Information Retrieval Algorithms: A Survey

Prabhakar Raghavan[*]

Abstract

We give an overview of some algorithmic problems arising in the representation of text/image/multimedia objects in a form amenable to automated searching, and in conducting these searches efficiently. These operations are central to information retrieval and digital library systems.

1 Introduction.

The design and implementation of information retrieval systems is concerned with methods for storing, organizing and retrieving information from a collection of documents. The quality of such a system is measured by how useful it is to the typical users of the system. The rôle of the user in this process cannot be underestimated: for instance, a scholar seeking a research paper from a corpus of specialist articles is very different from a consumer seeking a good small car to buy, on the world-wide web. The availability of hypermedia has opened up new paradigms for seeking information: a user may do a simple keyword search, or may search through a hierarchical index of categories, or may follow hyperlinks through a sequence of related documents (commonly referred to as "browsing") — or may use some combination of these.

The source of the documents also important: a corpus of scholarly articles may contain a great deal of specialist information, but will generally enjoy a level of authorship and consistency of style not found in less structured corpora (such as the world-wide web, where the variegated content and style present serious obstacles to retrieval).

In this paper we focus on algorithmic issues arising in two aspects of information retrieval systems: (1) representing objects [1] in a form amenable to automated searching, and (2) efficiently searching such representations. We do not consider in detail algorithmic issues arising in the following other important aspects of information retrieval systems: (1) file structures and database population and maintenance; (2) computational linguistics; (3) user interfaces and user models; (4) network and distributed retrieval issues including server/network performance and load balancing; (5) security, access control and rights management.

In the remainder of this paper, we will refer to a collection of objects as a *corpus*. Documents are a particular type of object consisting of a sequence of terms; by a term we typically mean a word, but more generally a sequence of words that tend to occur together (e.g., "Massachusets Institute of Technology"). For a corpus consisting of text documents, a query is a set of terms.

We have noted above that the eventual test for an information retrieval system comes from user satisfaction. Can this be translated to something more quantitative that can be studied experimentally? What does this translate to in terms of the component algorithms?

Besides retrieval speed (and, less obvious to the user, database population or "preprocessing" speed), there are two common quantitative measures for traditional information retrieval systems. Consider a setting in which a user enters a query into the system, and receives a set of documents in return. The *precision* of the system on this query is the fraction of the returned documents that are relevant to the user's query. The *recall* of the system on this query is the fraction of relevant documents in the underlying corpus that are returned. A system can trivially provide perfect recall simply by returning all the documents in the corpus on every query; but this leads to poor precision. A wealth of experimental evidence suggests that in general, there is a tradeoff between the precision and the recall that a system can provide.

Some noteworthy issues: (1) in the above, "relevance" is thought of as a binary attribute: a document in the corpus is either relevant to the query, or it is not; (2) when reporting experimental research, it is common to average the precision and recall figures over a large number of queries; (3) in benchmark suites such as the TREC series [17], it is typical to provide a corpus and a set of queries; in addition there is a set of relevance judgements from an expert panel giving, for each query, the set of documents in the corpus deemed to be relevant to that query.

Clearly the operations we have chosen to con-

[*]IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose CA 95120.

[1]We will in general speak of "objects" that are understood to be text, images, video or some combination thereof as in multimedia documents; we will use the simpler term "documents" to mean text alone.

sider in this paper — structures for storing objects, and searching these structures — will influence system speed/scalability, precision and recall. In § 2 we review the most basic algorithmic operations found in virtually all retrieval systems. In § 3 we study a suite of algorithms for storing and retrieving objects in the general setting of *vector space models*. In § 5 we review the current state of art in the systems most commonly in use, especially for web search.

2 Basic algorithmic operations.

In this section we discuss some ideas that pervade most information retrieval systems. The first three — indexing, negative dictionaries (also known as "stop word lists") and stemming — exist in some form in virtually all systems. For the remainder of this section, we only discuss text retrieval from a corpus of documents.

Indexing is the most basic mechanism for preparing a corpus of documents for searching. In the simplest form the index stores, for every term, a list of all documents containing that term. Consider a query consisting of a set of search terms, stored (for instance) in alphabetical order. A simple lookup now retrieves all documents containing at least one of the search terms. However, this algorithmically expeditious approach leads to poor recall, as users typically do not launch a query consisting of a Boolean OR of search terms. Far more likely (especially in a large corpus such as the web) is a search for documents matching a Boolean AND of search terms. One answer would be to retrieve the set of documents matching each search term, and then take the intersection of the resulting sets. One problem with this approach: it may process many more documents than appear in the output. More generally, it may be desirable for the system to return a list of documents ranked by the number of search terms occurring in the document, or some variant thereof.

GOAL 1. *To devise efficient algorithms for (1) retrieving the set of documents containing every one of several search terms, and (2) for returning documents containing any of the search terms, ranked by the number of search terms that occur in the document.*

It should be stressed, though, that in designing a system the algorithm design is intertwined with the user interface, and with an understanding of the target user population. For instance, an algorithm that works well as the number of search terms goes to infinity is not much use if the user interface only allows the user to enter 3 search terms (or if the user population is one that seldom enters more than, say, 2 terms).

A more general type of index encodes not only all documents containing each search term, but also the position of each occurrence of the term in the document. This enables such queries as finding all occurrences of a string that is not already indexed as a search term (for instance, if "Massachusets Institute of Technology" were not indexed as a term, we could use such an index provided the terms "Massachusets", "Institute" and "Technology" were all indexed with their positions recorded). The retrieval system that currently appears to support the largest corpus for such queries is Digital's AltaVista.

GOAL 2. *Devise efficient algorithms and index designs for supporting more complex queries such as string queries.*

For an algorithm to succeed at such goals, it is essential to understand the underlying corpus. Do all documents use the same underlying vocabulary? What is the distribution of terms in the vocabulary? A commonly held thesis is that terms are distributed according to *Zipf's law* [26], by which the ith most frequently occurring term has probability proportional to $i^{-\alpha}$, where α is a positive constant; for the various indexing/searching goals mentioned here, how can we exploit Zipf's law? What fraction of the terms typically identify a document reasonably well? Are there many proper nouns? What terms should be handled through an index and what terms through other means?

These considerations lead to techniques that exploit the terms statistics in the corpus. A standard first step in this direction is the use of negative dictionaries (also known as *stop-word lists*) — a list of words that are not indexed because they occur too commonly in the corpus. Prepositions and articles are often included in such negative dictionaries; a typical negative dictionary in current practice might include several hundred terms. There are however tradeoffs in using this technique: while it improves precision on the vast majority of queries, it becomes very difficult to search for the string "to be or not to be". Further, the contents of a negative dictionary are highly dependent on the corpus — for instance, "can" is typically a stop-word, except perhaps in a corpus of documents on waste management and recycling. This issue looms large in searching using categories and clusters, as discussed below.

A related technique is *stemming:* it is common to reduce search and index terms to their etymological roots so that, for instance, a search for "dentistry" returns documents containing the term "dental". This is typically implemented using a set of rules derived from the grammatical structure of the language. In practice, these rules are far from perfect (it has been reported [5] for instance that documents from mathematics and anesthesiology were often confused because of the occurrence of "number" in both fields, with different meanings of course).

We have discussed above how retrieval systems may treat multiple search terms in various ways. In general, retrieval systems will assign a score to the degree to which a document in the corpus matches the query. This score may combine the number of times each search term occurs in the document, the frequency of occurrence in the document relative to the frequency in the corpus, and possibly the relative order in which the user typed in the search terms. The formula for such combinations is typically a mixture of statistical theory and experimental tuning. One idea [20] is to consider the *lexical affinity* of search terms: two search terms that are adjacent in the list of search terms, for instance, score higher in a document in which they occur close to each other (say, within five words).

One approach to improving precision in text retrieval is to use *categorized search:* each document is assigned to one or more *categories*. Typically, the categories are described by descriptive names; a user may then restrict their search to one of the categories. In the Yahoo! search directory for the web, for instance, the top-level categorization partitions documents into categories such as Arts, Business and Economy, Government, etc. Frequently (as in the case of Yahoo!), the categories are hierarchically organized: each category is further divided into sub-categories. One issue that arises is that in hierarchical categories, the importance of a term for searching depends on the position in the hierarchy. For instance, the term "computer" is useful in deciding whether a document should lie in the category "computer science"; within that category, though, it is essentially useless at distinguishing sub-categories and can be considered a stop-word.

How should the categories be chosen so as to improve the relevance of the documents returned by the search? To being with, the categories should be chosen in a manner that is intuitive to the anticipated user-population (rather than to the person designing the system). Next, the categories should be reasonably balanced: a categorization in which a small set of categories contains most of the documents is not likely to be as useful as a balanced taxonomy. Finally, the categories should "span" the corpus in the following sense: it is not very useful to have two categories that are very similar to each other (this actually makes it harder for the user to decide which category to select when searching).

A typical algorithmic operation in building and maintaining taxonomies is *clustering*. The idea is to partition the document into clusters of closely-related documents. One consequence is that the categorization need no longer be static (as above); instead, clustering partitions the documents into related clusters each of which may be described, perhaps, by the terms most frequently occurring in the cluster. (This may not always be the most intuitive representation from the standpoint of the user, but preliminary experience with the technique is encouraging [5].) Further, the clustering can change, as new documents are added (and old ones deleted); thus in a corpus of news articles, the clustering may change as the focus of the news changes. Such dynamic clustering, again, could be hierarchical [5]. We will explore this idea in greater detail in § 3 below.

The final basic idea we discuss is that of *relevance feedback*. Consider a user who enters a query into a document retrieval system. The system first returns a small number of matching documents; the user scans these, marking each document as "relevant" or "irrelevant". The system then uses this feedback from the user to formulate and launch a new query that better matches what the user is seeking. This is an interactive process, and intuitively leads to increasing precision and recall. A possible implementation might be to extract the terms most commonly occurring in the documents marked "relevant" by the user, and add them to the set of search terms in the query. Other, more sophisticated implementations come from techniques in § 3 below.

3 Vector space models.

Many of the text document search operations in § 2 are conveniently performed using the *vector-space representation* of documents. The added advantage of this representation is that it can be extended to more general objects that include, for instance, images and video.

We begin with a simple vector-space representation of m text documents in a corpus. Let n denote the number of terms in the vocabulary. Consider an $n \times m$ matrix A, whose entry a_{ij} indicates the presence or absence of the term i in document j. The entry is 0 if the term does not occur in the corresponding document, and some non-zero value if it does. A great deal of theoretical and empirical attention has been devoted to studying the non-zero value that makes up this entry: in the simplest form it is 1 whenever the term occurs at least once. More generally, the frequency of occurrence (or some function of the frequency) may be used. In more sophisticated versions, the relative frequencies of the term in the document and in the corpus may be combined in some form to determine a_{ij}; such combinations are sensitive to changes in the corpus (addition/deletion of documents). Ideally, one should not have to update the entire representation frequently, as the corpus changes. Each document thus becomes a point in n-dimensional space. A query can also be thought of as a point in n-dimensional space, so that retrieval becomes a matter of finding the document points in the space that are closest to the query The number of retrieved points varies, but is typically a quantity a user can handle at one time (say, in the neighborhood of 8 to 16 documents). Considerable effort then goes into finding the "right" distance function in this vector space.

The vector-space approach is generally attributed to Luhn, and was popularized by Salton in the Cornell SMART system [3]. The number of dimensions in the representation could be tens of thousands, which makes it challenging to perform the near-neighbor computation mentioned above. At the time of this writing most systems will, at query time, compute the distance from the query to every document, and then output the closest documents.

GOAL 3. *To find a near(est)-neighbor search procedure in "very high" dimensions that requires (in the worst case) fewer than n distance computations.*

A number of solutions have been proposed for this problem in the literature [25], but none can get below n comparisons in the worst case; in fact, we know of no method that uses fewer than n comparisons even when the inputs are drawn from an arbitrary probability distribution. Some empirical success has been observed in experiments by Hafner and Upfal [16] with the QBIC system. A robust, distribution-independent probabilistic analysis would be a good first step.

GOAL 4. *To find a near(est)-neighbor search procedure in "very high" dimensions that requires an expected number of comparisons that is smaller than n, when the points and queries are drawn from an arbitrary but fixed probability distribution.*

Notice that several of the operations discussed in § 2, e.g., indexing, clustering and relevance feedback all have natural geometric interpretations in the vector-space model. Consider for instance relevance feedback: the user's feedback may be used as follows. Let q denote the initial query vector. Let D denote the subset of the retrieved documents that the user marked as relevant, and D' the subset marked as irrelevant. Then relevance feedback results in a new query q' given by

$$q' = q + C \cdot [\sum_{d \in D} d - \sum_{d \in D'} d],$$

where all the sums are n-dimensional vector sums, and C is a well-chosen "weighting" constant.

Clustering and categorization, too, have natural geometric interpretations: we are clustering points in n-space. The difficulty in practice, though, is finding the right criteria for clustering — if two documents are "close" to each other from the user's standpoint, what criterion for clustering keeps them together in automated clustering? There has been considerable discussion of this in the literature [5, 23], and the answers are not very clear. This is related to the issue of the "right" distance measure for measuring proximity between points in the vector space, but automated clustering raises additional issues. How does one ensure that the clusters remain reasonably balanced?

GOAL 5. *To devise generic procedures for clustering documents in a vector space with efficient implementations.*

Experience shows that automated categorization is very corpus-dependent. Chekuri *et al.* [4], for instance, report experiments on automatically categorizing web documents according to the broad categories at the top-level of Yahoo!. They note that the results depend substantially on the "proximity" of the categories themselves — for instance, documents from "Arts" were more likely to be misclassified under "Entertainment" than with other pairs of categories, due to the heavy overlap of these categories. Further, they note, the wide variety of styles and quality of authorship on the web can easily confuse a categorizer. On the web it is a common practice to insert spurious terms (often in large numbers) into a document, with the intent of (mis)leading search engines to point to the document when given commonly-asked queries (these are typically commercial sites). Issues such as these show that the web is very different from the scholarly and journalistic corpora that have been the primary domain for experimental information retrieval. At a pragmatic level, it appears that categorizing the web will require very powerful tools and perhaps substantial human intervention (the Yahoo! directory is currently created manually by a team of ontologists; the InfoSeek directory, on the other hand, is reported to be automatically generated).

3.1 Representing multimedia objects.

A major advantage of the vector-space representation is that it is not specific to text documents; in fact it is used in virtually all current multimedia retrieval systems. The main idea is that instead of terms, we now represent the presence/absence of a set of features that are extracted from the object. In an image one might, for instance, record for each primary color the average density of that color; each color might become an axis in the space. More sophisticated versions of this are used in the *Query By Image Content* (or *QBIC*) system [12], and the related Photobook system [22]. These systems allow the user to specify an image query using a variety of query interfaces, and the system returns objects that – to a human user – appear to be close to the query. Image similarity systems such as QBIC try to match by image color composition, texture and shape. For instance, the user may specify a color mixture as a histogram, and search for a striped shirt from a fashion catalog. The crucial words in the notion of similarity are "appear to be" and "to a human": human perception will not in general conform to a clean mathematical notion of proximity (say, the difference between RGB color vectors averaged over an image). Indeed, experience with the QBIC system [12] has shown considerable variation between simple machine representations and human perception in the color component alone. Thus the choice of the distance measure (in the vector space representation) is especially difficult and important in such systems. The solution (in practice) is to fine-tune the system (typically, as "bad examples" are noticed) until the distance function appears to be right most of the time. Given the heuristic nature of this process, it is not surprising that the resulting distance function is (1) computationally non-trivial, and (2) difficult to capture in a mathematically clean form. For instance, the function will not in general obey the triangle inequality; moreover, if the system allows multiple query interfaces (say, search by color versus search by shapes) there may be multiple distance functions involved. The number of dimensions in these applications may range in the several hundreds. The lack of the triangle inequality further complicates Goals 3 and 4 above. For one of the distance functions in the QBIC system, Fagin and Stockmeyer [9] have established a *semi-triangle inequality:* given any triangle in the space, the sum of any two sides is shown to be at least $(1 - \epsilon)$ times the third. The following goal is analogous to Goals 3 and 4 above.

GOAL 6. *To devise efficient near-neighbor search algorithms given the semi-triangle inequality.*

In the multimedia search systems currently being developed an object may consist, for instance, of a set of images and some text. A simple approach to implementing such systems is to rely on text annotation. For instance, AltaVista can search for a text string appearing in the caption of an html "image" tag. This approach works provided images are well-annotated; if the caption is misplaced, or does not contain the keywords the user enters, it fails.

In the more sophisticated systems in development, the user may use any combination of text/image/video search. The system must then combine the results of the search by each of the query media. Fagin [10] gives some algorithms for combining such search results according a fixed combination rule. More generally, the user may *at query time* give a weighting with which to combine the results of such searches. In the vector-space representation, this may be thought of as near-neighbor searches in which the distance function is derived from a convex combination of two vector spaces, where the weights in the convex combination are given at query time. We know of no such work even in simple geometric settings; it appears that for the very simple case of the Euclidean plane (say, where the x-axis can be "stretched" or "compressed" at query time), the obvious approach of representing the problem in three dimensions gives a solution (probably not a very practical one).

GOAL 7. *To devise algorithms for near-neighbor computations in which the components of the distance function have weights given at query time.*

3.2 Algebraic methods.

Given the vector space representation of objects, it is natural to consider the use of algebraic methods to facilitate retrieval. An intriguing technique known as *Latent Semantic Indexing* does just this. We revert for a moment to the discussion of text retrieval, although all our comments here can be generalized to vector space representations of arbitrary objects.

The main idea is that vectors representing the documents are projected down to a new, low-dimensional space obtained by the *singular value decomposition* of the term-document matrix A. This low-dimensional space is spanned by the eigenvectors of $A^T A$ that correspond to the few largest eigenvalues — and thus, presumably, to the few most striking correlations between terms. Queries are also projected and processed in this low-dimensional space. This results not only in great savings in storage and query time (at the expense of some considerable preprocessing), but also, according to empirical evidence reported in the literature, to *improved information retrieval* [1, 7, 8].

Let A be an $n \times m$ matrix as before; let the rank of A be r. Let the singular values of A be $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r$ (not necessarily distinct), i.e., $\sigma_1^2, \sigma_2^2, \ldots \sigma_r^2$ are the eigenvalues of AA^T. The *singular value decomposition* of A decomposes it into the product of three matrices $A = UDV^T$, where $D = \text{diag}(\sigma_1, \ldots, \sigma_r)$ is an $r \times r$ matrix, $U = (u_1, \ldots, u_r)$ is an $n \times r$ matrix whose columns are orthonormal, and $V = (v_1, \ldots, v_r)$ is an $m \times r$ matrix which is also column-orthonormal.

LSI works by omitting all but the k largest singular values in the above decomposition, for some appropriate k (here k is the dimension of the low-dimensional space alluded to in the informal description above). It should be small enough to enable fast retrieval, and large enough to adequately capture the structure of the corpus. Let $D_k = diag(\lambda_1, \ldots, \lambda_k)$, $U_k = (u_1, \ldots, u_k)$ and $V_k = (v_1, \ldots, v_k)$. Then

$$A_k = U_k D_k V_k^T$$

is a matrix of rank k, which is our approximation of A. The rows of $V_k D_k$ above are then used to represent the documents.

How good is this approximation? The following well-known theorem gives us some idea.

THEOREM 3.1. *(Eckart and Young, see [15].) Among all $n \times m$ matrices C of rank at most k, A_k is the one that minimizes $||A - C||^2 = \sum_{i,j}(a_{ij} - c_{ij})^2$.*

Therefore, LSI preserves (to the extent possible) the relative distances (and hence, presumably, the retrieval capabilities) in the term-document matrix while projecting it to a lower-dimensional space. However, the literature [1, 6, 7, 8] goes beyond suggesting LSI as merely a dimension-reduction technique: in fact, the reported experience is that LSI tends to bring together documents that are *semantically* related. Thus, a search for "car" might retrieve documents containing the term "automobile", even if they did not contain the term "car". Despite the wealth of empirical evidence supporting such improved semantic retrieval from LSI, Theorem 3.1 above only suggests that LSI doesn't do serious damage during dimension reduction. Can one theoretically explain the observed improved retrieval of LSI?

In recent work, Papadimitriou *et al.* [21] show that under a probabilistic model for document generation, LSI will cluster documents by topic. They also point out that if distance preservation during dimension reduction were the only criterion, then projecting the documents down to a random subspace gives a good approximation, and suggest performing LSI after dimension reduction by random projection. This builds on theorems on random subspace projection due to Johnson and Lindenstrauss [14, 18].

One deterrent to the widespread proliferation of LSI in commercial systems has been the computational cost of computing the singular value decomposition on a large corpus (such as the web). A number of attempts have been made to circumvent this bottleneck. The published literature suggests that it is currently difficult to compute the SVD for about 100,000 documents, and virtually impossible to do this for a million documents. Intriguingly, though, the web search engine Excite reportedly uses a variant of LSI, and apparently indexes tens of millions of documents. At least in theory [21], random projection speeds up the subsequent LSI computation; in practice, though, random projection may reduce the sparseness of the document matrix to the point where LSI code for sparse matrices cannot be used.

GOAL 8. *To devise a computationally effective variant of LSI.*

4 A computational meta-question

The above sections focused on a number of ways of speeding up existing information retrieval systems through improved algorithms. However, there is a compelling argument that users of current web search engines might be willing to sacrifice some of the retrieval speed for the sake of better relevance in what they retrieve. In an extreme version of this view, one might argue that the bottleneck in information retrieval currently is not computational, but rather that of better determining the relevance of a document to a query. This prompts the following "ultimate" question, which is admittedly more of a thought experiment than a system waiting to be built:

GOAL 9. *Given "infinite" computational power, can one improve the accuracy to which we compute the relevance of a document to a query?*

A positive answer to this question would shift the bottleneck in information retrieval from relevance judgements and human factors (which are intangible to algorithms researchers), to algorithms and computation.

5 Current work and further resources.

In this section we give an overview of resources for finding out about current work on information retrieval. Two classic texts on LSI are the books by van Rijsbergen [23] and by Salton [24]. Further detail on algorithmic issues can be found in the volume edited by Frakes and Baeza-Yates [13]. The survey of information retrieval by Faloutsos and Oard [11] is fairly recent; Lesk [19] gives a broad historical perspective on the evolution of the field.

We next provide an annotated list of information retrieval URLs on the web; these links are active at the time of this writing. Many of the articles/books listed below may be accessed through these URLs.

A good starting point is the home page for ACM's special interest group on information retrieval:

`http://info.sigir.acm.org/sigir/`

A comprehensive bibliography may be found in (the URL has been broken into two lines here to fit the column width) `http://www.sils.umich.edu/~mjpinto/ILS609Page/Bibliography/IRBibliography.html`

A number of pages on the web offer pointers to information retrieval efforts around the world, including multimedia search. Examples include:

`http://ruff.cs.umbc.edu:4080/IR.html`
`http://www.cs.jhu.edu/~weiss/glossary.html`
`http://www-ir.inf.ethz.ch/ir_groups.html`

More resources may be found (with some difficulty!) using web search engines. The QBIC project may be found at:

`http://wwwqbic.almaden.ibm.com/~qbic/qbic.html`

Finally, we mention an announcement of the ConText system by Oracle corporation, which suggests that powerful new methods in computational linguistics are being used to build a system purported to give significantly improved retrieval performance:

`http://www.oracle.com`

6 Acknowledgements.

I thank Rakesh Agrawal, Soumen Chakrabarti, Chandra Chekuri, Byron Dom, Michael Goldwasser, David Karger, Daphne Koller, Nimrod Megiddo, Hisao Tamaki, Eli Upfal, Santosh Vempala and Mark Wegman for ongoing collaboration and many insightful discussions on various aspects of information retrieval. Special thanks to Christos Papadimitriou for teaching me a great dela about information retrieval, and for many hours of interesting discussions.

References

[1] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. SIAM Review, 37(4), 1995, 573-595, 1995.

[2] M. W. Berry, T. Do, G. W. O'Brien, V. Krishna, and S. Varadhan. SVDPACKC (Version 1.0) User's Guide. University of Tennessee, April 1993.

[3] C. Buckley, A. Singhal, M. Mitra, G. Salton. New Retrieval Approaches Using SMART: TREC 4. Proceedings of the Fourth Text Retrieval Conference, National Institute of Standards and Technology, 1995.

[4] C. Chekuri, M. Goldwasser, P. Raghavan and E. Upfal. Automated categorization on world-wide web documents. Unpublished manuscript, 1996.

[5] D. R. Cutting, J. O. Pedersen, D. R. Karger and J. W. Tukey. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. Proceedings of ACM SIGIR, 318-329, 1992.

[6] S. Deerwester, S. T. Dumais, T.K. Landauer, G.W. Furnas, and R.A. Harshman. Indexing by latent semantic analysis. Journal of the Society for Information Science, 41(6), 391-407, 1990.

[7] S.T. Dumais, G.W. Furnas, T.K. Landauer and S. Deerwester. Using latent semantic analysis to improve information retrieval. In Proceedings of CHI'88: Conference on Human Factors in Computing, New York: ACM, 281-285, 1988.

[8] S.T. Dumais. Improving the retrieval of information from external sources. Behavior Research Methods, Instruments and Computers, 23(2), 229-236, 1991.

[9] R. Fagin and L. Stockmeyer. Relaxing the triangle inequality in pattern matching. IBM Research Report RJ 10031, June 1996.

[10] R. Fagin. Combining fuzzy information from multiple systems. Proceedings of the 15th ACM Symp. on Principles of Database Systems, Montreal, 1996, pp. 216-226.

[11] C. Faloutsos and D. W. Oard. A Survey of Information Retrieval and Filtering Methods. Dept. of Computer Science, Univ. of Maryland, August 1995.

[12] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele and P. Yanker. Query by image and video content: The QBIC system. IEEE Computer, **28**, 23–32, 1995.

[13] W. Frakes and R. Baeza-Yates, editors. Information Retrieval: Data Structures and Algorithms. Prentice-Hall, 1992.

[14] P. Frankl and H. Maehara. The Johnson-Lindenstrauss Lemma and the Sphericity of some graphs, J. Comb. Theory B **44** (1988), 355-362.

[15] G. Golub and C. Reinsch. Handbook for matrix computation II, Linear Algebra. Springer-Verlag, New York, 1971.

[16] J. Hafner and E. Upfal. Nearest neighbors in high dimensions using random sampling: theory and experiments. Unpublished manuscript, 1996.

[17] D. Harman. Overview of the Fourth Text REtrieval Conference (TREC-4). Proceedings of the Fourth Text Retrieval Conference, National Institute of Standards and Technology, 1995.

[18] W. B. Johnson and J. Lindenstrauss. Extensions of Lipshitz mapping into Hilbert space, Contemp. Math. **26** (1984), 189–206.

[19] M. Lesk. The Seven Ages of Information Retrieval. Proceedings of the Conference for the 50th anniversary of As We May Think, 12-14, 1995. Available at the time of this writing as

`http://community.bellcore.com/lesk/ages/ages.html`

[20] Y. Maarek and F. Smadja. Full text indexing based on lexical relations. an application: Software libraries. In Proceedings of SIGIR'89, N.Belkin and C.van Rijsbergen, Eds., ACM Press, 198–206, 1989.

[21] C. H. Papadimitriou, P. Raghavan, H. Tamaki and S. Vempala. Latent Semantic Indexing: a probabilistic

analysis. Unpublished manuscript.

[22] A. Pentland, R. W. Picard and S. Sclaroff. Photobook: Tools for Content-Based Manipulation of Image Databases, Proc. Storage adn Retrieval for Image and Video Databases II, **2**, SPIE, 34–47.

[23] C. J. van Rijsbergen. Information Retrieval. Butterworths, London 1979.

[24] G. Salton. Automatic Text Processing. Reading, MA: Addison Wesley, 1989.

[25] H. Samet. The design and analysis of spatial data structures. Addison-Wesley, 1989.

[26] G.K. Zipf. Human Behavior and the Principle of Least Effort: an Introduction to Human Ecology. Addison-Wesley, 1949.