# Report on the TREC-10 Experiment: Distributed Collections and Entrypage Searching

Jacques Savoy, Yves Rasolofo

Institut interfacultaire d'informatique, Université de Neuchâtel (Switzerland)
E-mail: {Jacques.Savoy, Yves.Rasolofo}@unine.ch  Web site: http://www.unine.ch/info/

## Summary

For our participation in TREC-10, we will focus on the searching distributed collections and also on designing and implementing a new search strategy to find homepages. Presented in the first part of this paper is a new merging strategy based on retrieved list lengths, and in the second part a development of our approach to creating retrieval models able to combine both Web page and URL address information when searching online service locations.

## Introduction

The Web of today represents a new paradigm, one that generates new challenges for the IR community. Included among these are: managing huge amounts of documents via distributed IR models, crawling through the Web in order to find appropriate Web sites to index, accessing documents written in various languages, measuring the quality or authority of available information, providing answers to very short user requests often expressed in ambiguous terms, satisfying a large range of search types (ad hoc, question-answering, location of online services, and interactive searches for specific document types or Web pages in order to satisfy a particular geographical or time constraint).

For our participation in TREC-10, we are focusing on two problems. One involves the presentation of a new merging strategy (collection fusion problem, Chapter 1) for the Web ad hoc track, and the other developing a search strategy intended to resolve homepage search problems (Chapter 2).

In order to evaluate our hypothesis when implementing the Okapi probabilistic model (Robertson *et al.*, 2000) we will use the SMART system as a test bed. This year our experiments are fully automated.

## 1. Distributed collections

In order to evaluate the retrieval effectiveness of various merging strategies, we formed four separate collections from the WT10g test collection (Savoy & Rasolofo, 2001). The same indexing scheme and retrieval procedure is used for each collection involved in this study. This type of distributed context more closely reflects digital libraries or search engines available on the Internet than do meta search engines, where different search engines may collaborate in response to a given user request (Selberg, 1999; Le Calvé & Savoy, 2000).

This chapter is organized as follows: Section 1.1 explains our indexing and search model. Section 1.2 describes related work on database merging strategies, while Section 1.3 presents our merging procedure. Finally, in Section 1.4 we evaluate our search model.

### 1.1. Indexing and retrieval scheme

From the original Web pages, we retained only the following logical sections: <TITLE>, <H1>, <CENTER>, <BIG>, with the most common tags <P> (together with </P>) being removed. Texts delimited by <DOCHDR>, </DOCHDR> tags were also removed. For longer requests, various insignificant keywords were removed (such as "Pertinent documents should include …"). Moreover, search keywords appearing in topic title sections were assigned a term frequency of 3 (a feature that should have no impact on short requests).

For the ad hoc Web track, we conducted different experiments using the Okapi probabilistic model, in which the weight $w_{ij}$ was assigned to a given term $t_j$ in a document $d_i$ and was computed according to the following formula:

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

$$\text{with } K = k_1 \cdot \left[ (1-b) + b \cdot \frac{l_i}{avdl} \right] \qquad (2)$$

where $tf_{ij}$ indicates the within-document term frequency, and b, $k_1$ are parameters. K represents the ratio between the length of $d_i$ measured by $l_i$ (sum of $tf_{ij}$) and the document mean length is denoted by advl.

To index each search keyword $t_j$ included in the request, the following formula was used:

$$w_{qj} = \frac{tf_{qj}}{k_3 + tf_{qj}} \cdot \ln\left[\frac{N - df_j}{df_j}\right] \qquad (3)$$

where $tf_{qj}$ indicates the search term frequency, $df_j$ the collection-wide term frequency, N the number of documents in the collection, and $k_3$ is a parameter.

To adjust the underlying Okapi search model parameters, we used the values suggested by Walker *et al.* (1998): advl = 900, b = 0.75, $k_1 = 1.2$, and $k_3 = 1000$. We did however believe that it might be more effective to assign a lower value to the parameter b, and in order to verify this assumption. We also evaluated the Okapi model using b = 0.7 or b = 0.5, values, resulting in interesting retrieval performances for TREC-9 topics.

Finally, for the request q containing m search terms the retrieval status value (denoted $RSV_i$) of a Web page $d_i$ was estimated as:

$$RSV(d_i, q) \;=\; RSV_i \;=\; \sum_{j=1}^{m} w_{ij} \quad w_{qj} \qquad (4)$$

In order to obtain a broader picture of our evaluations, we considered two different query formulations: (1) using only the Title section (T) or (2) all three logical sections (Title, Descriptive and Narrative, noted T-D-N). Finally, we should mention that these queries were "real topics" in the sense that they were taken from a MSNSearch log.

### 1.2. Previous work on merging strategies

Various solutions have been suggested for merging separate result lists obtained from distributed collections. As a first approach, and taking only the rank of the retrieved items into account, we might interleave results in a round-robin fashion. According to previous studies (Voorhees *et al.*, 1995; Callan *et al.*, 1995; Savoy & Rasolofo, 2001; Rasolofo *et al.*, 2001), such interleaving schemes have a retrieval effectiveness of around 20% to 40% below that achieved from single retrieval schemes, working with a single huge collection representing an entire set of documents.

In order to account for document scores computed for each retrieved item (or its retrieval status value), we might formulate the hypothesis that each collection is searched by the same or very similar search engines and that RSV values are therefore directly comparable (Voorhees *et al.*, 1995; Savoy & Rasolofo, 2001; Rasolofo *et al.*, 2001). Such a strategy, called raw-score merging, produces a final list sorted by the document score computed by each collection. However, as indicated by Dumais (1994), collection-dependent statistics contained in document or query weights may vary widely among collections, and therefore this phenomenon may invalidate the raw-score merging hypothesis.

To deal with this fact, we could normalize document scores within each collection through dividing them by the maximum score (i.e., the document score of the retrieved record found in the first position).

Callan *et al.* (1995) and Xu & Callan (1998) suggested a merging strategy called CORI, one that incorporates scores achieved by both collection and document. The collection score corresponds to the probability that the related collection would respond appropriately to the current request. In this scheme, each collection is viewed as a huge document and we might therefore use an IR scheme to rank the various collections according to the submitted request, since IR systems rank these documents according to their retrieval status values. In a second step, we simply multiply the document scores by the corresponding collection scores and then sort the result lists according to this value.

### 1.3. Our merging strategy

Our new merging strategy, denoted LMS for "using result Length to calculate Merging Score", and as does the CORI model, begins by estimating a score for each collection. The underlying idea is to use these weights to increase document scores from those collections having scores greater than the average score, and to decrease those for any collections having scores less than the average score. Our approach has the advantage of being simple, since it only uses document scores and result lengths as input. Also, since collection statistics are not required, systems using our approach do 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 sort of cooperation between the main system 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 were to find more documents. The score for the *k*th collection is determined by:

$$s_k = \ln\left(1 + l_k \cdot K \Big/ \sum_{j=1}^{|C|} l_j\right)$$

where

- K is a constant (set to 600 in our evaluations),
- $l_k$ is the number of documents retrieved by the *k*th 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 (Le Calvé & Savoy, 2000). Based on this collection score, our merging algorithm calculates the collection weight denoted $w_k$ for the *k*th collection as follows:

$$w_k = 1 + \left[\left(s_k - s_m\right)/s_m\right]$$

where

- $s_k$ is the *k*th collection score, and
- $s_m$ is the mean collection score.

As in the CORI approach, the final document score is the product of $w_k$ and the document score $RSV_i$ computed by the server for this document. The value of this product is used as the key to sort the retrieved items in the final single result list.

## 1.4. Evaluation

To evaluate our propositions, we first used the TREC-9 topics (50 queries, 2,617 relevant documents) taken from the WT10g test collection. Average precision comparisons (computed by the TREC-EVAL system based on 1,000 retrieved items) achieved by the raw-score merging approach are depicted in the second column of Table 1a. On the other hand, average precision decreases with the use of round-robin merging strategy (third column of Table 1a). As one can see, considering result lengths in the merging process may marginally improve average precision over this baseline (last column of Table 1a).

After having considered different values for the parameter b, a smaller value (e.g., b = 0.5) seems to improve average precision (from 20.04 to 20.64, meaning an enhancement of +3% using raw-score merging or +2.6% when using our result length merging scheme (20.24 vs. 20.76)).

From studying the retrieval performance using TREC-10 topics (Table 1b), our previous findings were confirmed: the round-robin strategy results in lower average precision. From an analysis of parameter b, one can see that when setting b = 0.5, there is an improvement of +3.4% (from 16.59 to 17.16 in average precision). This fact is not however confirmed by longer requests, where the best value for the parameter b seems to be 0.7.

The data in Table 1b also indicates that by taking more search terms into account we can increase retrieval effectiveness substantially (around +30%). Finally, Table 3 provides a detailed analysis of our official runs applied to the Web ad hoc track, and Table 1b lists their retrieval performance in bold characters.

In order to analyze our merging strategy, we listed the number and the percentage of relevant items provided by each collection in Table 4. As one can see,

| | Average precision (% change) | | |
|---|---|---|---|
| Query (Title only)<br>Model / merging strategy | TREC-9<br>50 queries<br>Raw-score | TREC-9<br>50 queries<br>Round-robin | TREC-9<br>50 queries<br>Result lengths |
| Okapi (b=0.75) | 20.04 | 17.66 (-11.9%) | 20.24 (+0.9%) |
| Okapi (b=0.7) | 20.34 | 17.96 (-11.7%) | 20.60 (+1.3%) |
| Okapi (b=0.5) | 20.64 | 17.66 (-14.4%) | 20.76 (+0.6%) |

**Table 1a.** Average precision of various retrieval schemes based on TREC-9 topics

| | Average precision (% change) | | | | |
|---|---|---|---|---|---|
| Query<br>Model / merging | TREC-10<br>Title only<br>Raw-score | TREC-10<br>Title only<br>Round-robin | TREC-10<br>Title only<br>Result lengths | TREC-10<br>Title-Desc-Narr<br>Raw-score | TREC-10<br>Title-Desc-Narr<br>Round-robin |
| Okapi (b=0.75) | **16.59** | 16.43 (-1.0%) | **17.15** (+3.4%) | 22.12 (+33.3%) | 20.39 (+22.9%) |
| Okapi (b=0.7) | 16.73 | 16.24 (-2.9%) | **16.99** (+1.6%) | **22.42** (+34.0%) | 20.76 (+24.1%) |
| Okapi (b=0.5) | 17.16 | 16.03 (-6.6%) | 17.50 (+2.0%) | 21.68 (+26.3%) | 19.87 (+15.8%) |

**Table 1b.** Average precision of various retrieval schemes based on TREC-10 topics

| Run name | Aver. pr. | Query | Parameter | Merging |
|---|---|---|---|---|
| UniNEtd | 16.59 | T | b = 0.75 | raw-score merging |
| UniNEtdL | 17.15 | T | b = 0.75 | result-length merging |
| UniNEt7dL | 16.99 | T | b = 0.7 | result-length merging |
| UniNEn7d | 22.42 | T-D-N | b = 0.7 | raw-score merging |

**Table 3.** Description of official Web ad hoc run

the first collection (WT10g.1) contains a larger number of pertinent Web pages and the third collection (WT10g.3) a smaller number. From looking at the top 10, the top 100 and the first 1,000 retrieved pages, we can see how the percentage of pages extracted from each collection varies. More precisely, these numbers increase for the first and fourth collection and decrease for the other two.

| | |
|---|---|
| Number of queries | 50 |
| Number of relevant doc. | 3,363 |
| Mean rel. doc. / request | 67.26 |
| Standard error | 11.81 |
| Median | 39 |
| Maximum | 372 (q#: 541) |
| Minimum | 2 (q#: 506, 538, 548) |

**Table 2.** Relevance judgment statistics (TREC-10)

| | Percentage of retrieved items | | | |
|---|---|---|---|---|
| | WT10g.1 | WT10g.2 | WT10g.3 | WT10g.4 |
| # rel. items | 1007 | 800 | 640 | 916 |
| % of rel. | 29.94% | 23.79% | 19.03% | 27.24% |
| Round-robin | 25% | 25% | 25% | 25% |
| Top 10 | 13.2% | 28.8% | 39.6% | 18.4% |
| Top 100 | 19.54% | 27.02% | 30.62% | 22.82% |
| Top 100 | 23.53% | 25.16% | 27.50% | 23.82% |

**Table 4.** Distribution of retrieved items (UniNEtd, raw-score merging, 50 topics)

## 2. Homepage searching

In the previous chapter, users sending a request to our search engine would obtain a ranked list of Web pages containing pertinent information about their information need. In this chapter, our objective is to design and implement a search strategy that would retrieve, at the limit, only one pertinent Web page that corresponds to the entrypage or to the online service location being sought by the user. For example, when users submit a request for "Quantas", they will retrieve the Quantas Airlines homepage, not several Web pages about this airline company.

To achieve this objective, we will first search URL addresses (Section 2.1). As a second search strategy, we will implement a combined retrieval model (Section 2.2) that searches the Web pages (Section 2.3) and then reranks the retrieved list by URL address length (Section 2.4). We will then examine any similarity between the query and the corresponding URL addresses (Section 2.5), and finally combine these three approaches (Section 2.6). An evaluation of our official runs is given in Section 2.7.

### 2.1. Searching the URL address only

For each of the 1,692,096 Web pages included in the WT10g test collection, we know the corresponding URL address. Thus as a first approach, we will build a text collection from these URLs and then obtain a mean number of distinct indexing terms (5.58 per URL address, max = 28, min = 1). From the available requests, we might then search this text database using the various IR models described using SMART notations (Savoy & Picard, 2001).

In this first attempt, we considered using a classical retrieval scheme to find the correct URL address (e.g., "www.cdsnet.net:80/vidiot/") when responding to the query "Vidiot". From examining usability studies, it was recommended that URL addresses contain information about the owner's name (usually the company name) and/or about content that might help users find their way around the Web or within the site (Nielsen, 2000, p. 246). If this principle is applied, our approach may work well.

| | Simple queries | | Extended queries | |
|---|---|---|---|---|
| IR model | MRR | # top 10 | MRR | # top 10 |
| Okapi | 0.161 | 29 | 0.141 | 27 |
| Lnu-ltc | 0.077 | 15 | 0.091 | 17 |
| atn-ntc | 0.013 | 3 | 0.016 | 6 |
| dtu-dtn | 0.108 | 20 | 0.108 | 21 |
| ltn-ntc | 0.010 | 2 | 0.014 | 5 |
| ntc-ntc | 0.215 | 37 | 0.187 | 41 |
| ltc-ltc | **0.217** | **40** | 0.192 | 44 |
| lnc-ltc | 0.203 | 37 | **0.197** | **47** |
| bnn-bnn | 0.009 | 1 | 0.009 | 1 |
| nnn-ntn | 0.009 | 1 | 0.012 | 3 |
| nnn-nnn | 0.004 | 1 | 0.005 | 1 |
| Mean | 0.093 | 16.91 | 0.088 | 19.36 |

**Table 5.** Evaluation of URL searches (TREC-10, 145 topics)

In this first experiment, we evaluated eleven IR models, and as a retrieval performance measure, we calculated the mean reciprocal rank (MRR) over 145 topics (see Table 5, Column 2). As a second measure, we noted the number of topics for which a correct entrypage was found in the top 10 (see Table 5, Column 3). Overall, this search strategy does not work well for the problem of finding homepages. Moreover, although the Okapi probabilistic model provides the best search engine performance (Savoy & Picard, 2001), it is not best in terms of retrieval performance. For this particular retrieval task it seems that the vector-space model "doc=ltc, query=ltc" represents the best IR scheme, being able to retrieve 40 correct entrypages in the top 10 (over a total of 145, or 27.6% of the cases).

This rather limited performance thus reflects the fact that words found in an URL address are not necessarily those one would place in a query. For example, URLs often contain abbreviations (e.g., "www.iti.gov.sg" is the URL for "Information Technology Institute"). Moreover, if a given query contains very frequently occurring words (e.g., "of" "at", "in"), we add a second form of acronym that ignores these terms (e.g., from the request "Point of View Cafe", we form a first acronym as "povc" and a second as "pvc").

Concatenation represents another form of URL construction, with two (or more) words being joined (e.g., "www.dogzone.com" and "Dog Zone's dog clubs") or only the first (of the first two) letter(s) of a word are concatenated with the second word (e.g., "Digital Realms" expressed as "www.drealms.co.uk"). In order to deal with these various word formations, we designed our system such that it considers various URL construction possibilities. For example, for a simple request such as "Worldnet Africa", our system would construct the following expanded query: "Worldnet Africa wa worldnetafrica worldneta aworldnet woafrica worldnetaf".

Evaluating these extended request forms does not however result in appreciable performance enhancements, as can be seen in the last two columns in Table 5. Although the number of correct entrypages found in the top 10 seems to increase, the MRR measure indicates degradation. Thus, using only URL texts does not seem to be an adequate strategy, since establishing a link between query words and a URL addresses is a difficult task (e.g., based on the query "PiperINFO" which finds the URL "www.hamline.edu/" or "DaMOO" that finds the URL "lrc.csun.edu").

### 2.2. Guidelines for our combined search model

In order to develop a better search strategy, we decided to construct a two-stage retrieval strategy. In the first stage we used the Okapi probabilistic model to search Web page content for relevant homepages, although we did not employ the exact same Okapi search model used in the Web ad hoc track (see Section 1.1). Rather, we adapted the search model described in Section 2.3, and from the list of retrieved Web pages we were able to generate a corresponding list of URL addresses.

In the second stage of our retrieval strategy we inspected the corresponding URL addresses in order to verify whether or not they could be considered as appropriate URL candidates. To do so, we considered URL length, attributing more importance to short addresses (Section 2.4). Thus, in order to appear within the first positions in our final result list, a Web page would contain the search's keywords within its first 50 terms and its URL address would have to be short. As an alternative, we believe that URL address content should bear some similarity to the submitted request (Section 2.5), meaning we would again rank our retrieved list according to this similarity.

So far we have considered three types of retrieval expertise. The first retrieves and ranks Web sites according to page content, the second reranks these results according to URL address length and the last reranks the results according to URL address and submitted request similarity. In order to account for the results of these three approaches, we suggested reranking the retrieved items according to these three expert opinions (Section 2.6). Thus, with this search strategy, an entrypage will be found within the first positions if its Web page shares common words with the request, if its URL address is short and if this address contains some of the search keywords (or an abbreviation, a concatenation of two search keywords, or some URL construction as shown in Section 2.1).

### 2.3. Okapi model adaptation

In the first and most important stages of our combined retrieval strategy, we employed the Okapi probabilistic model to search Web page content for relevant homepages, although we did not employ the exact same Okapi search model as used in the Web ad hoc track (see Section 1.1). In fact, we added a precision device that considered only the first 50 words of each item retrieved and ranked only those documents having at least one query term within the first 50 words. This precision device was based on our intuition that document titles (or document headings) should provide an adequate indication as to whether or not corresponding Web pages are relevant to a given homepage search.

This feature works as follows. First, we form the set of all search keywords pairs. For example, from the query $q = (t_i, t_j, t_k)$, we may deduce the following six terms pairs $(t_i, t_j)$, $(t_j, t_i)$, $(t_i, t_k)$, $(t_k, t_i)$, $(t_j, t_k)$ and $(t_k, t_j)$. We then inspect the document to see if the corresponding couple of search terms appears within a maximum distance of 5 (or with a maximum of four terms between the pair of search words). For example, supposing we are looking for the pair of search terms $(t_j, t_k)$, then if we find a word sequence $(t_j, t_a, t_b, t_c, t_d, t_k)$, we search it for an occurrence of our pair of search terms within a maximum distance of 5. The weight assigned to the occurrence of this search keyword pair $(t_j, t_k)$ in the document $d_i$ is denoted as $\delta w_{jk}$ and computed as follows:

$$\delta w_{jk} = 0.5 + 5 / \sqrt{\text{position}(t_k)}$$

where position($t_k$) indicates the position (word number) of the term $t_k$ from the beginning of the Web page. Thus, if the term $t_k$ appears in the second position (in this case, the first position occupied by $t_j$), the function position($t_k$) returns 1 and $\delta w_{jk}$ is 5.5. On the other hand, if the distance is greater than 5, we ignore this occurrence.

It is possible however that a pair of search keywords ($t_j$, $t_k$) would appear more than once (within a maximum distance of 5) in the document $d_i$. To account for all occurrences of the search term pairs ($t_j$, $t_k$), we compute the following expression:

$$w_{i(j,k)} = \left[ \frac{(k_1 + 1) \sum_{occ(j,k)} \delta w_{jk}}{K + \sum_{occ(j,k)} \delta w_{jk}} \right] \min(w_{qj}; w_{qk})$$

where $w_{i\,(j,k)}$ represents the weight attached to the search term phrase ($t_j$, $t_k$) in the document $d_i$, the parameter $k_1$ and K are evaluated as described in Equation 2 and $w_{qj}$ and $w_{qk}$ represent the weights of the search keyword $t_j$ and $t_k$ in the request (as shown in Formula 3).

In order to consider all occurrences of all search keywords pairs, we compute an additional weight $ph_{(d_i,q)}$ attached to the presence of these multiple search keywords pair occurrences in document $d_i$ as follows:

$$ph_{(d_i,q)} = \sum_{\text{all } (j,k)} w_{i(j,k)}$$

The presence of search keyword pairs within the beginning of a given Web page $d_i$ would change the value of its $RSV_i$, and this would be done simply by adding the phrase weight $ph_{(d_i,q)}$ to the previously computed $RSV_i$ (see Equation 4), as follows:

$$RSV_i' = RSV_i + ph_{(d_i,q)} \qquad (5)$$

However, we suggest a variation that assigns a lower phrase weight $ph_{(d_i,q)}$ if the document $d_i$ does not appear in the top ranked retrieved documents. Thus an alternative retrieval status value is assigned using the following formula:

$$RSV_i' = RSV_i + (1 - \alpha) \; ph_{(d_i,q)} \qquad (6)$$

$$\text{with } \alpha = \frac{RSV_{max} - RSV_i}{RSV_{max} - RSV_{min}}$$

where $RSV_{max}$ and $RSV_{min}$ are the RSV values assigned by the first and the last retrieved items respectively (computed according to Equation 4).

| Run name | MRR | # in top 10 | # not found |
|---|---|---|---|
| Okapi stem | 0.261 | 65 (44.8%) | 31 (21.4%) |
| Okapi nostem | 0.274 | 72 (49.7%) | 29 (20.0%) |
| Eq.5, stem | 0.348 | 85 (58.6%) | 26 (17.9%) |
| Eq.5, nostem | 0.354 | 84 (57.9%) | 26 (17.9%) |
| Eq.6, stem | 0.343 | 83 (57.2%) | **24** (16.6%) |
| Eq.6, nostem | **0.367** | **86** (59.3%) | **24** (16.6%) |

**Table 6.** Evaluation of results using various Okapi probabilistic models (TREC-10)

Table 6 lists the various results of our search models, in which we reported the mean reciprocal rank (MRR), the number of queries for which the correct homepage was found within the first ten retrieved items, and the number of queries for which the relevant entry-page could not be found in our result list. Row two of this table contains an evaluation of the classical Okapi probabilistic model, as described in Section 1.1. As a variation in this particular search problem, we decided to analyze the impact of the stemming procedure, and as seen in row three, we were able improve retrieval effectiveness compared to the classical Okapi model, when the stemming procedure was discarded.

In the fourth and fifth rows are listed our adapted Okapi model's retrieval performance, based on the retrieval status values computed according to Equation 5. Finally, the last two rows show the performance achieved by using Equation 6 with our adaptation of the Okapi model.

From analyzing this data we concluded that the stemming procedure was not really appropriate for this type of search. Moreover, our modified Okapi model provides better results than does the classical Okapi model, and computing retrieval status value using Equation 6 exhibits the best retrieval performance.

### 2.4. Reranking based on URL length

Upon examining the result lists obtained using our Okapi homepage search model, we can find corresponding URL addresses using a look-up procedure and pass this list to one of our reranking schemes. In this second step, we first consider the URL address length, its length being defined by the number of "/"s contained within it. If however a URL address ends with a "/", then this final slash is ignored. Also, for any URL addresses ending with "index.html" or "index.htm", these terms are removed before we compute the length. Thus, the URL "www.ibm.com" has the same length as "www.ibm.com/index.html" or "www.ibm.com/".

| URL length | Number of observations | | |
|---|---|---|---|
| | Number | Percentage | Cumul. |
| = 1 | 11,622 | 0.69% | 11,622 |
| = 2 | 253,250 | 14.97% | 264,872 |
| = 3 | 458,356 | 27.09% | 723,228 |
| = 4 | 451,255 | 26.67% | 1,174,483 |
| = 5 | 297,339 | 17.57% | 1,471,822 |
| = 6 | 119,035 | 7.03% | 1,590,857 |
| = 7 | 69,091 | 4.08% | 1,659,948 |
| = 8 | 19,612 | 1.16% | 1,679,560 |
| = 9 | 6,399 | 0.38% | 1,685,959 |
| = 10 | 1,235 | 0.07% | 1,687,194 |
| ≥ 11 | 4,902 | 0.29% | 1,692,096 |

**Table 7a.** URL length distribution

Table 7a shows the URL length distribution across our test collection, while Table 7b depicts the same distribution based on our two relevance sets (entry-page 2000 and entrypage 2001). From the training data (denoted "2000"), we found that correct answers usually correspond to short URL addresses, those with lengths of 1 (77 cases in Table 7b) or 2 (15 observations). Data from the Entrypage 2001 test collection provided by relevance assessments displays a similar pattern. Thus it seems reasonable to assign more importance to short URL addresses than to longer ones.

| | All relevant items | | One rel. item / query | |
|---|---|---|---|---|
| URL length | 2000 | 2001 | 2000 | 2001 |
| = 1 | 79 | 138 | 77 | 93 |
| = 2 | 19 | 56 | 15 | 32 |
| = 3 | 8 | 33 | 7 | 9 |
| = 4 | 2 | 16 | 1 | 6 |
| = 5 | 0 | 7 | 0 | 4 |
| = 6 | 0 | 0 | 0 | 0 |
| = 7 | 0 | 2 | 0 | 1 |
| Total | 108 | 252 | 100 | 145 |

**Table 7b.** URL length distribution for a set of relevant items

Based on these findings, we reranked the retrieved URL addresses according to the inverse of their length, with ties being broken by using retrieval status values ($RSV'_i$) computed according to our Okapi model (Section 2.3).

Table 8 shows the first five URLs retrieved after the first query, according to our adaptation of the Okapi model (top part) or according to our reranking scheme based on URL length (second part). As one can see, the Okapi model retrieved various Web pages from the same Web site ("africa.cis.co.za"). The retrieval status value computed according to this search model, as depicted in Column 4, does not vary greatly. When we reranked this result list according to the inverse of URL length, the relevant item ("africa.cis.co.za:81") appears in the first position. However, the first five URLs have the same length (1 in this case), and the second key used is always the retrieval status value computed by the Okapi system.

## 2.5. Reranking based on URL similarity

URL address length does not however account for similarity between request and URL terms. Based on the data depicted in Table 8, for the response to "Worldnet Africa" we can see that the URL address "www.kvvp.com" response is listed in second place. Thus, it seems a good idea to rerank the result list provided by our Okapi model, based on a similarity between the URL address and the request.

This type of scheme is advantageous because we can account for any similarity between requests and Web pages, as well as requests and URL addresses. To compute this similarity between requests and URL addresses, we must however take various phenomena into consideration, including acronyms and concatenations of two search keywords found in URL addresses, etc. (see Section 2.1).

The basic principals underlying our similarity measure are described in Table 9, where we distinguish mainly between three cases. First, when a request is one word only, our similarity function determines whether this word appears in the URL head (defined as the server name) or in the URL's last component (defined as the tail of the URL). If it does and the corresponding URL is short (with a length of 1 or 2), we return the maximum value of 1.0. One the other hand, if this search term appears within the URL, the similarity is defined as the inverse of the URL's length. Finally, we determine whether there might be a fuzzy match between the search keyword and the URL. In this "fuzzySimilarity()" function, we counted the maximum length of the ordered sequence of letters between the search word and each term appearing in the URL. For example, for the search word "market" and the word "markCie", the maximum ordered sequence is "mark" which has a length of 4. This length is then divided by the maximum length of the two corresponding terms (7 in our case, giving a final fuzzy similarity of 4/7).

As a second case, we computed the similarity between the request and a URL with a length of one. Here we tried to establish a similarity between the request and some variations of this short URL (its acronym, the concatenation of adjacent word pairs, the concatenation of a word with the two letters of the next term).

| | | | Similarity measurement | | |
|---|---|---|---|---|---|
| Query | URL address | Rank | $RSV'_i$, Okapi | URL request | URL length |
| | based on our adaptation of the Okapi model | | | | |
| 1 | africa.cis.co.za:81/nibs/postnet/ad.html | 1 | 5242.61 | 0.25 | 0.25 |
| 1 | africa.cis.co.za:81/about/newprice.html | 2 | 5140.34 | 0.333333 | 0.333333 |
| 1 | africa.cis.co.za:81/buy/ad/kyalami/kyalami.html | 3 | 5116.08 | 0.2 | 0.2 |
| 1 | africa.cis.co.za:81/buy/ad/fasa/fbhs2.html | 4 | 5110.59 | 0.2 | 0.2 |
| 1 | africa.cis.co.za:81/cape/ctcc/business/taxation.html | 5 | 5109.63 | 0.2 | 0.2 |
| | rerank based on URL length | | | | |
| 1 | **africa.cis.co.za:81/** | 1 | 4967.93 | 0.9999 | 1.0 |
| 1 | www.kvvp.com:80/ | 2 | 2672.09 | 0.12 | 1.0 |
| 1 | www.krok.com:80/ | 3 | 2636.3 | 0.16 | 1.0 |
| 1 | www.starhustler.com:80/ | 4 | 2560.11 | 0.18 | 1.0 |
| 1 | www.lcrtelecom.com:80/ | 5 | 1987.4 | 0.2 | 1.0 |
| | rerank based on the similarity URL - request | | | | |
| 1 | **africa.cis.co.za:81/** | 1 | 4967.93 | 0.9999 | 1.0 |
| 1 | www.att.com:80/worldnet/ | 2 | 2607.29 | 0.997 | 0.5 |
| 1 | www.legnetwork.com:80/worldnet.htm | 3 | 2408.45 | 0.997 | 0.5 |
| 1 | interknowledge.com:80/south-africa/index.html | 4 | 2275.12 | 0.997 | 0.5 |
| 1 | www.biodiv.org:80/africa.htm | 5 | 2143.38 | 0.997 | 0.5 |
| | Merged results from our three experts | | | | |
| 1 | **africa.cis.co.za:81/** | 1 | 9935.86 | 1.9998 | 2.0 |
| 1 | africa.cis.co.za:81/facility.html | 2 | 10086.1 | 0.778 | 1.0 |
| 1 | africa.cis.co.za:81/fin&tegn/ | 3 | 9887.3 | 0.778 | 1.0 |
| 1 | africa.cis.co.za:81/cpyright.html | 4 | 9850.68 | 0.778 | 1.0 |
| 1 | africa.cis.co.za:81/comp.html | 5 | 9809.52 | 0.778 | 1.0 |

**Table 8.** Example of four ranking approaches for the request "Worldnet Africa" (relevant item depicted in bold)

For example as depicted in Table 9, the request "Worldnet Africa" and the URL "africa.cis.co.za:81" represents a similarity evaluated as 0.9999, and if no match was found, we applied our fuzzy match function.

In the latter case, we computed the similarity between each search keyword and a given URL (function inFuzzy()). To define the similarity measure, we took the number of matches, the length of the URL, the value of the match between the URL head and the URL tail into account, as shown in the last lines of Table 9.

In order to evaluate this reranking scheme, we ranked the URL address result list according to request their similarity. An example of the results of this reranking is shown in Table 8 (third part). For this query one can see the relevant item "africa.cis.co.za:81/" appears in the first position. The following four URL addresses have the same similarity value (depicted in fifth column of Table 8) and are ranked according to their retrieval status values, computed from our adaptation of the Okapi model (shown in the fourth column).

## 2.6. Evaluation and data mergers

So far, we have described two reranking schemes that might hopefully improve the ranking obtained from our adaptation of the Okapi model (for which the corresponding evaluation is shown in Table 6). Table 10a lists an evaluation of these two reranking schemes under the label "URL length" and "URL simil." The results depicted in this table indicate that we can enhance the mean reciprocal rank (MRR) and the number of queries for which the correct homepage can be found within the first ten retrieved items. Moreover, we may also decrease the number of queries for which the relevant entrypage cannot be found in our result list (having a size of 100). Based on these results, the best scheme seems to be that of reranking, based on URL length.

As shown in Table 8 however, each of our three search systems seems to retrieve different URL addresses. Thus, we have decided to combine these three expert techniques. To do so, we selected the first fifteen retrieved items from each of three result lists and separately added the corresponding similarities achieved by our three experts. This additive process was chosen because in other data merging contexts, it also provided the best results (Savoy *et al.*, 1997). The result lists are then sorted by URL length, with ties being broken by using the sum of retrieval status values computed by our Okapi model (Section 2.3). For example, the last part of Table 8 shows the first five retrieved items, listed according to this fusion scheme. For the first item, $RSV'_i = 9935.86$ because it was found by both the URL length expert ($RSV'_i = 4967.93$) and the URL similarity expert ($RSV'_i = 4967.93$). For the same reason, the new URL length is set to two and the new URL-request similarity is fixed at 1.9998 ($= 2 \cdot 0.9999$).

```
similarity (aQuery, anURL) :  Real;
    queryLength := queryLength (aQuery);                         //  john smith canada -> 3
    urlLength := urlLength (anURL);                              //  www.ibm.com/uk/products/  ->  3
    urlHead := urlHead (anURL);                                  //  www.ibm.com/uk/products/  ->  www.ibm.com
    urlTail := urlTail (anURL);                                  //  www.ibm.com/uk/products/  ->  products

    if (queryLength = 1) {
        if ( in(aQuery, urlHead) & (urlLength = 1))   return(1);         //  market  &  www.market.com/
        if ( in(aQuery, urlTail) & (urlLength <= 2))   return(1);        //  market  &  www.iti.com/market/
        if  (in(aQuery, anURL))   return(1/urlLength);                   //  market  &  www.iti.com/data/market/
        return (fuzzySimilarity (aQuery, anURL));                        //  market  &  www.marketCie.ch/prod/data/
        }
    if (urlLength = 1) {
        expandQuery := acronym (aQuery);                         //  chicago science center  +  csc
        if (in (expandQuery, anURL) >= 2)   return(1);           //  chicago science center csc &  www.csc.science.com/
        if (in (expandQuery, anURL))   return(0.9999);           //  chicago science center csc &  www.csc.com/
        expandQuery := concat (aQuery);                          //  john smith  +  johnsmith
        if (in (expandQuery, anURL))   return(1);                //  john smith johnsmith  &  www.johnsmith.com/
        expandQuery := concat2 (aQuery);                         //  advice corp  +  adviceco
        if (in (expandQuery, anURL))   return(1);                //  advice corp adviceco  &  www.adviceco.com/
        return (fuzzySimilarity (aQuery, anURL));                //  advice corp  &  www.advicorp.com/
        }
    simHead := fuzzySimilarity (aQuery, urlHead);
    simTail := fuzzySimilarity (aQuery, urlTail);
    if (urlLength = 2) {
        if (simTail >= 0.8)   return (simTail);                  //  sirius bar  &  www.store.com/sirius/
        if (simHead >= 0.8)   return (0.456)                     //  iris corp  &  www.iris.com/ca/
            else return (0);                                     //  iris corp  &  www.irt.com/canada/
        }
    aSetMatchFuzzy := inFuzzy (aQuery, anURL);                   //  desk publ  &  www.publ.com/uk/desk/ -> (1, 0, 1)
    nbMatch := sizeSet (aSetMatchFuzzy);                         //  (1, 0, 1)  ->  2
    if (nbMatch = 0)  return (0);                                //  when no fuzzy match can be found
    if (urlLength - nbMatch <= 1) {                              //  numerous matches
        if ((aSimHead >= 0.8) & (aSimTail >= 0.8))               //  if good match with the head and the tail
            return (aSimTail - 0.1);                             //  e.g., desk publ  &  www.publ.com/uk/desk  -> 0.9
        if (aSimHead >= 0.8)                                     //  if good match with only the head
            return (aSimTail - 0.15);
            else return (0.234);                                 //  if numerous match inside the url
        }
    return (max (0.2, nbMatch / urlLength));                     // if some matches
```

**Table 9.** Outline of algorithm used to determine similarity between query and URL address

However, from considering only the top 15 items for each of our three search models, a maximum of 45 retrieved items per query could be obtained. In order to increase these results to 100 (and to help generate relevance assessments), we expanded this list by adding URL addresses found by our Okapi model.

| Run name | MRR | # in top 10 | # not found |
|---|---|---|---|
| Okapi only | 0.367 | 86 (59.3%) | 24 (16.6%) |
| URL length | 0.653 | 112 (77.2%) | 21 (14.5%) |
| URL simil. | 0.470 | 95 (65.5%) | 18 (12.4%) |
| Fusion | **0.693** | **115** (79.3%) | **10** (6.9%) |

**Table 10a.** Evaluation of our three search models and their combinations (corrected results)

The evaluation of our combined search model is depicted in the last row of Table 10a. The retrieval performance seems to have increased when considering MRR values or the number of queries for which the relevant item appeared in the top ten records. Moreover, the combination of our experts seems to have a clear impact on the number of queries for which the retrieval system cannot find corresponding relevant items (last column of Table 10a). If our combined retrieval model seems to perform well, it also has a drawback in that it retrieves various items corresponding to the same Web site, as shown in the last part of Table 8. Thus incorporating a pruning process in our fusion scheme may hopefully enhance retrieval performance.

When we created our official results for the homepage search problem, we selected the wrong Okapi results list before considering our two reranking schemes and our combined approach. The evaluations based on this incorrect result list are shown in Table 10b, and they reveal the same conclusions as do our unofficial but corrected search schemes (Table 10a).

| Run name | MRR | # in top 10 | # not found |
|---|---|---|---|
| Okapi only | 0.295 | 64 (44.1%) | 38 (26.2%) |
| URL length | 0.598 | 96 (66.2%) | 21 (14.5%) |
| URL simil. | 0.431 | 81 (55.9%) | 31 (21.4%) |
| Fusion | **0.637** | **100** (69.0%) | **12** (8.3%) |

**Table 10b.** Evaluation of our three search models and their combinations (official results)

### 2.7. Description of our official runs

Table 11 shows our official homepage search runs. The "UniNEep1" run corresponds to the search model merges described in Section 2.6. To produce the "UniNEep2" run in positions 45 to 50 we added the top five URL addresses found by our simple search model, as described in Section 2.1 (doc=nnn, query=ntn), if these URLs were not found previously. As depicted in Table 11, this feature does not have any significant impact on retrieval performance.

| Run name | MRR | # in top 10 | # not found |
|---|---|---|---|
| UniNEep1 | 0.637 | 100 (69.0%) | 12 (8.3%) |
| UniNEep2 | 0.637 | 100 (69.0%) | 11 (7.6%) |
| UniNEep3 | 0.529 | 99 (68.3%) | 10 (6.9%) |
| UniNEep4 | 0.477 | 99 (68.3%) | 16 (11.0%) |

**Table 11.** Official run evaluation

The "UniNEep4" run represents a variation of our search model, based on the normalized merging of the URL address searches (more precisely the "doc=nnn, query=ntn" model using our both our extended queries (see Table 5)) and our adaptation of the Okapi model (search Web page content, Section 2.3). The last run labeled "UniNEep3" represents the combined search model based on the "UniNEep4" run (with reranking based on URL length and URL similarity).

## Conclusion

The various experiments carried out within the Web track demonstrate that:

- our new merging strategy based on results list length may improve average precision slightly;
- using a lower value for the parameter b when dealing with short requests may improve retrieval performance;
- our adaptation of the Okapi model for the homepage search problem performs relatively well;
- reranking the URL addresses based on a combination of URL length and URL similarity with the request improves retrieval performance for our Okapi model.

**Acknowledgments**

The authors would like to thank C. Buckley from SabIR for allowing us the opportunity to use the SMART system. This research was supported by the SNSF (Swiss National Science Foundation) under grant 21-58'813.99.

**References**

Callan, J.P., Lu, Z. & Croft, W.B. (1995). Searching distributed collections with inference networks. In Proceedings of the ACM-SIGIR'95, 21-28.

Dumais, S.T. (1994). Latent semantic indexing (LSI) and TREC-2. In Proceedings of TREC'2, 105-115.

Le Calvé, A., & Savoy, J. (2000). Database merging strategy based on logistic regression. *Information Processing & Management*, 36(3), 341-359.

Nielsen, J. (2000). *Designing Web usability: The practice of simplicity*. Indianapolis: New Riders.

Rasolofo, Y., Abbaci, F. & Savoy, J. (2001). Approaches to collection selection and results merging for distributed information retrieval. In Proceedings ACM-CIKM'2001, 191-198.

Robertson, S.E., Walker, S. & Beaulieu, M. (2000). Experimentation as a way of life: Okapi at TREC. *Information Processing & Management*, 36(1), 95-108.

Savoy, J. & Picard, J. (2001). Retrieval effectiveness on the Web. *Information Processing & Management*, 37(4), 543-569.

Savoy, J. & Rasolofo, Y. (2001). Report on the TREC-9 experiment: Link-based retrieval and distributed collections. In Proceedings TREC'9, (to appear).

Savoy, J., Le Calvé, A. & Vrajitoru, D. (1997). Report on the TREC-5 experiment: Data fusion and collection fusion. In Proceedings TREC'5, 489-502.

Selberg, E.W. (1999). Towards comprehensive web search. Ph.D. Thesis, University of Washington (WA).

Voorhees, E. M., Gupta, N. K. & Johnson-Laird, B. (1995). Learning collection fusion strategies. In Proceedings of the ACM-SIGIR'95, 172-179.

Walker, S., Robertson, S.E., Boughamen, M., Jones, G.J.F. & Sparck Jones, K. (1998). Okapi at TREC-6: Automatic ad hoc, VLC, routing, filtering and QSDR. In Proceedings of TREC'6, 125-136.

Xu, J. & Callan, J. P. (1998). Effective retrieval with distributed collections. In Proceedings of the ACM-SIGIR'98, 112-120.