval and Thresholding Experiments using PIRCS. Proceedings of TREC-3, 1995, pp. 247-255.

[13] Larkey L. S., Connell M. E., Callan J.: Collection Selection and Results Merging with Topically Organized U.S. Patents and TREC Data. Proceedings of CIKM'2000, 2000, pp. 282-289.

[14] Lawrence S., Giles C. L.: Inquirus, the NECI Meta Search Engine. Proceedings of The Seventh International World Wide Web Conference, 1998, pp. 95- 105

[15] Le Calve A., Savoy J. : Database Merging Strategy Based on Logistic Regression. Information Processing & Management, 36(3), 2000, pp. 341-359.

[16] Moffat A. , Zobel J.: Information Retrieval Systems for Large Document Collections. Proceedings of TREC-3, 1995, pp. 85-94.

[17] Powel A. L., French J. C., Callan J., Connell M., Viles C. L.: The Impact of Database Selection on Distributed Searching. Proceedings of the ACM-SIGIR-2000, 2000, pp. 232-239.

[18] Robertson S. E., Walker, S., Beaulieu M.: Experimentation as a Way of Life: Okapi at TREC. Information Processing & Management, 36(1), 2000, pp. 95-108.

[19] Selberg E.W.: Towards Comprehensive Web Search. Ph.D. Thesis, University of Washington, 1999.

[20] Savoy J., Rasolofo Y.: Report on TREC-9 Experiment: Linked-based Retrieval and Distributed Collections. Proceedings of TREC9, 2000, to appear.

[21] Towel1 G., Voorhees E. M., Narendra K. G., Johnson-Laird B.: Learning Collection Fusion Strategies for Information

Retrieval. Proceedings of The Twelfth Annual Machine Learning Conference, 1995, pp. 540-548.

[22] Voorhees E. M., Gupta N. K., Johnson-Laird B.: Learning Collection Fusion Strategies. Proceedings of the ACM-SIGIR'95, 1995, pp. 172-179.

[23] Xu J., Callan J. P.: Effective Retrieval with Distributed Collections. Proceedings of the ACM-SIGIRP8, 1998, pp. 112-120.

[24] Xu J., Croft, W. B.: Cluster-based Language Models for Distributed Retrieval. Proceedings of ACM-SIGIR'99, 1999, pp. 254-261.

[25] Zobel J.: Collection Selection via Lexicon Inspection. Proceedings of The Second Australian Document Computing Symposium, 1997.

Appendix 1. Search Model Equation

The Okapi probabilistic model [18] calculates the weight of the term t within a document d as follows:

$$w_{td} = (k_1 + 1) \cdot \frac{tf_{td}}{K + tf_{td}}$$

where:

$$K = k \cdot \left[(1-b) + b \cdot \frac{l_d}{advl_I} \right]$$

l_d is the document length,

$advl$ is the average of document length (set to 750),

b is a constant (set to 0.9),

k is a constant (set to 2),

k_1 is a constant (set to 1.2),

tf_{td} is the occurrence frequency of the term t in document d.

The following formula shows the weight given to the same term t within the query q:

$$w_{tq} = \frac{tf_{tq}}{k_3 + tf_{tq}} \log\left(\frac{n - df_t}{df_t}\right)$$

where:

tf_{tq} is the search term frequency,

df_t is the number of documents in the collection containing the term t,

n is the number of documents included to the collection.

k_3 is a constant (set to 1000).

Appendix 2. Precision of Various Selection and Merging Strategies Combinations

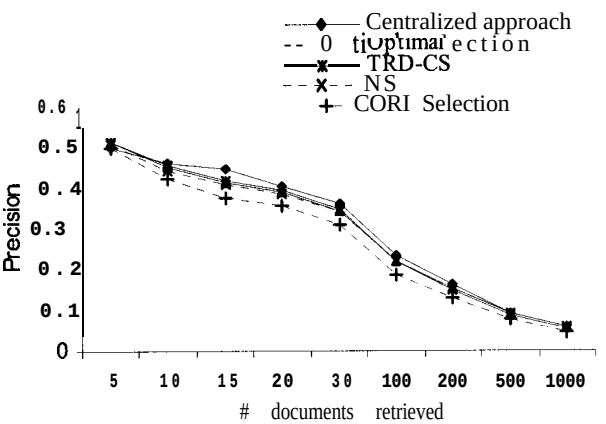

Figure 1. TREC8 testbed, LMS strategy.

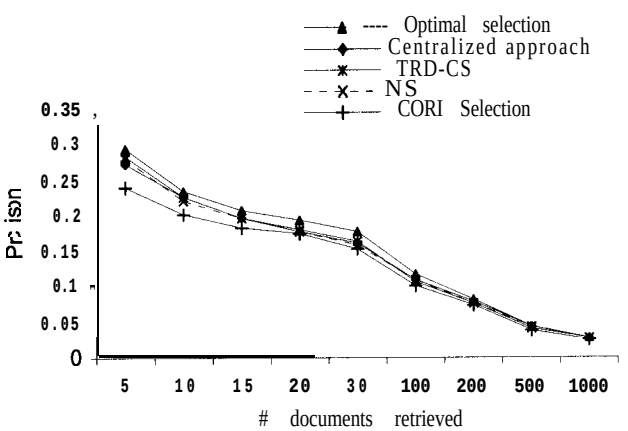

Figure 2. TREC9 testbed, LMS strategy.

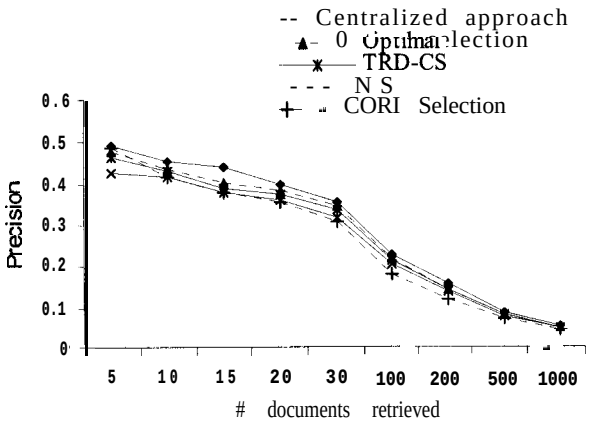

Figure 3. TREC8 testbed, Cori merging.

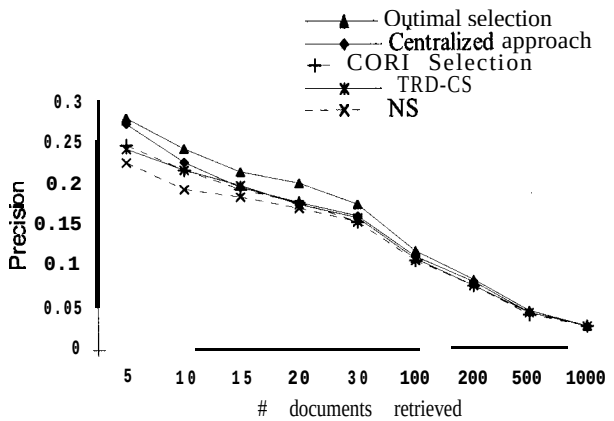

Figure 4. TREC9 testbed, Cori merging.

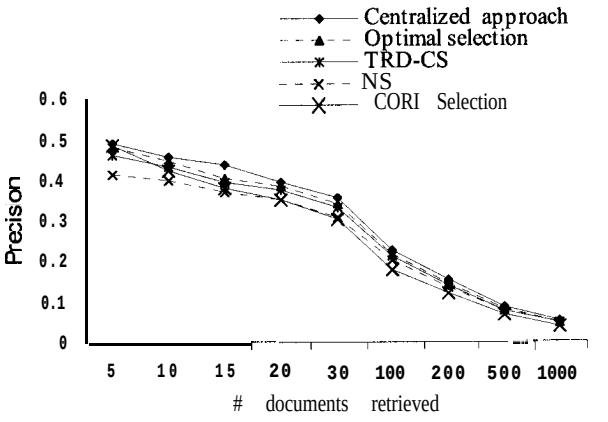

Figure 5. TREC8 testbed, RSM strategy.

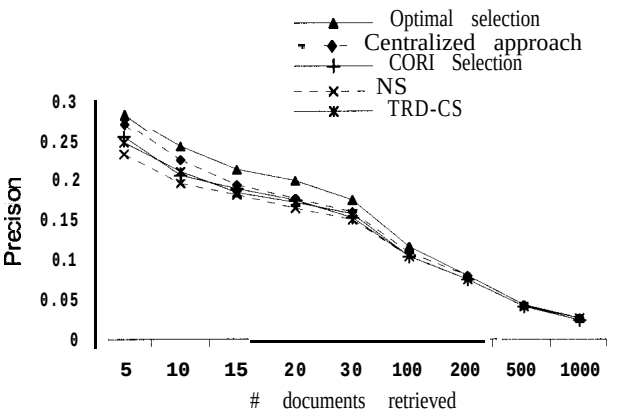

Figure 6. TREC9 testbed, RSM strategy.