Strathprints Institutional Repository

Wu, S. and Gibb, F. and Crestani, F. (2003) *Experiments with document archive size detection.* In: Advances in information retrieval : 25th European Conference on IR Research, ECIR 2003, Pisa, Italy, April 14-16, 2003 : proceedings. Lecture notes on computer science, 2633 . Springer, Berlin, Germany, pp. 294-304. ISBN 3540012745

http://strathprints.strath.ac.uk/

Document Archive Size Detection

Shengli Wu, Forbes Gibb, and Fabio Crestani

Department of Computer and Information Sciences
University of Strathclyde, Glasgow, Scotland, UK
{S.Wu,F.Gibb,F.Crestani}@cis.strath.ac.uk

Abstract The size of a document archive is a very important parameter
for resource selection in distributed information retrieval systems. In this
paper, we present a method for automatically detecting the size (i.e.
number of documents) of a document archive, in case the archive itself
does not provided such information. In addition, a method for detecting
the incremental change of the archive size is also presented, which can
be useful for deciding if a resource description has become obsolete and
needs to be regenerated. An experimental evaluation of these methods
shows that they provide quite accurate information.

1 Introduction

When a huge number of document archives or document resources are available
via the Internet or large corporate networks, using a broker to select a subset
of available resource servers to satisfy the user's information need becomes an
important research issue. Many algorithms (see below) have been proposed which
can automatically rank a set of document resources, according to the degree of
their match with the given query.

Content-based resource selection algorithms need information about what
each resource contains. This information is called resource description of the re-
source. In some situations a resource description such as the size of the resource,
DF (document frequency) / ICF (inverse collection frequency) to a given query,
and so on, can be obtained from a cooperative resource. However, in multi-party
environments such as the Internet or large corporate networks, this information
may not be available. As discussed in [3], this can be caused by several reasons:
older database systems may be unable to cooperate; some services will refuse
to cooperate because they have no incentive or are allied with competitors; and
some services may misrepresent their contents, for example, to lure people to the
site. Callan, Lu and Croft concluded that all of these characteristics can be found
today on the Internet; some of them also occur in large corporate networks.

The size of a document resource is one of the most important pieces of in-
formation needed for resource selection. It is used in many resource selection
methods such as GlOSS/gGlOSS [5], CVV [10], decision-theoretic approach [4],
multi-objective model [9], and so on. CORI [3] does not depend on this informa-
tion, but considers that it is desirable and that discovering the size of a document
resource by sampling is an open problem.

A number of researches and surveys (e.g. [1,6]) have been conducted with respect to the WWW for estimating various statistical characteristics, such as the total number of web pages, web sites, institutions having web sites, web pages indexed by major search engines, and so on. Very recently, Liu, Yu, and Meng [7] proposed a method for estimating the number of documents indexed by a search engine. They randomly chose a group of sample documents n (without overlap), then randomly chose another group of documents m (with possible overlap). A formula $Total = n \times \frac{m}{o}$ is used for estimating the total number of documents, where o is the number of overlaps between these two groups. The method may be useful, but the discussion is very brief and many aspects remains unknown.

In this paper we present a method for detecting the size of document resources automatically where such information cannot be accessed from the resource server directly; or, if it is provided by the resource server, its accuracy is questionable and confirmation is required.

The paper is organised as follows: In Section 2 we present an algorithm for detecting the size of a document resource. Section 3 gives the setting and results of experiments for the algorithm introduced in Section 2. In Section 4 we discuss a method for detecting the incremental change in the resource size, provided that a thorough analysis has been done before. Section 5 concludes the paper.

2 Detecting the Size of a Document Resource

We use the following algorithm to detect the size of a document resource:

1. Use query-based sampling to obtain the language model of that resource.
2. Delete the words which have the highest frequency in the language model.
3. Use the remaining words in the language model to generate a group of queries.
4. Send the queries to the document resource and collect the top N documents returned by the resource for all those queries.
5. Count the number of different documents that have been retrieved.

A number of specific choices could be taken for some of these steps. For example, how many queries should be used for query-based sampling in step 1, or how to generate queries in step 3. In the following section, we will explain some of these choices, others of which have been discussed in Section 3.

2.1 Acquiring the Language Model by Sampling

The query-based sampling method was proposed by Callan, Connell and Du in [2] to create the language model (a group of words with their frequencies) of a document resource. This language model can be used for various activities, e.g., resource selection, summarising resource contents, query expansion, and so on. Callan, Connell and Du found that about 400 documents taken from the results of 100 queries can usually produce quite an accurate language model,

even though the words included in the language model are much less than those in the resource. A similar method has been used in our approach to create the language model. However, for our purpose, it is better to discover a larger set of words which appear in the resource than that used for resource selection. The reason for this is that in Steps 3 and 4 of the above algorithm, if there are more different words in the queries this should result in more documents and less document overlaps.

2.2 Query Formation

After we have obtained the language model of the resource, another issue is how to use those words in the language model to form queries. This mainly depends on the information retrieval system which may use a particular form of query interface. Here we assume that a query is composed of a group of words. This is the normal method used in many retrieval systems including Lemur [8], with which we carried out the experiments.

A simple method has been used for generating queries from those words. After deciding on the number of words which should be used in each query, words were randomly selected from the available word collection. Once a word was chosen, it was removed from the word collection so that it would not be used for any other queries.

From the language model, we can estimate the frequency of each word that appears in the resource. Through careful selection, e.g., by generating queries with words with diversified frequencies, and by avoiding using only low (or high) frequency words in a single query, we could have a better chance that the resource would return a similar number of documents for each query. Otherwise, some queries may only retrieve very few documents, while some others may retrieve a lot more. However, in our experiments we still used the simple random selection method, as we found that when a query is composed of more than 10 words, either carefully or randomly selected, the difference between the two approaches is negligible.

After a group of queries was generated, we specified a maximum number of documents which needed to be retrieved for every query. This is a simple approach. A more complicated approach would have been to assign to each query a different number of retrieved documents. However, we consider that that is more complicated and the effect is dubious since there is not much evidence for us to make sensible decisions. In the following, we have not considered this option.

If we have s queries, and specify that each query retrieves up to m documents, then we can get $n = s \times m$ documents in all. The ideal situation is that all the retrieved documents from different queries are different; that is to say, we can get n different documents in all. However, that is not likely and we need to find solutions to reduce document overlaps after retrieving a certain number of documents.

One issue that needs addressing is: how many words should we use for each query? We answer this question in Section 3 as a result of experiments.

3 Experiments and Results

Three full-text resources were used for the experiments. They were the Financial Times (FT), the Wall Street Journal (WSJ) and the first three CDROMs of the TREC Collection (TREC-123). The characteristics of these resources are presented in Table 1. The Lemur information retrieval system, developed at Carnegie Mellon University, was used [8]. Lemur provides three options as the retrieval model: vector space, okapi, and language model. Users can select one of the models for his/her retrieval task.

Resource	Size	N. of documents
FT (1991-94)	564 MB	210,158
WSJ (1988-90, 1992-94)	518 MB	173,252
TREC-123	3.2 GB	1,078,166

Table 1. Resources used in the experimentation.

The first experiment focused on testing the relationship between the number of sampling documents and the number of different words appearing in them. Figure 1 presents the results, in which the horizontal axis indicates the percentage of documents that were randomly selected with respect to the total documents in the resource, while the vertical axis indicates the percentage of words that were obtained from selected documents with respect to the total words used in the resource. At first, the number of words increased very fast, slowing down as about 5% of all documents were retrieved. However, the curves continued to increase when we obtained more and more documents. With 5% of the documents, we retrieved about 30% to 40% of the words; With 50% of the documents, we retrieved about 80% of the words for each of the three resources. Probably among the three resources, TREC-123 is the biggest of the three resources and the most heterogeneous, and its rate of increase was the lowest. On the contrary, WSJ is the smallest and the most homogeneous, and its rate of increase was the highest. However, the difference between the three cases is not large.

As we will see later, experiments showed that using 5% to 15% of all words in the queries is enough for our purpose. Therefore, using 0.5% to 1% of all documents is reasonable for document sampling, as about 10% to 20% of the words could be identified. In practice, as we do not know the exact size of the resource, several options can be used:

- If we have some knowledge of the resource, we can make a guess of the resource size, then decide how many document samples we need.
- Take a certain number (say, 40,000) as the threshold, and try to identify that number of words in the resource.
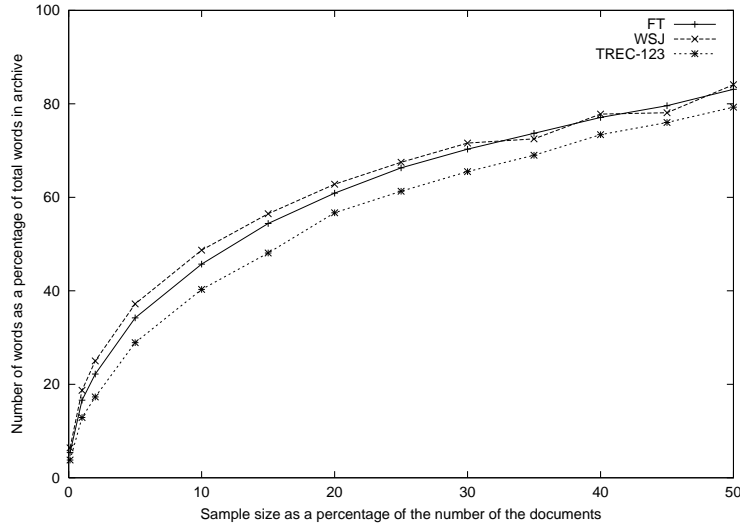
Figure 1. The relationship between documents and words.

— Specify a threshold t. Each time take a certain number of new documents. Check how many new words that these new documents introduce. If it is less than t, we stop the sampling process; otherwise, continue.

The second experiment was carried out to find an appropriate way of generating queries by checking the results of different queries. We generated queries with different numbers of words, from 2 to 100, then saw which one would produce the best result. We report the results from using two resources: FT and TREC-123. For FT, 2500 randomly selected documents (1.2%) were used for the sampling. Over 25,000 different words (Porter stems) were identified, which is about 11% of the total words in the resource. For TREC-123, 3200 documents (0.3%) were used for the sampling. Over 56,000 different words (Porter stems) were identified (about 5%). Then only a few (100-300) top frequency words were removed from the word collection in each case. As we only used a small percentage of all documents, low frequency words may not be really rare in the resource and therefore we did not remove any of them. Using the remaining words, we generated queries by randomly selecting a certain number of words from the collection. Each word could only be included in one query and every query had the same number of words. We submitted those queries to Lemur with a specified number of retrieved documents. The same number applied to all queries. Since the number of words was fixed, the less words in each query the more queries were generated. Tables 2 to 5 present the experimental results from resources FT and TREC-123, using the vector and okapi models in Lemur.

Each data item in Tables 2 to 5 represents the percentage (p) of different documents in the resource, which is achieved by all queries with n words and $p \times |D|$ documents retrieved, where $|D|$ is the size of the resource. n is given in

Number of words in query	Percentage of retrieved documents compared to total documents in resource							
	50%	100%	150%	200%	250%	300%	350%	400%
2	29.2	49.6	61.6	70.6	76.3	81.2	84.5	87.5
4	30.8	50.4	62.5	71.2	78.3	83.4	86.2	89.0
6	31.2	51.4	64.0	71.5	79.7	84.2	87.6	90.2
8	31.7	51.9	65.1	71.8	80.1	84.7	87.8	90.3
10	31.9	52.0	65.4	72.1	80.4	84.9	88.2	90.7
14	32.3	52.1	65.6	72.6	80.5	85.1	88.4	90.9
16	32.4	51.8	65.4	72.8	80.4	85.3	88.7	91.2
20	32.6	52.0	65.6	73.1	80.8	85.8	89.0	91.6
25	33.5	52.8	65.9	73.5	81.3	86.0	89.4	92.1
30	32.8	52.1	65.7	73.2	80.9	85.7	89.0	91.7
40	32.2	51.3	64.3	72.4	79.9	84.9	88.6	90.6
60	30.3	50.5	63.2	71.5	78.7	82.5	85.6	89.9
80	30.0	50.1	62.5	71.3	77.0	81.2	84.4	88.8
100	29.3	49.8	62.1	71.0	76.5	81.0	83.5	88.1

Table 2. Percentage of different documents retrieved from the resource compared to the total number of documents in the resource (vector space model, FT collection).

the first column, and p is given in the first row for that data item. The general tendency is that when we retrieved more and more documents, we obtained more and more different documents, while the percentage of overlaps increased as well. However, it seems that the percentage of overlapping documents was quite stable once a certain percentage of documents were retrieved. For example, when the number of retrieved documents by all queries was equal to the size of resource (100%), the number of different documents we obtained was around 50% of the total documents.

The experiment also shows that using between 10 and 40 words for each query produce better results. When the number of documents retrieved is 4 times the size of the resource, we obtain over 90% of different documents in most cases. The experiment also indicates differences between the two retrieval models (vector or okapi).

In addition to the experiment above, we used more sample documents and more words to generate queries. The experimental results do not show significant improvement over the above results. This suggests that using about 10% of the total words in the resource is enough for generating queries.

Also, in Figure 2, we present more detailed results for four specific situations. Each of these is composed of queries with 30 words, the number of retrieved documents ranging from 50% to 800% of resource size. Two resources TREC-123 and FT were used, as were the vector and okapi models from Lemur. These showed that in every case, when we retrieved more documents, we obtained more different documents. Over 99% of the documents could be detected when we retrieved a total of 650% of resource documents using the okapi model and

Number of words in query	Percentage of retrieved documents compared to total documents in resource							
	50%	100%	150%	200%	250%	300%	350%	400%
2	34.5	54.8	68.0	76.8	83.5	87.7	90.7	93.0
4	34.6	55.9	69.2	77.9	84.3	88.4	91.3	93.7
6	35.7	57.5	70.5	79.3	85.2	89.1	92.1	94.2
8	36.2	57.8	71.0	79.8	85.4	89.4	92.3	94.3
10	36.7	58.3	71.6	80.1	85.8	89.6	92.5	94.5
16	36.7	58.5	71.9	80.6	86.4	90.4	93.0	95.0
18	36.7	58.6	72.0	80.9	86.5	90.5	93.2	95.1
20	36.6	58.6	72.2	81.1	86.8	90.7	93.4	95.2
25	36.7	58.8	72.4	81.5	87.2	91.0	93.7	95.4
30	36.3	58.3	72.3	81.2	87.1	91.1	93.8	95.5
40	35.8	57.6	71.6	80.7	86.6	90.7	93.4	95.3
60	34.9	55.9	69.7	79.2	85.4	89.5	92.5	94.6
80	34.2	54.8	67.9	77.0	83.4	87.9	91.2	93.4
100	33.2	52.9	66.1	75.2	81.7	86.3	89.6	92.0

Table 3. Percentage of different documents retrieved from the resource compared to the total number of documents in the resource (okapi model, FT collection).

FT resource; the same level was obtained for the other three cases: 800%, okapi model and TREC-123; 900%, vector model and FT; 1000%, vector model and TREC-123.

4 Detecting the incremental change

Most document resources change their contents over time, and hence their sizes. As the above detection algorithm may require considerable time and resources, it would be desirable if we had a more efficient algorithm for detecting changes. In the following, we discuss an algorithm for detecting incremental change in resource size supposing we have already estimated the size of the resource before by using the algorithm in Section 2, and that a log, which records the queries and results returned from the resource, is available. The algorithm is as follows:

1. Randomly select a subset of queries S from all queries in the log.
2. Send out those S queries to the resource and collect the results.
3. For each query that has been selected, compare its old and new result list in reverse order (from the end to the beginning), identify the overlapping documents which appear in both lists. For each pair of overlapping documents, calculate the ratio of its position in the new list to that in the old list. Average all these ratios.
4. Average all the values obtained in Step 3 and take the final value as the ratio of the size change. If we multiply it by the previously detected resource size, we obtain the estimated size of the present resource.

Number of words in query	Percentage of retrieved documents compared to total documents in resource							
	50%	100%	150%	200%	250%	300%	350%	400%
2	28.9	49.0	62.0	71.7	78.3	83.4	86.9	89.7
6	35.1	56.9	70.5	79.1	84.9	88.9	91.6	93.5
10	36.9	58.8	72.3	80.9	86.4	90.0	92.5	94.3
14	37.4	59.4	72.7	81.2	89.8	90.2	92.6	94.3
16	37.7	59.6	73.1	81.4	86.8	90.3	82.7	94.4
20	37.8	59.9	73.2	81.5	86.8	90.4	92.8	94.4
25	37.9	60.0	73.4	81.8	87.1	90.6	92.9	94.5
30	37.3	59.2	72.7	81.2	86.8	90.5	93.0	94.7
40	38.1	60.1	73.4	81.7	87.0	90.5	92.8	94.4
50	36.6	58.2	71.6	80.3	86.1	90.0	92.6	94.4
60	36.5	57.8	71.2	79.9	85.7	89.6	92.3	94.1
70	36.3	57.6	70.8	79.4	85.1	89.1	91.8	93.7
80	36.2	57.1	70.3	78.9	84.6	88.6	91.4	93.4
90	35.9	56.7	69.7	78.3	84.1	88.1	90.9	93.0
100	35.6	56.3	69.2	77.7	83.5	87.5	90.5	92.5

Table 4. Percentage of different documents retrieved from the resource compared to the total number of documents in the resource (okapi model, TREC-123 collection).

One experiment was carried out with 2 resources: WSJ 1 (WSJ 87-90), and WSJ 2 (WSJ 87-92). These have 120,437 and 173,252 documents, respectively and the former is a subset of the latter resource. 10 queries containing 25 words were used in each test. 100 groups of tests were conducted. On average, the error rate $r = abs(|D_e| - |D_r|)/D_r$ was 2.8%, where $|D_e|$ is the estimated size of resource D, and $|D_r|$ is the real size of resource D.

Another experiment was carried out with 2 resources WSJ 3 (WSJ 87, 89, 91) and WSJ 4 (WSJ 88-92). This time, each of the two resources included some documents that the other did not have. This was designed to simulate the situations where some new documents may be added to the initial resource, and at the same time, some documents may be removed from the resource as well. 10 queries with 20 words were used for each test and 100 tests were carried out. On average, the error rate was 3.2%.

In the above experiments, change in document content was not considered. If many documents have had their contents changed, then the above algorithm may not work as effectively. We can use the following method to check if that is the case.

Once we have the two lists of overlapping documents from the above algorithm, we can compare their rankings by either Kendall Tau or Spearman correlation coefficients. Here only relative rankings of overlapping documents are considered. For example, if we have two lists l_1 and l_2 for a given query. $l_1 = (d_1, d_2, d_3, d_4, d_5, d_6)$, and $l_2 = (d_2, d_4, d_6, d_8, d_{10}, d_{12})$, then we consider the overlapping parts of l_1 and l_2. That is, $l_{1'} = (d_2, d_4, d_6)$, and $l_{2'} = (d_2, d_4, d_6)$, and the two lists correlate perfectly. If the two rankings correlate quite positively,

Number of words in query	Percentage of retrieved documents compared to total documents in resource							
	50%	100%	150%	200%	250%	300%	350%	400%
2	27.7	46.6	57.7	66.4	72.5	77.3	80.9	83.8
4	31.8	50.3	62.2	70.4	76.2	80.5	83.8	86.4
6	33.1	52.1	64.1	72.1	77.8	82.0	85.1	87.5
8	33.8	52.7	64.5	72.5	78.1	82.1	85.2	87.6
10	34.2	53.2	65.0	72.8	78.3	82.3	85.3	87.6
15	34.9	53.8	65.5	73.3	78.7	82.5	85.5	87.8
20	34.9	53.8	65.4	73.1	78.5	82.4	85.2	87.4
25	34.9	53.6	65.2	72.9	78.2	82.1	85.1	87.3
30	33.8	52.5	64.4	72.2	77.8	81.8	84.9	87.1
40	32.9	52.1	63.1	71.4	76.4	80.9	84.1	86.9
50	32.0	50.2	61.9	70.0	75.7	80.0	83.3	85.8
60	31.7	49.6	61.1	69.2	75.0	79.3	82.6	85.2
70	31.3	49.0	60.5	68.5	74.3	78.6	82.0	84.6
80	30.9	48.2	59.6	67.6	73.4	77.9	81.3	84.0
90	30.5	47.6	58.9	66.9	72.7	77.1	80.6	83.4
100	30.1	47.0	58.2	66.2	72.1	76.5	80.0	82.8

Table 5. Percentage of different documents retrieved from the resource compared to the total number of documents in the resource (vector space model, TREC-123 collection).

that suggests we can use the algorithm to estimate the incremental change; otherwise, the algorithm for detecting incremental change may not work properly. The reason could be any one or a combination of the following three reasons.

1. Either or both of the two documents has been changed in content;
2. The information retrieval model has been changed/updated;
3. The information retrieval model does not rank documents or does not use a deterministic algorithm to rank documents for a query.

Item 3 is beyond our consideration. Content change in documents may happen more often than changes or upgrades to the information retrieval model. However, for our purpose, it is not necessary to identify which situation has happened. When we find that the two rankings do not correlate very positively, it is an indication that the resource size should be estimated by the thorough estimation process discussed in Section 2.

5   Conclusion

In this paper, we have discussed a method of automatically detecting the size of a document resource. It is a two-phase process, including sampling documents to find a certain percentage of words used in the resource, and querying the information retrieval system with those words to detect its resource size. Also,
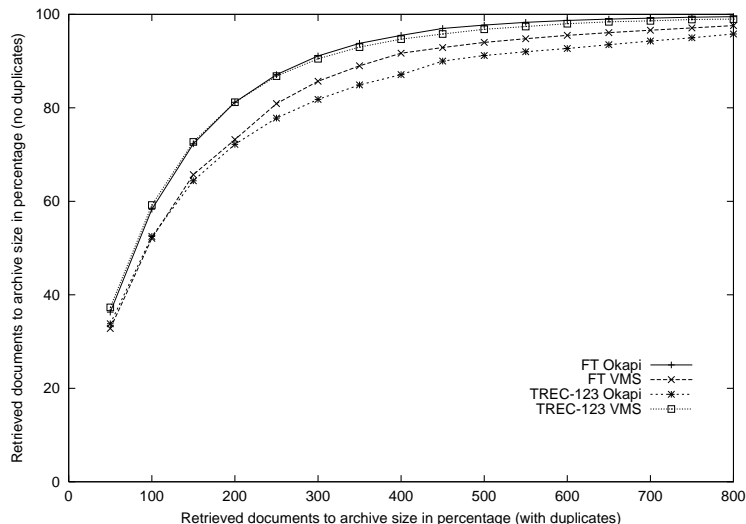
Figure 2. Performance of four groups of queries, each with 30 words.

an algorithm for detecting incremental change to a resource is proposed. Experiments have been carried out to test the effectiveness of the algorithm.

From those experiments with TREC data, we have the following observations:

- Sampling 1% of total documents randomly discovers over 10% of words used in a resource.
- Usually, using 5% to 10% of all words in the resource is adequate for making queries. Using more words does not bring better results.
- Each query should contains between 10 and 40 randomly selected words.
- In many cases, when the overlap rate from the results aggregated from all the queries reaches 75%, we retrieve over 90% of the total documents in the resource.

References

1. K. Bharat and A. Z. Broder. A technique for measuring the relative size and overlap of public web search engines. In *Proceedings of 7th WWW Conference*, Brisbane, Australia, April 1998.
2. J. K. Callan, M. Connell, and A. Du. Automatic discovery of language models for text databases. In *Proceedings of ACM SIGMOD International Conference*, pages 479–490, Philadelphia, USA, May 1999.
3. J. K. Callan, Z. Lu, and W. Croft. Searching distributed collections with inference networks. In *Proceedings of the 18th Annual International ACM SIGIR Conference*, pages 21–28, Seattle, USA, June 1995.
4. Norbert Fuhr. A decision-theoretic approach to database selection in networked ir. *ACM Transaction on Information Systems*, 17(3):229–249, 1999.

5. L. Gravano and H. García-Molina. Generalizing gloss to vector-space database and broker hierarchies. In *Proceedings of 21st VLDB Conference*, pages 78–89, Zűrich, Switzerland, 1995.

6. S. Lawrence and C. L. Giles. Searching the world wide web. *Science*, 280(3):98–100, April 1998.

7. K. Liu, C. Yu, and W. Meng. Discovering the representative of a search engine. In *Proceedings of ACM CIKM'02*, pages 652–654, Mclean, Virginia, USA, November 2002.

8. P. Ogilvie and J. Callan. Experiments using the lemur toolkit. In *Proceedings of the 2001 Text REtrieval Conference*, pages 103–108, Gaithersburg, Maryland, USA, November 2001.

9. S. Wu and F. Crestani. Multi-objective resource selection in distributed information retrieval. In *Proceedings of IPMU02, International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 1171–1178, Annecy, France, July 2002.

10. B. Yuwono and D. Lee. Server ranking for distributed test retrieval systems on the internet. In *Proceedings of the Fifth International Conference on Database Systems for Advanced Application*, pages 41–50, Melbourne, Australia, April 1997.