INFORMATION RETRIEVAL ON THE WORLD WIDE WEB

Effective search and retrieval are enabling technologies for realizing the full potential of the Web. The authors examine relevant issues, including methods for representing document content. They also compare available search tools and suggest methods for improving retrieval effectiveness.

Venkat N. Gudivada
Dow Jones Markets
Vijay V. Raghavan
University of Southwestern Louisiana
William I. Grosky
Wayne State University
Rajesh Kasanagottu
University of Missouri

The World Wide Web is a very large distributed digital information space. From its origins in 1991 as an organization-wide collaborative environment at CERN for sharing research documents in nuclear physics, the Web has grown to encompass diverse information resources: personal home pages; online digital libraries; virtual museums; product and service catalogs; government information for public dissemination; research publications; and Gopher, FTP, Usenet news, and mail servers. Some estimates suggest that the Web currently includes about 150 million pages and that this number doubles every four months.

The ability to search and retrieve information from the Web efficiently and effectively is an enabling technology for realizing its full potential. With powerful workstations and parallel processing technology, efficiency is not a bottleneck. In fact, some existing search tools sift through gigabyte-size precompiled Web indexes in a fraction of a second. But retrieval effectiveness is a different matter. Current search tools retrieve too many documents, of which only a small fraction are relevant to the user query. Furthermore, the most relevant documents do not necessarily appear at the top of the query output order.

Few details concerning system architectures, retrieval models, and query-execution strategies are available for commercial search tools. The cause of preserving proprietary information has promulgated the view that developing Web search tools is esoteric rather than rational. In this article, we hope to promote innovative research and development in this area by offering a systematic perspective on the progress and challenges in searching the Web.

1089-7801/97/$10.00 ©1997 IEEE

We begin with a brief discussion of navigation strategies for searching the Web, followed by a review of methods for representing the information content of Web documents and models for retrieving it. We then classify, describe, and compare current search tools and services, and conclude by examining some techniques for improving their retrieval effectiveness.

TRAVERSING THE WEB

One way to find relevant documents on the Web is to launch a Web robot (also called a wanderer, worm, walker, spider, or knowbot). These software programs receive a user query, then systematically explore the Web to locate documents, evaluate their relevance, and return a rank-ordered list of documents to the user. The vastness and exponential growth of the Web make this approach impractical for every user query.

An alternative is to search a precompiled index built and updated periodically by Web robots. The index is a searchable archive that gives reference pointers to Web documents. This is obviously more practical, and many existing search tools are based on this approach.

Generating a comprehensive index requires systematic traversal of the Web to locate all documents. The Web's structure is similar to that of a directed graph, so it can be traversed using graph-traversal algorithms. Because Web servers and clients use the client-server paradigm to communicate, it is possible for a robot executing on a single computer to traverse the entire Web. There are currently three traversal methods:

- Providing the robot a "seed URL" to initiate exploration. The robot indexes the seed document, extracts URLs pointing to other documents, then examines each of these URLs recursively in a breadth-first or depth-first fashion.
- Starting with a set of URLs determined on the basis of a Web site's popularity and searching recursively. Intuitively, we can expect a popular site's home page to contain URLs that point to the most frequently sought information on the local and other Web servers.
- Partitioning the Web space based on Internet names or country codes and assigning one or more robots to explore the space exhaustively. This method is more widely used than the first two.

The frequency of Web traversal is another design variable for Web robots with important implications for the currency and completeness of the index.

INDEXING WEB DOCUMENTS

We can view effective Web searches as an information retrieval problem.[1,2] IR problems are characterized by a collection of documents and a set of users who perform queries on the collection to find a particular subset of it. This differs from database problems, for example, where the search and retrieval terms are precisely structured. In the IR context, *indexing* is the process of developing a document representation by assigning content descriptors or terms to the document. These terms are used in assessing the relevance of a document to a user query. They contribute directly to the retrieval effectiveness of an IR system.

IR systems include two types of terms: objective and nonobjective. *Objective terms* are extrinsic to semantic content, and there is generally no disagreement about how to assign them. Examples include author name, document URL, and date of publication. *Nonobjective terms*, on the other hand, are intended to reflect the information manifested in the document, and there is no agreement about the choice or degree of applicability of these terms. Thus, they are also known as *content terms*. Indexing in general is concerned with assigning nonobjective terms to documents. The assignment may optionally include a weight indicating the extent to which the term represents or reflects the information content.

The effectiveness of an indexing system is controlled by two main parameters. *Indexing exhaustivity* reflects the degree to which all the subject matter manifested in a document is actually recognized by the indexing system. When the indexing system is exhaustive, it generates a large number of terms to reflect all aspects of the subject matter present in the document; when it is nonexhaustive, it generates fewer terms, corresponding to the major subjects in the document. *Term specificity* refers to the breadth of the terms used for indexing.[2] Broad terms retrieve many useful documents along with a significant number of irrelevant ones; narrow terms retrieve fewer documents and may miss some relevant items.

The effect of indexing exhaustivity and term specificity on retrieval effectiveness can be explained by two parameters used for many years in IR problems:

- *Recall* is the ratio of the number of relevant documents retrieved to the total number of relevant documents in the collection.
- *Precision* is the ratio of the number of relevant documents retrieved to the total number of documents retrieved.

Ideally, you would like to achieve both high recall and high precision. In reality, you must strike a compromise. Indexing terms that are specific yields higher precision at the expense of recall. Indexing terms that are broad yields higher recall at the cost of precision. For this reason, an IR system's effectiveness is measured by the precision parameter at various recall levels.

Indexing can be performed either manually or automatically. The sheer size of the Web together with the diversity of subject matter make manual indexing impractical. Automatic indexing does not require the tightly controlled vocabularies that manual indexers use, and it offers the potential to represent many more aspects of a document than manual indexing can. However, it also remains at a primitive level of development, despite many years of study. (For details on current ways to automatically assign content terms to documents, see the sidebar below.)

INFORMATION RETRIEVAL MODELS

An IR model is characterized by four parameters:

- representations for documents and queries,
- matching strategies for assessing the relevance of documents to a user query,
- methods for ranking query output, and
- mechanisms for acquiring user-relevance feedback.

IR models can be classed into four types: set theoretic, algebraic, probabilistic, and hybrid models. In the following sections, we describe instances of each type in the context of the IR model parameters.

AUTOMATIC INDEXING METHODS

The automatic assigning of content terms to documents can be based on single or multiple terms.

Single-Term Indexing

The *term set* of the document includes its set of words and their frequency. Words that perform strictly grammatical functions are compiled into a *stop list* and removed. The term set can also be refined by *stemming* to remove word suffixes.

Approaches to assigning weights for single terms may be grouped into the following categories: statistical, information-theoretic, and probabilistic. While the first two categories just use document and collection properties, the probabilistic approaches require user input in terms of relevance judgments.

Statistical methods. Assume that we have N documents in a collection. Let tf_{ij} denote the term frequency, which is a function of the frequency of the term T_j in document D_i.

Indexing based on *term frequency* fulfills one indexing aim, namely, recall. However, terms that have concentration in a few documents of a collection can be used to improve precision by distinguishing documents in which they occur from those in which they do not. Let df_j denote the document frequency of the term T_j in a collection of N documents, which is the number of documents in which the term occurs. Then, the inverse document frequency, given by $\log(N / df_j)$, is an appropriate indicator of T_j as a document discriminator.

The term-frequency and inverse-document-frequency components can be combined into a single frequency-based indexing model,[1,2] where the weight of a term T_j in document D_i denoted w_{ij} is given by

$$w_{ij} = tf_{ij} \log(N / df_j)$$

Another statistical approach to indexing is based on *term discrimination.* This approach views each document as a point in the document space. As the term sets for two documents become more similar, the corresponding points in the document space become closer (that is, the density of the document space increases) and vice versa.

Under this scheme, we can approximate the value of a term as a document discriminator based on the type of change that occurs in the document space when a term is introduced to the collection. We can quantify this change according to the increase or decrease in the average distance between the documents. A term has a good discrimination value if it increases the average distance between the documents; in other words, terms with good discrimination value decrease the density of the document space. The term-discrimination value of a term T_j, denoted dv_j, is then computed as the difference of the document space densities before and after the term T_j is introduced. The net effect is that high-frequency terms have negative discrimination values, medium-frequency terms have positive discrimination values, and low-frequency terms tend to have discrimination values close to zero.[1] A term-weighting scheme such as $w_{ij} = tf_{ij} dv_j$ is used to combine term frequency and discrimination values.

Information-theoretic methods. In information theory, the least-predictable terms carry the greatest information value.[3] Least-predictable terms are those that occur with smallest probabilities. Information theory concepts have been used to derive a measure, called *signal-noise ratio*, of term usefulness for indexing. This method favors terms that are concentrated in particular documents. Therefore, its properties are similar to those of inverse document frequency.

Probabilistic methods. Probabilistic approaches require a training set of documents obtained by asking users to provide relevance judgments with respect to query results.[4] The training set is used to compute term weights by estimating conditional probabilities that a term occurs given that a document is relevant (or irrelevant). Assume that a collection of N documents of which R are relevant to the user query, R_t of the relevant documents contain term t, and t occurs in f_t documents. Two conditional probabilities are estimated for each term as follows:

$$\Pr[t \text{ in document} \mid \text{document is relevant}] = R_t / R;$$
$$\Pr[t \text{ in document} \mid \text{document is irrelevant}] = (f_t - R_t) / (N - R).$$

From these estimates, Bayes' theorem is used, under certain assumptions, to derive the weight of term t as

Set Theoretic Models

The *Boolean model* represents documents by a set of index terms, each of which is viewed as a Boolean variable and valued as True if it is present in a document. No term weighting is allowed. Queries are specified as arbitrary Boolean expressions formed by linking terms through the standard logical operators: AND, OR, and NOT. Retrieval status value (RSV) is a measure of the query-document similarity. In the Boolean model, RSV equals 1 if the query expression evaluates to True; RSV is 0 otherwise. All documents whose RSV evaluates to 1 are considered relevant to the query.

This model is simple to implement and many commercial systems are based on it. User queries can employ arbitrarily complex expressions, but retrieval performance tends to be poor. It is not possible to rank the output since all retrieved documents have the same RSV, nor can weights be assigned to query terms. The results are often counter-intuitive. For example, if the user query specifies 10 terms linked

$$w_t = \log \frac{R_t / \left(R - R_t\right)}{\left(f_t - R_t\right) / \left(N - f_t - \left(R - r_t\right)\right)}$$

The numerator (denominator) expresses the odds of term *t* occurring in a (irrelevant) relevant document. Term weights greater than 0 indicate that the term's occurrence in the document is evidence of the document's relevance to the query; values less than 0 indicate its irrelevance.

Multi-term or phrase indexing

Single terms are less than ideal for an indexing scheme because their meanings out of context are often ambiguous. Term phrases, on the other hand, carry more specific meaning and thus have more discriminating power. *Phrase generation* is intended to improve precision; *thesaurus-group generation* is expected to improve recall. A thesaurus assembles groups of related specific terms under more general, higher level class indicators.

Methods for generating complex index terms or term phrases automatically may be categorized as statistical, probabilistic, or linguistic.

Statistical methods. A term phrase consists of the phrase head, which is the principal phrase component, and other components. A term with document frequency exceeding a stated threshold, such as *df > 2*, is designated as the phrase head. Other components of the phrase should be medium- or low-frequency terms with stated co-occurrence relationships with the phrase head, for example, that the phrase components should occur in the same sentence as the phrase head within a stated number of words.

Term grouping or *term clustering* methods are used to generate groups of related words by observing word co-occurrence patterns in a document collection. Given a term-document matrix as a 2D array, one method compares the columns of the matrix to each other and assesses whether the terms are jointly assigned to many documents in the collection. If so, the terms are assumed to be related and are grouped into the same class.

Probabilistic methods. Probabilistic methods generate complex index terms based on term-dependence information. Since this requires considering an exponential number of term combinations and, for each combination, estimating the probabilities of coincidences in relevant and irrelevant documents, only certain dependent-term pairs are considered in practice. In theory, these dependencies can be user specific.

In the statistical and probabilistic approaches, terms that occur together are not necessarily related semantically. Therefore, these approaches are not likely to lead to high-quality indexing units.

Linguistic methods. Assigning syntactic class indicators such as adjective, noun, or verb to terms can enhance the statistical method described above. Phrase formation is then limited to sequences of specified syntactic indicators (for example, noun-noun, adjective-noun). A simple syntactic analysis process can be used to identify syntactic units. The phrase elements can then be chosen from within the same syntactic unit.

Linguistic approaches for generating term relationships usually involve the use of an electronic lexicon.[5] There are also proposals for generating term relationships based on user feedback.[6] Though various automatic methods for thesaurus construction have been proposed, their effectiveness is questionable outside the special environments in which they are generated.

REFERENCES

1. G. Salton, *Automatic Text Processing*, Addison-Wesley, Reading, Mass., 1989.
2. W.B. Croft, "Experiments with Representation in a Document Retrieval System," *Information Technology*, Vol. 2, No. 1, 1983, pp.1-21.
3. C.E. Shannon, "Prediction and Entropy in Printed English," *Bell Systems J.*, Vol. 30, No. 1, 1951, pp. 50-65.
4. S. E. Robertson and K. Sparck-Jones, "Relevance Weighting of Search Terms," *J. Am. Soc. of Information Sciences*, 1976, pp.129-146.
5. G.A. Miller, "WordNet: A Lexicon Database for English," *Comm. ACM*, Vol. 38, No. 11, Nov. 1995, pp. 39-41.
6. G.S. Jung and V.V. Raghavan, "Connectionist Learning in Constructing Thesaurus-like Knowledge Structure," *Working Notes of AAAI Symp. on Text-Based Intelligent Systems*, Mar. 1990, Palo Alto, Calif., pp. 123-127.

by the logical connective AND, a document that has nine of these terms is not retrieved. User relevance feedback is often used in IR systems to improve retrieval effectiveness. Typically, a user is asked to indicate the relevance or irrelevance of a few documents placed at the top of the output. Since the output is not ranked, however, the selection of documents for relevance feedback elicitation is difficult.

The *fuzzy-set model* is based on fuzzy-set theory,[3] which allows partial membership in a set, as compared with conventional set theory, which does not. It redefines logical operators appropriately to include partial set membership, and processes user queries in a manner similar to the case of the Boolean model. Nevertheless, IR systems based on the fuzzy-set model have proved nearly as incapable of discriminating among the retrieved output as systems based on the Boolean model.

The strict Boolean and fuzzy-set models are preferable to other models in terms of computational requirements, which are low in terms of both the disk space required for storing document representations and the algorithmic complexity of indexing and computing query-document similarities.

Algebraic Models

The *vector-space model* is based on the premise that documents in a collection can be represented by a set of vectors in a space spanned by a set of normalized term vectors.[4] If the vector space is spanned by n normalized term vectors, then each document will be represented by an n-dimensional vector. The value of the first component in this vector reflects the weight of the term in the document corresponding to the first dimension of the vector space, and so forth. A user query is similarly represented by an n-dimensional vector. A query-document's RSV is given by the scalar product of the query and document vectors. The higher the RSV, the greater is the document's relevance to the query.

The strength of this model lies in its simplicity. Relevance feedback can be easily incorporated into it. However, the rich expressiveness of query specification inherent in the Boolean model is sacrificed.

Probabilistic Models

The vector-space model assumes that the term vectors spanning the space are orthogonal and that existing term relationships need not be taken into account. Furthermore, the model does not specify the query-document similarity, which must be chosen somewhat arbitrarily. The *probabilistic model* takes these term dependencies and relationships into account and, in fact, specifies major parameters such as the weights of the query terms and the form of the query-document similarity.

The model is based on two main parameters—Pr(rel) and Pr(nonrel), the probabilities of relevance and nonrelevance of a document to a user query—which are computed using the probabilistic term weights (see the sidebar, "Automatic Indexing Methods") and the actual terms present in the document. Relevance is assumed to be a binary property so that Pr(rel) = 1 – Pr(nonrel). In addition, the model uses two cost parameters, a_1 and a_2, to represent the loss associated with the retrieval of an irrelevant document and nonretrieval of a relevant document, respectively.

The model requires term-occurrence probabilities in the relevant and irrelevant parts of the document collection, which are difficult to estimate. However, this model serves an important function for characterizing retrieval processes and provides a theoretical justification for practices previously used on an empirical basis (for example, the introduction of certain term-weighting systems).

Hybrid Models

As in the case of the vector-space model, the *extended Boolean model* represents a document as a vector in a space spanned by a set of orthonormal term vectors. However, the extended Boolean (or p-norm) model measures query-document similarity by using a generalized scalar product between the corresponding vectors in the document space. This generalization uses the well-known L_p norm defined for an n-dimensional vector, d, where the length of d is given by

$$|\mathbf{d}| = |(w_1, w_2, \ldots, w_n)| = \left(\sum_{j=1}^{n} w_j^p\right)^{\frac{1}{p}}$$

where $1 <= p <= \infty$, and $w_1, w_2, \ldots, w_n$ are the components of the vector d.

Generalized Boolean OR and AND operators are defined for the p-norm model. The interpretation of a query can be altered by using different values for p in computing query-document similarity. When $p = 1$, the distinction between the Boolean operators AND and OR disappears, as in the case of the vector-space model.

When the query terms are all equally weighted and $p = \infty$, the interpretation of the query is the same as that in the fuzzy-set model. On the other hand, when the query terms are not weighted and $p = \infty$, the p-norm model behaves like the strict Boolean model. Varying the value of p from 1 to ∞ offers a retrieval model whose behavior corresponds to a point on the continuum spanning from the vector-space model to the fuzzy and strict Boolean models.

The best value for p is determined empirically for a collection, but is generally in the range $2 <= p <=5$.

A TAXONOMY FOR SEARCH TOOLS AND SERVICES

Automated methods for retrieving information on the Web can be broadly classed as search tools or search services.

Search tools employ robots for indexing Web documents. They feature a user interface for specifying queries and browsing the results. At the heart of a search tool is the search engine, which is responsible for searching the index to retrieve documents relevant to a user query. Search tools can be distinguished as type 1 or type 2 based on the transparency of the index to the user.

Search services provide users a layer of abstraction over several search tools and databases and aim at simplifying the Web search.

We describe type 1 and type 2 search tools along the following dimensions:

- methods for Web navigation,
- indexing techniques,
- query language or specification scheme for expressing user queries,
- strategies for query-document matching, and
- methods for presenting the query output.

Type 1 Search Tools

These tools completely hide the organization and content of the index from the user.

*AltaVista** has a spider (called Scooter) that traverses the Web and Usenet newsgroups. Indexing is based on the full text of the document. The first few lines are used as an abstract. HTML document authors can use the META tag to specify index terms and a short description of their documents. The AltaVista index is updated at least once a day. Scooter visits pages according to how frequently they appear to be changing: A page that has been stable for months will be revisited less often than a page that is different every time Scooter visits it. AltaVista supports full Boolean, phrase, and case-sensitive searches. It ranks results on the basis of relevance, giving a higher score to documents that contain the query terms in the first few words, documents in which the query terms are found close to each other, and documents containing more than one instance of the query terms. Results include the title, a short abstract, size, and date of the last modification for each retrieved document.

*Excite** also has a spider and an indexer for the full text of documents. The spider retrieves only Web and Usenet newsgroup documents. Users can submit URLs for indexing. The indexer generates index terms and a short document summary. The Excite index consists of about 50 million URLs. It supports proper name searches, the Boolean operators AND, OR, and NOT, and Boolean expression queries. Search results are rank ordered, and a summary is provided for each retrieved document. Excite provides a “similar” query and a “sort by site” option to display sites rank ordered according to the number of documents retrieved from the site.

*HotBot** retrieves and indexes Web documents using a robot called Slurp and a parallel network of workstations. The robot extracts all the URLs from a retrieved document and places them into a scheduling data structure, which assigns the URLs to different CPUs according to criteria such as how recently a host has been accessed. Users can also submit URLs for indexing. HotBot indexes the full text of only HTML and plain-text documents. Terms are weighted, and the indexer generates a short abstract. Indexes are distributed across several computers, which enables HotBot to process a query in parallel. Users can search by a term, phrase, proper noun, or URL. HotBot also supports case-sensitive and Boolean searches. Advanced users have options to specify a search for a particular type of media or format. Users can also limit searches to specific Internet domains. The search results are rank ordered. HotBot assigns relevance to documents according to various factors such as term frequency and document length. If query terms occur in a document’s title or META tag, higher relevance is assigned to that document. For retrieved documents, HotBot provides the last modified date and a short abstract consisting of the first few lines of the document.

*InfoSeek Guide** is a popular search engine with a robot that retrieves HTML and PDF documents, indexes full text, and generates a short summary of each document. InfoSeek allows searches in the Web, Usenet groups, and Web FAQs. Its indexes are distributed. It supports case sensitivity as well as searches for symbols, phrases, and proper names. It also allows image searches based on the captions or index terms assigned to images. InfoSeek ranks its output, calculating the RSV by giving more weight to documents that contain the query terms at the beginning of the document. It returns a short summary, relevancy score, and document size. InfoSeek also provides a “similar pages” query.

*Lycos** has a robot that uses heuristics to navigate the Web and build a searchable index. For each document indexed, the robot keeps the outgoing links (anchor text or link tags) in a queue and selects a URL from it. One heuristic, for example, might force the robot to select a URL that points to a Web server’s home page. Users can submit URLs for indexing. Lycos indexes titles, headings, and subheadings of HTML, FTP, and Gopher documents. When the number of index terms exceeds 100, only the 100 most-weighted terms based on the *tf.idf* scheme are kept. The indexer also keeps the first 20 lines of a document, its size in bytes, and the number of words. Lycos has options for matching any term, all terms, or some number of terms. It allows searches for word fragments and has options for loose, fair, close, good, and strong matches. It supports the Boolean operator NOT. The RSV is computed as the sum of the weights of matched terms in the document to the query. Index terms that appear in the title and near the beginning of the document are given more weight. The output is ranked and presented with clickable URLs. The document size and RSV are included in the output.

*OpenText** has a robot that traverses the Web by selecting a URL from a URL pool, retrieving the document at that URL, and indexing the document. It also extracts all the URLs in the retrieved document and places them in the

URL pool. Users can submit URLs for indexing. OpenText indexes the full text of HTML documents and updates its index continuously. The indexer generates a short summary consisting of the first 100 words of the document. It supports full Boolean searching as well as searches for proper names, symbols, and phrases.

*WebCrawler** has a robot that starts with a known set of HTML documents and uses the URLs in them to retrieve new documents. The search engine directs the navigation in a modified breadth-first mode. It maintains a list of Web servers and URLs to fetch from them, which it does in a round-robin fashion to avoid fetching documents consecutively from the same server. WebCrawler aims at indexing at least one document from each server. Users can also submit URLs. It indexes both the title and full text of HTML documents, and its index is updated weekly. Terms are weighted by their frequency of occurrence in the document divided by their frequency in the reference domain (*tf.idf*). Terms that appear frequently in the document and infrequently in the reference domain are heavily weighted, while those that appear infrequently in either are given lower weights. WebCrawler supports full Boolean and phrase searches. The query processor uses the vector-space model to compute RSV. The output is rank ordered in a list of clickable URLs that includes a short summary and a relevancy score. It also provides “similar pages” query.

The *Worldwide Web Worm** (WWWW) consists of two components: a resource locator and a search engine. The resource locator traverses the Web in depth-first mode and indexes the titles, anchor text, and URLs of HTML documents. The resource locator stores the indexing information in a flat file. The WWWW also indexes in-line images by indexing their HTML titles and clickable hypertext (if any). It supports the Boolean AND and OR operators. The WWWW is limited by its lack of content-based indexing. Title, anchor text, and URL names alone cannot represent document content. Also, it has been estimated that 20 percent of the HTML documents on the Web have no title.

Type 2 Search Tools

Type 2 search tools feature a hierarchically organized subject catalog or directory of the Web, which is visible to users as they browse and search.

*Yahoo** is a semi-automatically constructed, hierarchically organized Web subject catalog that is both browsable and searchable. Links to various resources are collected in two ways: by user submissions and by robots that retrieve new links from well-known pages such as NCSA/GNN’s What’s New Page. Yahoo indexes Web, Usenet news, and e-mail addresses. It provides Boolean AND and OR operators and phrase searching. The query output is a list of documents and related Yahoo categories, along with the first few lines of the document.

*Magellan** is another subject catalog. It indexes Web sites, FTP and Gopher servers, Usenet news, and telnet sessions. A team of editors and writers review and rate Web sites according to factors such as comprehensiveness and ease of exploration. Users can submit URLs for review. The Magellan catalog includes original editorial content, a directory of rated and reviewed sites, a database of yet-to-be-reviewed sites, and a search engine. Magellan provides +/– options similar to the Boolean AND and OR operators. RSV is assigned based on the frequency of query terms in the document. Higher relevance is assigned to those documents that contain query terms in their title, META tag, or URL. The query output is ranked.

Other tools in this category include WWW Virtual Library and Galaxy. *WWW Virtual Library** is a distributed subject catalog, browsable and maintained by volunteers. No engine is provided for searching the catalog. *Galaxy** indexes Web and Gopher documents. Its index is both searchable and browsable.

Search Services

Search services broadcast user queries to several search engines and various other information sources simultaneously. Then they merge the results submitted by these sources, check for duplicates, and present them to the user as an HTML page with clickable URLs. For example, *IBM InfoMarket** searches Yahoo, OpenText, Magellan, various business resources, and Usenet newsgroups simultaneously and generates a rank-ordered query output. *MetaCrawler** is another search service that sends queries to eight different search engines: OpenText, Lycos, WebCrawler, InfoSeek, Excite, AltaVista, Yahoo, and Galaxy. MetaCrawler supports both Boolean and phrase search.

RETRIEVAL EFFECTIVENESS ASSESSMENT

The formal precision and recall measures used to quantify retrieval effectiveness of IR systems are based on evaluation experiments conducted under controlled conditions. This requires a testbed comprising a fixed number of documents, a standard set of queries, and relevant and irrelevant documents in the testbed for each query. Realizing such experimental conditions in the Web context is extremely difficult. Search engines operate on different indexes, and the indexes differ in their coverage of Web documents.

We must therefore compare retrieval effectiveness in terms of qualitative statements and the number of documents retrieved. We evaluated various search tools and services using two queries: “latex software” and “multiagent system architecture.” The first query was intended to find both public-domain sources and commercial vendors for obtaining LaTex software, whereas the second query was intended to locate relevant research publications on multiagent system architecture.

Table 1 presents results for the first query. The second column indicates the number of documents retrieved by interpreting the query as a *disjunction* of the query terms.

Disjunction is the default interpretation in all the search tools, except OpenText. The third column shows the number of documents retrieved by interpreting the query as a *conjunction* of the query terms. Finally, the fourth column denotes the number of documents retrieved by interpreting the query as a phrase. N/A in the table cell indicates that the query option is not available.

Among the search tools, InfoSeek retrieved the largest number of documents (over 3 million); WWWW retrieved the smallest number (4,999). In general, the number of documents retrieved decreased moving from disjunctive to conjunctive to phrase queries. In the case of Excite, the number of documents retrieved for both conjunctive and phrase queries was the same.

We examined the first 10 documents of the ranked results to see if they contained information on obtaining LaTex software. The following results did:

- Excite* (all three query types)
- HotBot* (conjunctive and disjunctive) queries)
- InfoSeek Guide* (conjunctive query)
- OpenText* (phrase query)
- WebCrawler* (phrase query)
- WWW Worm* (conjunctive and disjunctive queries)
- Magellan* (conjunctive and disjunctive)
- MetaCrawler* (conjunctive query)

Table 1. Comparison of number of results for “latex software” query.

Search tool/ service	Disjunctive query	Conjunctive query	Phrase query
AltaVista	200,000	30,000	100
Excite	134,669	29,287	29,287
HotBot	3,696,449	61,830	17,630
InfoSeek Guide	3,111,835	427	100
Lycos	29,881	26	N/A
OpenText	481,846	2,541	6
WebCrawler	158,751	864	6
WWW Worm	4,999	2	N/A
Galaxy	6,351	20	N/A
Magellan	17,658	17,658	N/A
Yahoo	373 categories 18,344 sites	1 category 3 sites	N/A 101 sites
IBM InfoMarket	100	N/A	N/A
MetaCrawler	29	32	34

Table 2. Results for “multiagent system architecture” query.

Search tool/ service	Disjunctive query	Conjunctive query	Phrase query
AltaVista	40,000,000	700	6
Excite	514,321	3	561
HotBot	18821	383	0
InfoSeek Guide	10,960,353	79	38
Lycos	53,525	0	N/A
OpenText	604,487	33	0
WebCrawler	150,619	0	0
WWW Worm	2,000	0	N/A
Galaxy	797	0	N/A
Magellan	58,080	84	N/A
Yahoo	615 categories 23,560 sites	0 5,890 sites	0 6 sites
IBM InfoMarket	100	N/A	N/A
MetaCrawler	38	29	6

Table 2 presents results for the second query, multiagent system architecture. Excepting WWWW and Galaxy, all tools and services retrieved literature on multiagent system architecture. Conjunctive and phrase queries performed better than disjunctive queries. This query is more difficult to evaluate than the first one, since the phrase has several facets to it. The results are useful in that the top-ranked documents can seed a manual refinement of the query specification to a single facet of multiagent system architecture.

For both test queries, relevant documents were interspersed with irrelevant documents in the ranked query output. This means that the retrieval user cannot afford to examine only a few documents placed at the top of the rank ordering and discard the rest. However, since the number of documents in the ranked query output is in the order of thousands, manually sifting through this output to glean relevant documents is tedious and error-prone.

IMPROVING RETRIEVAL EFFECTIVENESS

The design and development of current-generation Web search tools have focused on query-processing speed and database size. This is largely a response to the lack of features in the original HyperText Markup Language* for representing document content to search tools[5,6] (not surprising, given HTML’s original purpose: to render documents on a wide array of output devices without concern for the computer to which the device was connected).

HTML Version 3 introduced the META tag, which allows authors to specify indexing information. We expect this trend to continue, establishing standardized tags for Web document content. Meanwhile, as we have seen, the ranked responses from many searches are in the order of thousands, and the burden is on the user to sift through the list and identify relevant documents. The focus should instead shift to providing a short, ranked list of documents, even if the system takes longer to process the query. In this section, we discuss two methods for improving the retrieval effectiveness of Web search tools.

Relevance Feedback Techniques

Unlike a database environment, an IR environment lacks precise representations for user queries and documents. Users typically start with an imprecise and incomplete query, and improve the query specification—hence, the retrieval effectiveness—iteratively and incrementally.[1,4] The user is asked to provide evaluations or relevance feedback on the documents retrieved from the initial query. This feedback is used in subsequently improving the retrieval effectiveness.

Relevance feedback is elicited in either *two-level* or *multilevel* relevance relations. In the former case, the user simply labels a retrieved document as relevant or not; in the latter, a document can be relevant, somewhat relevant, or not relevant. Multilevel relevance can also be specified in terms of relationships. For example, for three retrieved documents d_1, d_2, and d_3, d_1 can be more relevant than d_2, and d_2 more relevant than d_3.

For simplifying our presentation of relevance feedback, we assume two-level relevance and the vector-space model. The set of documents deemed relevant by the user constitute *positive feedback*; those deemed irrelevant constitute *negative feedback*.

Figure 1 shows the two major approaches to utilizing relevance feedback: modifying the query and modifying the document representations. Methods based on modifying the query representation affect only the current user-query session and have no effect on other user queries; methods based on modifying the representation of documents in the collection affect the retrieval effectiveness of future queries. In all the methods, more than two or three iterations may result in minimal improvements.

The basic assumption for relevance feedback is that documents relevant to a particular query resemble each other—in the vector-space model, their corresponding vectors are similar. Using relevance-feedback techniques in Web search engines requires document representations to be more descriptive and semantically rich than just indexing the title or abstract of the document. One way to achieve this is to index the entire document. The vector-space model readily accommodates all relevance feedback techniques, whereas the probabilistic model needs special extensions to accommodate query expansion.

Modifying the query representation.

There are three ways to improve retrieval effectiveness by modifying the query representation. The first, *modification of term weights*, involves adjusting the query term weights by adding document vectors in the positive feedback set to the query vector. Optionally, negative feedback can be used to subtract document vectors in the negative feedback set from the query vector. The reformulated query should retrieve additional relevant documents similar to the documents in the positive feedback set. This process can be carried out iteratively until the user is satisfied with the quality and number of relevant documents in the query output.

Experimental results indicate that positive feedback is more consistently effective. This is due to the fact that documents in the positive feedback set are generally more homogeneous than documents in the negative feedback set. However, an effective feedback technique, termed *dec hi*, uses all documents in the positive feedback set and subtracts from the query only the vectors of highest ranked irrelevant documents in the negative feedback set.[7]

The second method, *query expansion*, modifies the original query by adding new terms to it. The new terms are selected from the positive feedback set and sorted using measures such as

- noise (a global term-distribution measure similar to idf),
- postings (the number of retrieved relevant documents containing the term),
- noise within postings,
- noise.frequency within postings (frequency is $\log_2$ of the total frequency of the term in the retrieved relevant set),
- noise.frequency.postings, and
- noise.frequency.

A predefined number of top terms from the sorted list are added to the query. Experimental results show that the last three sort methods produced the best results and that adding only selected terms is superior to adding all terms. There is no performance improvement by adding more than 20 terms.[7]

In some cases, the above two techniques do not produce satisfactory results because the documents in the positive feedback set are not homogeneous (that is, they do not form a tight cluster in the document space) or because the irrelevant documents are scattered among certain relevant ones. One way to detect this situation is to cluster the documents in the positive feedback set to see if more than one homogeneous cluster exists. This method is called *query splitting*. If the documents cluster, the query is split into subqueries such that each subquery represents one cluster in the positive feedback set. The weight of terms in the subquery can then be adjusted or expanded as in the previous two methods.

Modifying the document representation.

This approach involves adjusting the document vectors in the collection based on relevance feedback. It is also referred to as *user-oriented clustering*.[8] It is implemented by adjusting the weights of retrieved and relevant document vectors to move them closer to the query vector. The weights of retrieved irrelevant-document vectors are adjusted to move them farther from the query vector. Care must be taken to ensure that individual document movement is small, since user-relevance assessments are necessarily subjective.

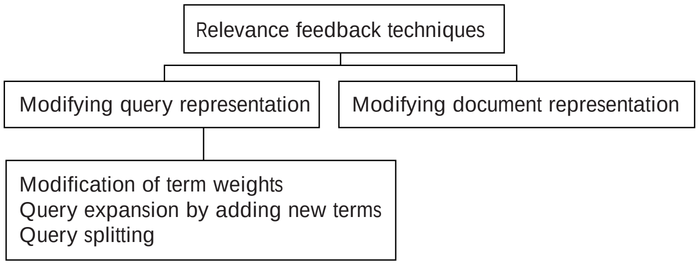
Relevance feedback techniques
Modifying query representation
Modifying document representation
Modification of term weights
Query expansion by adding new terms
Query splitting

Figure 1. A taxonomy for relevance feedback techniques.

Agent-Based Filtering and Routing

Information agents offer a distinctly different paradigm for harnessing Web information. These computer programs work collaboratively with a user but do not require explicit initiation by the user. Intelligent agents are equipped with knowledge, problem-solving methodologies, and data pertinent to the problem; they also have their own built-in control mechanism.[9]

In the context of Web searching, agents can perform tasks such as discovering documents (much like the Web robots), indexing documents, filtering them, and automatically routing useful and interesting information to users. Agents lend themselves to personalization, and they have learning and adaptation capabilities.

In a single-agent system, the agent works alone, doing all the tasks by itself. Single-agent systems are centralized. Multiagent systems are decentralized and distribute tasks among a number of agents. In fact, the multiagent paradigm perfectly suits information discovery and retrieval in the Web. For example, information discovery can be handled by one agent, another agent can specialize in indexing, yet another can implement an information retrieval model, and so forth.

Agents employ domain and user models in performing their tasks. The *domain model* encodes domain-specific background knowledge about the Web, its processes, information resources relevant to the various disciplines of a user's interests, and domain ontologies.[10] The *user model* captures information about the user's background, interests, and preferences.[11]

Expressiveness is an important issue in the representation chosen for both models. Initial models are constructed manually. The agent's learning component helps to evolve the domain and user models with changes in Web resources and user interests. Relevance-feedback techniques are an important source for evolving the user model.

A multiagent architecture for information-retrieval based on these ideas is described in Gudivada and Tolety.[12] The system features generic templates for the user and domain models, and the models are instantiated for individual users. The system then semi-autonomously selects or discards information of interest to the user from a dynamically changing Web index.

CONCLUSIONS

None of the current generation of search tools incorporates relevance-feedback or user-modeling techniques. Applying these techniques to the list of documents retrieved by a search tool could substantially weed out unrelated documents and improve the ranking quality of the remaining documents. Agent-based personalized information filtering and retrieval is a promising research direction to improve the retrieval effectiveness of search tools. The agent-based information retrieval paradigm offers a natural means to incorporate domain models using ontologies in a way transparent to the system user.[12]

Indexing quality has an overwhelming effect on retrieval effectiveness. In fact, it has been called one of the grand challenges in the digital libraries realm.[13] The task of automatically assigning high-quality terms to documents remains elusive, though the problem has been studied for many years. Recently, this problem has received a renewed interest in the information retrieval area under the name *text categorization*, which aims at automatically discovering subject categories and domain concepts manifested in documents. High-quality retrieval requires work on this grand challenge problem.

Comprehensively indexing the entire Web and building one huge integrated index will only further deteriorate retrieval effectiveness, since the Web is growing at an exponential rate. On the other hand, a collection of Web indexes, each with its own specialized search tool, holds promise. Under this scheme, each Web index is targeted to comprehensively represent documents of a specific information space. Information spaces are bounded by, for example, academic disciplines, a class of industries, a group of services. The commonality in the subject matter indexed supports the capture of semantic-level features and the incorporation of domain semantics into the indexing process. The search tool for such an index can also be specialized for the information space.

An open-ended system with an appropriate suite of protocols can provide a layer of abstraction on the individual indexes and the associated search tools. Such a system should also be able to handle the participation and withdrawal of individual indexes. This situation is analogous to a federat-

URLs FOR THIS ARTICLE

*AltaVista • www.altavista.digital.com/av/content/about.htm
*Excite • www.excite.com
*HotBot • www.hotbot.com
*IBM InfoMarket • www.infomarket.ibm.com
*InfoSeek Guide • www.infoseek.com
*Lycos • www.lycos.com
*Magellan • www.mckinley.com
*MetaCrawler • www.metacrawler.com/index_text.html
*OpenText • www.opentext.com
*WebCrawler • www.webcrawler.com
*WorldWide Web Worm • www.goto.com
*Yahoo • www.yahoo.com
*WWW Virtual Library • vlib.stanford.edu
*Galaxy • galaxy.einet.net

ed database system, in which the component databases operate under complete autonomy yet cooperate in processing queries that require information stored in multiple component databases. Such a system can be implemented using the agent-based information retrieval paradigm.

Unless the current generation of search tools and services significantly improve their retrieval effectiveness, the Web will continue to evolve toward an information entertainment center for users with no specific search objectives. ■

ACKNOWLEDGMENT

The authors are grateful to the anonymous referees whose suggestions have significantly improved the clarity and content of this article.

REFERENCES

1. W.B. Frakes and R. Baeza-Yates, eds., *Information Retrieval: Data Structures and Algorithms*, Prentice Hall, Englewood Cliffs, N.J., 1992.
2. G. Salton, *Automatic Text Processing*, Addison-Wesley, Reading, Mass., 1989.
3. T. Radecki, "Fuzzy Set Theoretical Approach to Document Retrieval," *Information Processing and Management*, Vol. 15, 1979, pp. 247-259.
4. V. Raghavan and S.K.M. Wong, "A Critical Analysis of Vector Space Model for Information Retrieval," *J. Am. Soc. Information Science*, Vol. 37, No. 5, 1986, pp. 279-287.
5. O. Etzioni and D. Weld, "Intelligent Agents on the Internet: Fact, Fiction, and Forecast," *IEEE Expert*, Vol. 10, No. 4, 1995, pp. 44-49.
6. O. Etzioni, "The World-Wide Web: Quagmire or Gold Mine?" *Comm. ACM*, Vol. 39, No. 11, Nov. 1996, pp. 65-68.
7. D. Harman, "Relevance Feedback Revisited," *Proc. 15th Ann. Int'l ACM SIGIR Conf.*, ACM Press, New York, 1992, pp. 1-10.
8. J. Bhuyan et al., "An Adaptive Information Retrieval System Based on User-Oriented Clustering," submitted to *ACM Transactions on Information Systems*, Jan. 1997.
9. D.E. O'Leary, "The Internet, Intranets, and the AI Renaissance," *Computer*, Vol. 30, No. 1, Jan. 1997, pp. 71-78.
10. A. Farquhan et al., "Collaborative Ontology Construction for Information Integration," Tech. Report: KSL-95-63, Knowledge Systems Laboratory, Dept. of Computer Science, Stanford Univ., Stanford, Calif., Aug. 1995.
11. M.F. McTear, "User Modeling for Adaptive Computer Systems: A Survey of Recent Developments," *Artificial Intelligence Review*, Vol. 7, 1993, pp. 157-184.
12. V. Gudivada and S. Tolety, "A Multiagent Architecture for Information Retrieval on the World-Wide Web," to appear in *Proc. Fifth RIAO Conf. Computer Assisted Information Searching on the Internet*, Centre de Hautes Etudes Internationales d'Informatique Documentaires, Paris, 1997.
13. B. Schatz et al., "Federating Diverse Collections of Scientific Literature," *Computer*, Vol. 29, No. 5, May 1996, pp. 28-36.

Venkat N. Gudivada is a senior database designer at Dow Jones Markets. His research interests are in multimedia information retrieval and heterogeneous distributed database management. He received his PhD in computer science from the University of Southwestern Louisiana in 1993.

Vijay V. Raghavan is a distinguished professor of computer science at the University of Southwestern Louisiana. His research focuses on information retrieval strategies for text and image databases. He received a B. Tech in mechanical engineering from the Indian Institute of Technology, Madras; an MBA from McMaster University; and a PhD in computing science from the University of Alberta. Raghavan is currently an ACM National Lecturer. He is a member of the ACM and the IEEE.

William I. Grosky is professor and chair of the Computer Science Department at Wayne State University in Detroit, Michigan. His research interests include multimedia information systems, hypermedia, and Web technology. He received a BS in mathematics from MIT in 1965, MS in applied mathematics from Brown University in 1968, and PhD in engineering and applied science from Yale University in 1971. Grosky is currently on the editorial boards of *IEEE MultiMedia*, *Pattern Recognition*, and the *Journal of Database Management*.

Rajesh Kasanagottu is a graduate student at the University of Missouri, Rolla. He received his bachelor's degree in computer science from Osmania University, India. His research interests include software agents, search engines, and information retrieval.

Readers may contact Raghavan at the Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette, LA 70504, USA; raghavan@cacs.usl.edu.