ELECTRONIC PUBLISHING, VOL. 1(2), 105–116 (SEPTEMBER 1988)

A search strategy for large document bases

DARIO LUCARELLA

Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
Via Moretto da Brescia 9, I-20133 Milano
Italy

SUMMARY

In this paper, we emphasize the need of modelling the inherent *uncertainty* associated with the information retrieval process. Within this context, a search strategy is proposed for locating documents which are *likely* to be relevant to a given query. A notion of closeness between document(s) and query is introduced and the implementation of an improved algorithm for the identification of the closest document set is presented with emphasis on computational efficiency.

KEY WORDS Information storage and retrieval Retrieval models Similarity computation Document access methods Search algorithms Search efficiency

INTRODUCTION

The increasing utilization of document preparation systems and document scanning techniques in combination with optical storage is moving towards end-users having large machine readable document collections. As a side effect of this impressive trend in *electronic publishing*, a crucial point becomes the need for sophisticated and innovative retrieval systems which provide satisfactory access to these growing textual databases[1,2].

Traditionally, logical design and implementation of document retrieval systems have been influenced by data base management technology. The result being that the *document* retrieval problem has been approached as merely a variant of *data* retrieval without taking into account that the logical foundations of the two processes are quite different[3].

Most of the commercially available systems rely upon Boolean logic and exact matching. In such environments, documents are represented by a list of significant terms without taking into account the different contribution of each term to the document characterization. A term either applies or does not apply to a document. The same is true for the query which consists of single search terms, denoting concepts, combined together via Boolean operators. In response, documents are retrieved when the attached index terms, regarded as secondary keys, match the query specification *exactly*. Thus the document collection is simply partitioned into two sets: documents that satisfy the query and documents that do not.

The considerable disadvantages of this approach, for document management applications, have been frequently pointed out[4,5].

Boolean formulations are difficult to generate by novices. Ordinary people asking for information have not an exact picture of what they are looking for so that it becomes difficult to formulate their queries in terms of a rigid and unnatural Boolean notation. Furthermore, it is difficult to get the right focus for the queries without a good knowledge

0894–3982/88/020105–12$06.00

Received 22 January 1988
Revised 30 April 1988

of the document base content. The use of general terms with poor filtering capabilities leads to the retrieval of a lot of items whereas the use of specific terms may be too selective yielding no documents in response.

In addition, all of the retrieved items are supposed to be of equivalent importance for the user. Obviously this is not realistic. Even if many factors influence in a complex way the users' judgment of relevance, the system should make an assessment as to whether or not a document is relevant to the users' needs and rank returned documents in order of their estimated usefulness.

All these limitations can be partially removed assuming correctly that, unlike data retrieval where the information is well structured and named, document retrieval is a process inherently *uncertain*. Search strategies must be conceived based on different retrieval models.

In the following, we recall briefly the research work concerning the attempt of modelling uncertainty in information retrieval and discuss the implications for system design. Within this context, the emphasis is on search strategies and the objective is to show that an efficient and effective retrieval system can be constructed using innovative techniques.

UNCERTAINTY MODELS

Any attempt of modelling the information retrieval process first requires a clear statement of what the process is.

Given a document collection, information retrieval systems are designed with the goal of providing, in response to a users' request, references to those documents that would contain the information desired by the user. In other words, the system operates indirectly: it does not answer the query retrieving the specific information required but provides instead references to a set of documents that are likely to contain the information that the user is interested in.

There are several sources of imprecision in this apparently simple process. First, the user of such a system is looking for information that (s)he does not actually know so (s)he expresses the information need in the form of a request for information. Such a request is further transformed in a request for documents. As a result, it is unlikely that the actual user needs can be exactly reflected and the submitted query will be only an imperfect expression of the information need.

A document is assumed to contain information, thus a method is required for extracting and representing the information contained in the documents. A representation of what the document is about is difficult to build and so, despite any sophisticated technique, we cannot expect it will be completely satisfactory. We have to assume it will provide only a partial and imprecise characterization of the document content.

In addition to the problem of documents and queries characterization, a crucial point is to establish a relationship between the users' request and the documents to determine whether a document is likely to be relevant or not. It should be recalled that a document must be considered relevant if, and only if, the document meets the actual users' needs. Hence relevance is a user dependent concept connected to a notion of *aboutness*. A document is relevant if it is about the information sought[6]. This means there is only a probabilistic relationship between the request and the likelihood that the user will be satisfied with that response. From a logical point of view, this selection process is essentially a decision under uncertainty.

With these premises, it is reasonable to assume the impossibility of building a perfect system in which *only* and *all* the relevant documents are retrieved.

This is the reason why there is a general consensus among researchers about the need that traditional information systems have to evolve in order to capture the vagueness and uncertainty which occur in reality.

Different approaches have been undertaken to cope with inexact representation of documents and queries and the consequent difficulties in determining the relevance of documents to a given users' request. They are based on fuzzy set concepts such as degree of membership, similarity relations and possibility distribution[7,8]. With the same objective, other researchers have approached the design of intelligent retrieval systems with a knowledge base and inferential capabilities to establish connections between a request and a set of documents[9]. Recently, VanRijsbergen has proposed a new framework to deal with the inherent uncertainty of the retrieval process. He proposes to use formal semantics for text representation and to formalize the relationship between a document and a request as a logical implication to which an uncertainty measure is attached[6].

A well known approach adopting fairly simple techniques for the design of a document retrieval system with the above goals is the Vector Space Model. It has been proposed and extensively investigated by Salton[4] and recently generalized by Wong and Raghaven[10]. According to this model, documents, identified by a set of attributes, are represented by vectors in n-dimensional space, each dimension denoting a concept. A particular document D_i is represented by the following vector $\mathbf{D}_i = (t_{i1}, \ldots t_{ij}, \ldots t_{in})$ where the j^{th} coordinate of the vector is a real number between zero and one which reflects the degree of membership, that is to say, the importance of the j^{th} term for qualifying the document D_i. The same applies to each query. It is represented by the vector $\mathbf{Q} = (q_1, \ldots q_j, \ldots q_n)$ where q_j denotes the weighted presence of the j^{th} term in the query Q.

In this setting, a retrieval operation involves the identification of those documents in the collection which exhibit a higher degree of resemblance with the submitted query. A similarity function S must be defined in order to measure the *closeness* between two given vectors. So, given Q, the function S produces for each document a real number $S(D_i,Q)$ in the range [0,1] with a high value implying a high degree of resemblance. As a consequence, it is possible to arrange and present the documents in decreasing order of similarity as measured by the S-values with respect to the query Q. A threshold can be defined for the S-values so that the trade-off can be controlled varying the threshold in proper way.

The adoption of this model could help to remove many of the problems mentioned in the introduction. There is no need for the user to specify Boolean interconnections between terms, it is easy to take into account weights reflecting the relative importance of different terms and the returned documents can be ranked in order of presumed relevance to the users' request.

In the following section, the main features of a prototype document retrieval system based on this model and implemented on personal workstation are overviewed[11].

SYSTEM FEATURES

The activities involved in the design and implementation of the system cover three main areas: document indexing, search strategies and access methods.

Indexing refers to the process of document characterization that requires the application of an automatic procedure to the full text or some surrogate of the document in order to identify index terms to be used in the vector representation.

In principle, an indexing method suited for content representation of natural language texts should be based on linguistic considerations. Unfortunately, linguistic analysis techniques are difficult to apply efficiently to large text samples and, furthermore, they require general knowledge about the subjects covered in the documents. On the other hand, complex indexing methods aimed at identifying syntactic components of the text have never been proved to be more effective in terms of retrieval performance[9,12].

A large number of studies on indexing strategies has been reported by Salton and the results of this work suggest that the simplest techniques are the most effective[12,13]. According to these suggestions, an automatic procedure can be designed to perform the content analysis on the text excerpt. Stop-words are skipped and word suffixes are removed to reduce words to common stems. Remaining terms are weighted and lowest weighted terms are dropped to reduce the size of the document representative.

In order to assign a weight to each term, several weighting schemes have been devised. Once again fairly simple techniques are sufficiently effective. A measure that has been recently tested with good results is the *augmented normalized term frequency*[14]. The weight w_{ij} which reflects the presumed importance of term t_j for qualifying the content of document D_i is defined as:

$$w_{ij} = \left(0.5 + 0.5\frac{F_{ij}}{Fmax_i}\right) \tag{1}$$

where F_{ij} represents the occurrence frequency of the term t_j in the document D_i normalized by $Fmax_i$, the maximum occurrence frequency among the terms associated to the vector $\mathbf{D}_i$. The effect of such a normalization being that w_{ij} is in the range $[0.5 \ldots 1.0]$ and longer documents do not produce higher term weights than shorter ones.

The same indexing and weighting process is performed on the natural language text of the query presented by the user. In this case the weight attached to each query term is determined by its *IDF (Inverse Document Frequency)* value. For each term j it is computed as $IDF_j = \log(N/N_j)$ where N is the number of documents in the collection and N_j is the number of documents in which the term j appears. Such weights express the value of a term for searching as they indicate how useful it is to differentiate the document from the collection.

During the matching process in which the two weights are combined, the effect is to combine the local occurrence of a term within a document, which reflects its *importance*, with the global occurrence of the same term within the collection, which reflects its *discrimination* power.

The retrieval procedure locates all documents relevant to the users' request and arranges them in decreasing order of similarity.

When the user enters the query facility, the system displays an appropriate template. The user optionally specifies some values for fixed attributes and a natural language statement describing in a rough way the topic which (s)he is interested in.

Documents retrieved are those that satisfy the combination of selection conditions, if specified, and whose topic resembles as much as possible the proposed one. The user is presented with a few of the top documents and is requested to mark the relevant ones among those displayed. In this way, a better picture of the user's concept of the relevant document is obtained and used to rerank documents.

Such a *relevance feedback* process allows a successive improvement of the query combining the result of the earlier search operation with the users' judgment. The effect is equivalent to moving the query in the concept space toward the set of relevant documents by enhancing the importance of terms that appear in the descriptors of those documents marked as relevant by the user.

So, given the original query $Q = (q_1, q_2, \ldots q_k)$, a new expanded query $Q' = (q_1, q_2, \ldots q_k, t_{k+1}, \ldots t_n)$ is prepared consisting of the initial terms plus additional terms coming from the descriptors of the marked documents. The weight of each term in the new query is computed combining the value it has in the original query, if present, to the values it has in the marked descriptors.

The document access method is based on inverted files. Despite processing and storage overheads for index management, the main advantage of an inverted file organization is the speed with which responses to users' queries are obtained. No search operation is carried out on the main file but closest documents are located by manipulating only the pointer lists in the directories.

Index term stems are found in the concept dictionary where pointers to the inverted lists are stored together with the total number of postings for each term. This last value is used to compute the *inverse document frequency* weights for query terms.

The postings list for each term t consists of pairs of values (D_i, w_i), D_i referencing the document header and w_i indicating the weight the term t has for the document D_i. The variable length inverted lists are stored in fixed length blocks in order to improve their manipulation and updating.

When the document is loaded the indexing procedure is applied to the text excerpt in order to identify significant terms. Such terms and the document reference are used respectively to update the concept dictionary and the postings file.

A crucial point with models based on similarity evaluation is the efficiency of the search procedure considering the high number of query-document comparisons that must be carried out in order to locate the closest set. The problem is approached in further detail in the next section whereas a range of search procedures based on this file organization can be found in[15].

THE SEARCH PROCEDURE

Within the previous framework, documents as well as queries can be regarded as points in $n-$dimensional space. So, the retrieval problem can be rephrased in the following way, often referred to as *nearest neighbour* searching[16]. Given a set N, of points D_i in $n-$space and a specific point Q, find the set R $(R \subset N)$ of points $D_1 \ldots D_r$ *closest* to Q. Closeness being measured by some appropriate function.

In the following, assume to compute the query-document closeness with the well tested cosine correlation[4]. Thus the similarity of a document D_i with respect to the query Q is given by the following equation:

$$S(D_i, Q) = \frac{\sum_{j=1}^{n} q_j t_{ij}}{\sqrt{\sum_{j=1}^{n} q_j^2 \; . \; \sum_{j=1}^{n} t_{ij}^2}} = \frac{\sum_{j=1}^{n} q_j t_{ij}}{L_Q \; . \; L_{Di}} \qquad (2)$$

L_Q and L_D being, respectively, the query and document vector lengths.

The straightforward way to obtain the relevant set is to compute the similarity function for each document in the collection to obtain r closest neighbours. It requires $O(N)$ computations which is impractical for large real collections.

So the problem is to organize the document set N so that subsequent searches for the closest points to a new point Q can be answered quickly. The objective is to eliminate from further consideration those documents which fail to comply with the submitted query as soon as possible.

One approach consists in preprocessing the collection partitioning it in clusters, each cluster consisting of similar documents and a representative of the included documents. The other approach followed here is based on the availability of a combined inverted file/document file as presented before. It has been proved by Voorhees[17] that such an organization is comparable or better in terms of effectiveness but it is more efficient.

The availability of the inverted file definitely reduces the number of documents that must be considered as potential candidates for the closest set. With reference to the cosine similarity function used here, $S(D_i,Q) \neq 0$ *if and only if* the query and the document vectors have at least one common term. It means that we have to take into account only those documents which appear at least once in the postings corresponding to the query terms while all other documents can be discarded.

A first algorithm that reaches this goal *minimizing* the number of accesses to the document file was proposed by Noreault *et al*[18]. The basic idea is to process the query lists allocating a counter to each encountered document and setting it to one. When a document appears once again in a subsequent list, its counter is incremented by one. The end result is to have in each counter the number of matching terms between the document and the query. If the terms have attached weights, as in our setting, the counters will be incremented by the product of such weights. The result being that the counters will contain the inner product between the document and query vectors. The pseudo-code for this algorithm is reported.

```
Procedure Search;
  for each QueryTerm $q_j$ do
    Read InvertedList; (* composed of couples ($D_i$, $w_i$) *)
    for each Document $D_i$ in the list do
      if NewDocument then
        AllocateCounter $C(D_i)$;
        $C(D_i)$ := 0;
      endif;
      $C(D_i)$ := $C(D_i)$ + ($q_j$ * $w_i$);
    endloop;
  endloop;
  for each Counter $C(D_i)$ do
    $C(D_i)$ := $C(D_i)$/($LQ$ * $LD_i$);  (* evaluation of S-function *)
  endloop;
  Sort $C(D_i)$ in decreasing order;
  Present the top $r$ documents;
end Search.
```

The above procedure computes the document-query similarity for each document that appears in the inverted lists and, consequently, has a non-zero value for the S-function. Nevertheless the set of involved documents may be considerably large. Perry and Willet, experimenting with several document collections, have shown in[15] there is a considerable number of documents having a very small similarity with the query. That implies the useless calculation of many low-value similarities for documents that will not reach the closest set.

Starting from these considerations, various algorithms have been proposed in order to refine and optimize the basic procedure discussed before. The common objective is the elimination of many of the query-document comparisons while still ensuring that the documents at the top of ranking are identified. A critical review of such algorithms is given by Perry and Willet in[15]. Previous work related to our approach has been reported by Harper[19] and Buckley[14]. The first algorithm optimized the number of documents to be maintained in internal storage whereas the second author worked at the inverted lists to be inspected.

A BOUNDED STRATEGY

The considerations developed before suggest two ways for optimizing the basic algorithm. The first improvement is oriented toward identifying the closest set manipulating *only* the inverted lists without making any access to the document file. In the above procedure, access is required to get the document length and this operation is carried out for each distinct document appearing in the lists.

Such a number can be high, according to the previous consideration, and, furthermore, the document file could be located on slow access devices like compact disks. The problem can be solved if we are able to allocate in the inverted lists all the information necessary to compute the similarity function. A slight modification to the term weighting scheme is required: instead of storing for each document in the list the pair (D_i,w_i), we have to store the weight normalized by the document vector length $(D_i,w_i/L_{Di})$. So given the document $D = (w_1/L_D, \dots w_j/L_D, \dots w_n/L_D)$ and the query $Q = (q_1/L_Q, \dots q_j/L_Q, \dots q_n/L_Q)$, the inner product between the two vectors $\sum_{j=1}^{n} w_j/L_D \; . \; q_j/L_Q$ is equivalent to the cosine correlation reported in (2).

This change eliminates in the previous algorithm the presence of the last loop and no access to the document file is required. The counters, at the end, will contain the inner product between the vectors which already corresponds to the final similarity between the associated documents and the query.

Further improvements can be achieved if we minimize both the number of documents to be evaluated and the number of inverted lists to be inspected. Particularly, with reference to the second point, if we find out, before completion, that we have already determined the closest set then the algorithm can be stopped and the remaining lists need not be examined.

As a first step, the query terms are sorted in order of decreasing weight. Since the term weight is determined by its *IDF* value, the effect is to have those terms which apply to few documents at the top. Consequently, we process the shorter inverted lists first leaving at the bottom the lists which include a larger number of documents.

Then we start to process the query lists in this order. Let us assume we have processed m query lists out of k and we have a current closest set R including the documents $D_1, \dots D_r$ in decreasing order of similarity.

We have to process now the $m+1^{st}$ query list. For each document D_i referenced in the list, an *upperbound* for the similarity value can be computed assuming D_i should happen to match all the remaining query terms. If the computed upperbound for D_i is less than the similarity value associated to D_r, it means the document D_i will never reach the R set so that it can be removed from further consideration.

Moreover, we can terminate the algorithm if the document D_{r+1}, the *first out* of the set R is not expected to give a similarity better than the document D_r, the *last in* the set R.

```
Procedure Search;
  sort QueryTerms in decreasing order of weight;
  repeat (* for each query term $q_j$ *)
    Read InvertedList; (* composed of couples $(D_i, w_i)$ *)
    for each Document $D_i$ in the list do
      if RelSetNotFull then
        Compute $C(D_i)$;
        Enter $D_i$ into the RelSet;
      else
        Compute $U(D_i)$;
        if $U(D_i) \leq C$(LastDocIn) then
          DoNotAllocate/Remove $D_i$
        else
          Compute $C(D_i)$;
          if $C(D_i) > C$(LastDocIn) then Enter $D_i$ into RelSet endif;
        endif;
      endif;
    endloop;
    Compute $U$(FirstDocOut);
  until LastQueryTerm or $U$(FirstDocOut) $\leq C$(LastDocIn)
  Present the RelSet sorted in decreasing order;
end Search.
```

A modification of the previous algorithm to include these improvements is reported. It is assumed that the weights reported in the lists are normalized and that the locations `FirstDocOut` and `LastDocIn` are maintained and updated properly when documents are entered respectively in the document table and in the set R.

As in the basic algorithm, the procedure `Compute C(D`$_i$`)` allocates the counter when the document is a new one and then computes the partial similarity as $C(D_i) := C(D_i) + (q_j * w_i)$.

Let us now examine the way to evaluate an upperbound $U(D_i)$ for the total query-document similarity, given a query Q consisting of $(q_1, \ldots q_j \ldots q_k)$ terms and given the document D_i.

After having processed the first m query lists, let s_i be the current score for the document D_i. At this point, we can compute an upperbound for $S(Q, D_i)$ as:

$$U(D_i) = s_i + \sum_{j=m+1}^{k} w_{ij} \ . \ q_j \tag{3}$$

assuming the worst case that all the remaining un-inspected $(k - m)$ terms are in common between the document and the query.

The summation appearing in equation (3) $\sum_{j=m+1}^{k} w_{ij} \ . \ q_j$ corresponds to the maximum possible remaining similarity. It can be easily determined if we consider that the weighting scheme in effect computes the document term weight as the intra-document normalized frequency, see equation (1). So the document term weight w_{ij} is bounded by 1.0. Equation (3) can be transformed in the following one that still gives an upperbound:

$$U(D_i) = s_i + \sum_{j=m+1}^{k} q_j \tag{4}$$

The result being that we can compute an upperbound, using equation (4), taking into account only the weights of the remaining query terms which are already known.

EVALUATION

With reference to the presented algorithm, a worst case analysis is not meaningful considering that the complexity is linear with the length of the inverted lists. So the worst condition means scanning the entire lists when the stopping comparison is not satisfied and the computing cycle is not terminated before completion. However, it must be recalled that the initial sorting on the query terms moves the longer inverted lists to the bottom. This means that even if the condition applies nearly at the end, the lists that will not be inspected for the stopping condition will be the most expensive.

Better results in terms of optimization can be obtained if we accept that only the top $r'(r' < r)$ best matching documents are returned while the remaining $(r - r')$ are simply *good* matches. This heuristic implies changing, in the previous algorithm, the stopping condition to $U(FirstDocOut) \leq C(D_{r'})$. Clearly, as $C(D_{r'}) > C(D_r)$, the stopping condition works better dropping a higher number of query lists. It has been shown under several test conditions that recall values do not degrade significantly as the number of guaranteed top documents r' decreases[14].

Table 1. Experimental Results

Number of documents	1500
Number of terms	2137
Number of queries	50
Average terms per document	14.8
Average terms per query	7.2
Average docs. *referenced* per query	354.5
Average docs. *processed* per query	78.3
Average dropped lists per query	0.27
Recall	0.39
Precision	0.82

For our purposes, it is interesting to compare, on average, the total number of documents which would have been processed without any optimization, with the number of documents really maintained in the table as a combined effect of the two introduced optimizations. Furthermore, it is interesting to take note of the mean number of dropped query lists considering that documents in such lists are not examined at all.

In order to make experiments and test the effectiveness of the algorithm, a document collection in the area of Applied Mathematics and Computer Science has been used. The collection was supplied in machine readable form by the Enel, Italian Electricity Board, and relevant characteristics are reported in the first part of the table.

A set of fifty queries in natural language has been used. The set has been indexed with query terms weighted by the *IDF* function. Each query was processed twice, first with the basic algorithm and afterward with the improved version. The returned relevant set consisted of ten good documents ($r = 10$) while only five documents ($r' = 5$) were guaranteed to be the best ones.

Results are reported on average in the second part of the table. The entry “documents referenced per query” gives the total number of distinct documents indexed by the query terms. The entry “documents processed per query” gives the actual number of documents placed into the table for which partial similarities are evaluated. The last row gives the percentage of inverted lists which are dropped as a consequence of the stopping condition. A value of 0.27 means that, on avarage, 27% of the inverted lists are not processed. Note that the dropped lists are the longest ones since the query terms are sorted in order of decreasing weight and the weighting scheme in effect assigns the lowest weights to terms which occur in many documents.

The conclusion was of a meaningful decrease in the number of documents to be processed as well as in the number of query lists to be examined.

Although the main goal of the reported experiment was to evaluate the *efficiency* of the search algorithm, a figure of the system *effectiveness* is reported in the third part of the table. The effectiveness represents the ability of the system to retrieve the relevant documents. It is usually expressed in terms of two standard performance measures *recall* and *precision*[4]. Recall gives the proportion of relevant items actually retrieved whereas precision gives the proportion of retrieved items actually relevant. Given A the set of relevant documents, B the set of retrieved documents and $|\,.\,|$ the counting measure giving the size of the set:

$$Recall : \frac{|A \cap B|}{|A|} \qquad Precision : \frac{|A \cap B|}{|B|}$$

Average values for such measures at the 10 documents cut-off level are reported in the last two rows. The apparently poor recall level must be related to the low exhaustivity of the sample query set (7.2 terms per query) and, essentially, to the cut-off level applied. Conversely the system precision is very high as a consequence of the weighting and ranking mechanism. In this respect, the effect of the introduced optimization is a possible decrease in the ordering precision among the retrieved documents since they are ranked in order of their partial similarity instead of the total similarity to the query.

CONCLUSION

In the previous pages, we have discussed the need of modelling uncertainty in the design of document retrieval systems. Consequent enhancements in terms of system capabilities and user interface have been underlined with reference to a prototype implementation. The system has been developed in Modula-2 on a personal workstation AT-386 under MS-DOS operating system[11].

It is based on the vector space model. An automatic indexing procedure with a weighting scheme to reflect term importance is applied to both document excerpts and natural language queries.

Documents are stored using an inverted file scheme and an efficient search strategy has been presented. The algorithm returns the relevant document set computing an upperbound on closeness thus obviating the need for an exact computation of closeness in many instances. The result is a strong improvement in both the number of documents to be evaluated and the number of inverted index entries to be inspected.

In a different setting, comparable results have been obtained with a slight modification of the same bounded strategy applied to the retrieval of mathematical formulae[20]. Additional experiments are planned to face other document collections available on optical disks[21].

As already remarked, some of the reported improvements assume a particular meaning in connection with large document databases and slow access optical storage. We can expect an increasing importance of such retrieval strategies in the future with the wide circulation of optical disks and the consequent availability to end users of large machine readable document collections.

ACKNOWLEDGEMENTS

This research activity has been partially supported by the Enel, Italian Electricity Board, and is also related to the larger research project *Scientific Communication Systems* sponsored by the Mathematical Committee of the Public Education Ministry.

REFERENCES

1. E. A. Fox, ‘Information retrieval: research into new capabilities’ in S. Lambert, S. Ropiequet, eds., *CD-ROM: the New Papyrus*, Microsoft Press, Redmond, 1986 pp.143–174.
2. D. Lucarella, ‘Design issues for an integrated documentation system’ in J. J. Miller, ed., *Text Processing Systems II*, Boole Press, Dublin, 1985 pp.34–46.
3. D. C. Blair, ‘The data-document distinction in information retrieval’ *Communications ACM* **27** 369–374 (1984).
4. G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, 1983.
5. P. Willet, ‘Ranked output searching in textual and structural databases’ *Proc. Ninth Online Information Meeting*, London, UK, 3–5 December 1985 pp.343–353.
6. C. J. VanRijsbergen, ‘A non-classical logic for information retrieval’, *Computer Journal* **29**, 481–485 (1986).
7. A. Bookstein, ‘Probability and fuzzy sets applications to information retrieval’ in M. E. William, ed., *Annual Review of Information Science and Technology*, **20**, Knowledge Industry Publications Inc., White Plans, 117–151 (1985).
8. B. P. Buckles and F. E. Perry, ‘Uncertainty models in information and data base systems’ *Journal of Information Science* **11**, 77–87 (1985).
9. W. B. Croft, ‘Approaches to intelligent information retrieval’, *Information Processing and Management* **23** (4), 249–254 (1987).
10. S. K. M. Wong *et al*., ‘On modeling of information retrieval concepts in vector spaces’, *ACM Trans. on Database Systems* **12**, 299–321 (1987).

11. D. Lucarella, ‘A document retrieval system based on nearest neighbour searching’, *Journal of Information Science* **14**, 25–33 (1988).
12. G. Salton, ‘Another look at automatic text retrieval systems’, *Communications ACM* **29**, 648–656 (1986).
13. G. Salton, ‘A blueprint for automatic indexing’, *ACM SIGIR Forum* **16**, 22–38 (1981).
14. C. Buckley and A. F. Lewit, ‘Optimization of inverted vector searches’, *Proc. Eighth International ACM Conference on Research and Development in Information Retrieval*, Montreal, Québec, 5–7 June, 1985 pp.97–110.
15. S. A. Perry and P. Willet, ‘A review of the use of inverted files for best match searching in information retrieval systems’, *Journal of Information Science* **6**, 59–66 (1983).
16. A. F. Smeaton and C. J. vanRijsbergen, ‘The nearest neighbour problem in information retrieval’, *Proc. Fourth International Conference on Information Storage and Retrieval*, Oakland, CA, 31 May–2 June, 1981 pp.83–87.
17. E. Voorhees, ‘The efficiency of inverted index and cluster search’, *Proc. ACM Conference on Research and Development in Information Retrieval*, Pisa, Italy, 8–10 September, 1986 pp.164–174.
18. T. Noreault, M. Koll and M. J. McGill, ‘Automatic ranked output from Boolean searches in SIRE’, *Journal Amer. Soc. Information Science* **28**, 294–304 (1977).
19. D. J. Harper, ‘Relevance Feedback in Document Retrieval Systems: An Evaluation of Probabilistic Strategies’ *Phd. Thesis* The University of Cambridge (1980).
20. D. Lucarella, ‘Retrieving mathematical formulae’ in J. Désarménien, ed., *TEX for Scientific Documentation* 120–130 Springer-Verlag, Berlin, (1986).
21. F. Compagnoni and D. Lucarella, ‘Design of an automated dictionary on CD-ROM: internal structure and retrieval facilities’ in J. J. Miller, ed., *Text Processing Systems III* 111–118 Boole Press, Dublin, (1986).