Search and Ranking Algorithms for Locating Resources on the World Wide Web

Budi Yuwono
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio, U.S.A.

Dik L. Lee
Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong

Abstract

Applying information retrieval techniques to the World Wide Web (WWW) environment is a unique challenge, mostly because of its hypertext/hypermedia nature and the richness of the meta-information it provides. We present four keyword-based search and ranking algorithms for locating relevant WWW pages with respect to user queries. The first algorithm, Boolean Spread Activation, extends the notion of word occurrence in Boolean retrieval model by propagating the occurrence of a query word in a page to other pages linked to it. The second algorithm, Most-cited, is based on the number of citing hyperlinks between potentially relevant WWW pages to increase the relevance scores of the referenced pages over the referencing pages. The third algorithm, TFxIDF or vector space model, is based on word distribution statistics. The last algorithm, Vector Spread Activation, combines vector space model and spread activation model. We conducted an experiment to evaluate the retrieval effectiveness of these algorithms. From the results of the experiment, we draw conclusions regarding the nature of the WWW environment with respect to document ranking strategies.

Keywords: resource discovery, information retrieval, world wide web indexing, text database

1 Introduction

The World Wide Web (WWW) [4] has become one of the fastest growing applications on the Internet today. Its popularity can be attributed mainly to its uniform access method to various network information services and its hypermedia support which links a wide range of multimedia data physically distributed all around the world into a single gigantic virtual database. WWW also provides a powerful and easy to setup medium for almost any user on the Internet to disseminate information. More and more information has become available on-line through WWW, from personal data to scientific reports to up-to-the-minute satellite images. This information explosion leads to a problem commonly known as resource discovery problem. In order to find interesting WWW pages, a user has to browse through many WWW sites. This is a very time consuming process.

Methods to relief the users from this information overflow problem have been explored by others, from creating a special Usenet [9] newsgroup[1] for announcing new WWW sites, to sharing personal hotlists (accessible from the owner's home pages), compiled list and directories, to searchable full-text index databases. In this paper, we present a WWW index server designed to help users locate WWW pages using keyword search. Based on how the index is built, there are two categories of WWW searchable-index servers, namely manually generated index servers and robot generated index servers.

Among the well known manually built index servers are Yahoo[2] and EInet Galaxy.[3] The main advantage of manual indexing is that Web pages can be organized, hierarchically or otherwise, by subject such as the subject tree structure in Yahoo and EInet Galaxy. Of course, such categorizations are subjective and may be biased to the maintainer's knowledge background. A slightly different scheme of manually generated index system is one used by the Distributedly Administered Categorical List Of Documents (DACLOD[4]) and the Global On-Line Directory (GOLD[5]), which allows any user to add an entry (a pointer to a Web page along with other information) into the index database. Similar to the above scheme is that of Archie-like Web server (ALIWEB[6]) [8]. Instead of users registering their WWW pages, ALIWEB retrieves index data from each of the participating WWW servers. This index data is prepared manually by the respective WWW server maintainers in a standard text format containing the description of information provided by the servers.

Our index server falls into the robot-generated index server category. Robot-based indexing is faster and more comprehensive than manual indexing and, since it is automated, it is easy to update the index as often as necessary. We discuss some of the robot-based index servers in the Internet as of this writing, as compared to ours, in the last section of this paper.

Our WWW index server[7] takes a query from a user and returns a list of URL's (Uniform Resource Locators [1] or WWW page addresses) along with their titles, ranked by relevance score. It supports Boolean query constructs which make use of & (logical AND), | (logical OR), and arch brackets (scope markers) notations. To accommodate users who are not familiar with Boolean constructs, it allows the keywords to be separated by white spaces, which are treated as logical AND operators. Hyphens may also be used to specify phrases so

[1] *comp.infosystems.www.announce* newsgroup.

[2] ⟨http://www.yahoo.com/⟩.

[3] ⟨http://galaxy.einet.net/⟩.

[4] ⟨http://schiller.wustl.edu/DACLOD/daclod/⟩.

[5] ⟨http://www.gold.net/gold/⟩.

[6] ⟨http://web.nexor.co.uk/aliweb/⟩.

[7] The WWW index server currently covers most WWW pages in Hong Kong, and is publicly accessible at ⟨http://dbx.cs.ust.hk:8000⟩.

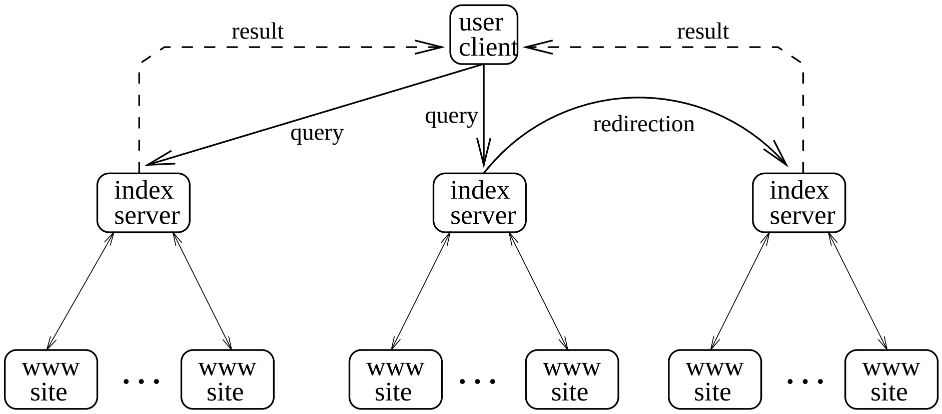

Figure 1: Multiserver organization and query routing.

that the search algorithm only searches for occurrences of words in the same word ordering as they are in the phrase. For example, the query "computer-science" matches only pages containing the word "computer" immediately followed by the word "science". Our index server also allows a user to save a query, along with an optional single-line comment, on the server so that other users can share his/her discovery. Saved queries are stored in a list of clickable query statements. By clicking on one of these statements, a user can resubmit the query to the index server.

The index server is a key component of the Wide-Area Information Discovery and Management (WIDM) project, which is being conducted at the Hong Kong University of Science and Technology. Figure 1 illustrates the global architecture of WIDM. An index server covers a number of WWW sites, which may be grouped by geographical area or institution. For instance, the index server currently operational covers most of the sites in Hong Kong; similar index servers may be developed for a particular institution (e.g., covering all of the NASA sites). A user client or browser can send a query to all of these servers or selectively to a subset of them and then combine the results it receives. In order to assist the user in finding the best server(s) for a query, the servers are further indexed by a set of representative keywords, referred to as server keywords, representing the topics covered by each of the servers. This approach is similar to that of other server indexing methods such as GLOSS [6] and directory of servers in WAIS [7]. Upon receiving a query, an index server checks whether it should process the query or redirect it to other index servers which can potentially give better results. A simple communication protocol is used by the servers to exchange server keywords with each other. The multiserver organization and mechanism are beyond the scope of this paper. In this paper, we describe the design and implementation of the index server as a stand-alone system.

It is worth noting that our goal is not to replace browsing with keyword search, but to supplement it. We believe that browsing is an intuitive and appealing paradigm for accessing information. However, the search paradigm can bring the user closer to potentially relevant sites or pages quickly, from which browsing can be used to further explore interesting sites

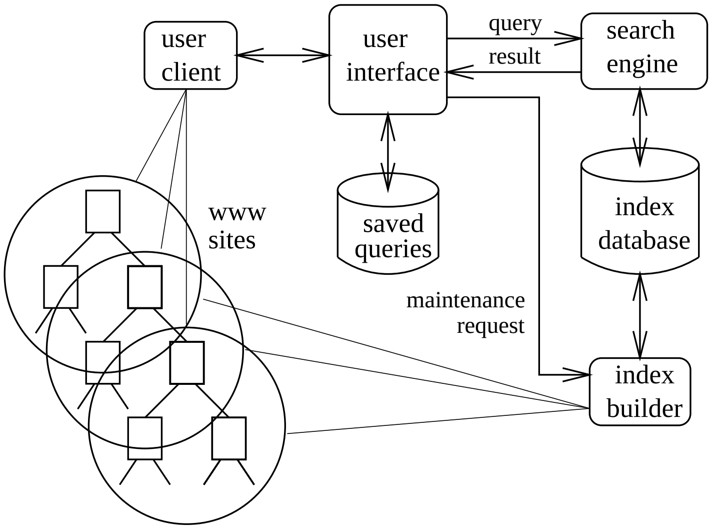

Figure 2: The WWW index server architecture.

or pages in the neighborhood. The objective of this paper is to explore and evaluate several different search strategies, including new ones which are designed to take advantage of hyperlink and other meta-information specific to the WWW environment.

The rest of the report is organized as follows. In section 2, we describe the design of our index server in general. Section 3 presents the search algorithms in detail. Section 4 discusses the experiment we conducted to evaluate the search algorithms. Finally, in section 5 we discuss the conclusions drawn from the results of the experiment, plans for future study, and some comparisons with other WWW index servers.

2 System Description

Our WWW index server consists of two main components: an index builder and a search engine. Figure 2 illustrates the system architecture.

2.1 Index Builder

The index builder is an autonomous WWW browser, also known as WWW robot, which communicates with WWW servers using HTTP (Hypertext Transfer Protocol [2]). It visits a given WWW site, traverses hyperlinks in a breadth-first manner, retrieves WWW pages, extracts keywords and hyperlink data from the pages, and inserts the keywords and hyperlink data into an index.

The index consists of a page-ID table, a keyword-ID table, a page-title table, a page modification-date table, a hyperlink table, and two index tables, namely, an inverted index and a forward index. The page-ID table maps each URL to a unique page-ID. The keyword-ID table maps each keyword to a unique keyword-ID. The page-title table maps every page-ID to the page's title. The page modification-date table maps every page-ID to the date when the page was last visited by the index builder. The hyperlink table maps each page-ID with two arrays of page-ID's, one array representing a list of pages referencing the page (incoming hyperlinks) and the other array representing a list of pages referenced by the page (outgoing hyperlinks). The inverted index table maps each keyword-ID to an array of ⟨page-ID, word-position⟩ pairs, each of which represents a page containing the keyword and the position of the word (order number) in the page. This word-position information is used in phrase searches mentioned in the introduction. Such information may also be useful for search strategies which take into account the distances between keywords. In this study, we do not investigate such strategies. The forward-index table maps a page-ID to an array of keyword-ID's representing keywords contained in the page. To obtain a fast access speed, hashing method is used to index each of these tables on the page-ID or keyword-ID attribute. The decision to store the index in separate tables instead of two large tables, one indexed by page-ID and the other indexed by keyword-ID, was based on its ease of maintenance and modularity. Moreover, the objective of this project is to develop algorithms that would work best in the WWW environment in terms of precision and recall. Thus, efficiency is not a primary concern at this stage of the project.

In extracting the keywords, we exclude high-frequency function words (stop-words), numbers, computer specific identifiers such as file-names, file directory paths, e-mail addresses, network host names, and HTML (Hypertext Markup Language [3]) tags. To reduce storage overhead, the index builder only indexes words enclosed by HTML tags indicating tokens such as page titles, headings, hyperlink anchor names, words in bold-face, words in italic, and words in the first sentence of every list item. We assume that a WWW author will use these tags only on important sentences or words in his/her WWW pages. Thus, these words make good page identifiers. This is one of the advantages of adopting SGML (Standard General Markup Language), of which HTML is a subset. Of course, there may be other important words which are not enclosed by any of the above HTML tags. Words chosen as keywords are then stemmed by removing their suffices.

Resources using protocols other than HTTP (FTP, Gopher, Telnet, etc.) or in formats other than HTML text file (non-inline image, sound, video, binary, and other text files), click-able image maps, and CGI scripts are indexed by the anchor texts referencing them.

Periodic maintenance of the index files is performed bi-weekly by the index builder. First, the index builder checks the validity of every URL entry in the database by sending a special request to the WWW server containing the page to check whether the page has been modified since the time it was last accessed by the index builder. This special request, known as HEAD request, is a feature supported by HTTP. Non-routine index maintenance is also supported. This is performed at night in response to user requests received during the day to index new pages (URL's) or re-index updated pages. Such user requests are facilitated by an electronic form provided by the index server.

Our WWW indexer robot has the capability of detecting looping paths, e.g., those caused by Unix symbolic links, and does not visit the same page more than once unless the page has been modified since the time it was last visited by the robot. The latter is made possible by supplying the last access date and time into the HTTP request.[8] As specified in the HTTP specification [2], the remote WWW server will not send the page content in response to the request if the page has not been modified since the specified time. Furthermore, the robot will not even send an HTTP request if the page was last accessed within the last 24 hours. This is to prevent the robot from sending more than one HTTP requests for the same page during a maintenance batch. To prevent the robot from endlessly roaming around from one server to the next, the robot accesses page at one site at a time and only references within the same site domain as that of the referencing page are traversed. Finally, the robot supports the proposed standard for robot exclusion[9] which prevents the robot from accessing places where, for various reasons, it is not welcome.

The index builder and the WWW robot is written in the C language. All index files are implemented using the GNU GDBM Database Manager library package [11].

2.2 Search Engine

The user interface to the search engine is a HTML form which can be invoked by standard WWW clients such as Mosaic and Netscape. The user types in the keywords and clicks on a submit button to send the query to the search engine.

Upon receiving a query, the search engine executes one of the ranking algorithms on the index database and produces a ranked list of URL's, from which the user can access the physical pages. Since the index database contains all of the information needed for ranking, the ranking process does not have to access to any WWW pages physically. If desired, the user can specify the maximum number of URL's to return and the ranking algorithm to use, instead of using the default setting. We discuss the ranking algorithms in detail in the next section.

It is clear that it is infeasible to search the WWW pages directly to compute the relevance scores without the help of the index. However, maintaining the currency of the index is a problem. We consider this a necessary tradeoff between speed and timeliness of the results. An immediate solution is to increase the frequency of index rebuild (notice that only the part of the index updated needs to be rebuilt). We are investigating within the WIDM project efficient ways of organizing the index to facilitate detection of updates and reorganization in a WWW server.

[8] Using the *If-Modified-Since* request header field.

[9] ⟨http://web.nexor.co.uk/users/mak/doc/robots/norobots.html⟩.

The search engine and its gateway mechanism run as CGI scripts (external programs executable by a WWW server on behalf of WWW clients using a standard mechanism called Common Gateway Interface). These scripts are written in C code. Our current server is running under NCSA HTTPD version 1.3 WWW server.

3 Ranking Algorithms

In this paper, we explore four ranking algorithms, namely, (1) Boolean Spread Activation, (2) Most-cited, (3) TFxIDF, and (4) Vector Spread Activation. The first two algorithms rely on WWW meta-information, namely, the hyperlink structure, for ranking the WWW pages without considering term frequencies. The TFxIDF method is based on word occurrence statistics [14], whereas the Vector Spread Activation method make use of both word occurrence statistics and the hyperlink structure in ranking.

3.1 Terminology and Notations

In the WWW environment, a document is commonly referred to as a WWW page. In this paper, we use the term *page* and *document* interchangeably. The following notations are used in the rest of this paper.

M : the number of query words.
Q_j : the j-th query word, for $(1 < j < M)$.
N : the number of WWW pages in the index database.
P_i : the i-th WWW page or its ID number for $1 < i < N$.
$R_{i,q}$: the relevance score of P_i with respect to query q.
$Li_{i,k}$: the occurrence of an incoming hyperlink from P_k to P_i,
where $Li_{i,k} = 1$ if such a hyperlink exists, or 0 otherwise.
$Lo_{i,k}$: the occurrence of an outgoing hyperlink from P_i to P_k,
where $Lo_{i,k} = 1$ if such a hyperlink exists, or 0 otherwise.
$C_{i,j}$: occurrence of Q_j in P_i,
where $C_{i,j} = 1$ if P_i contains Q_j, or 0 otherwise.

3.2 Description of the Algorithms

3.2.1 Boolean Spread Activation

This algorithm is based on the Boolean retrieval model, where retrieval is based solely on the occurrence or absence of keywords in the documents. We extend the Boolean model so that documents can be ranked based on the number of query words they contain. This strategy can be considered as a simple fuzzy set retrieval model [10] in contrast to the rigorous set membership of the Boolean retrieval model. More formally, document i is assigned a relevance score, $R_{i,q}$, with respect to query q.

$$R_{i,q} = \sum_{j=1}^{M} C_{i,j} \tag{1}$$

Notice that term frequency is not used in the formula. It is assumed that the query does not contain any disjunctions nor negations. Disjunctions can be removed by normalization or splitting the query into separate conjunctive clauses. Negations can be removed by disqualifying all documents containing the negated terms prior to the ranking.

The Boolean Spread Activation algorithm extends this strategy by propagating the occurrence of a query word in a document to its neighboring documents. This is possible in the WWW environment because a document can have hyperlink(s) to/from one or more other document(s), forming a network of documents. We assume that if two documents are linked to one another there must be some semantic relation(s) between the two. In other words, document P_i which does not contain query word Q_j but is linked to another document P_k containing Q_j is treated as if it contains Q_j. However, we assign P_i with a smaller score than if it actually contained Q_j. For each WWW page P_i, the algorithm assigns a relevance score with respect to query q as follows,

$$R_{i,q} = \sum_{j=1}^{M} I_{i,j} \tag{2}$$

where $I_{i,j}$ is defined as:

$$I_{i,j} = \begin{cases} c_1 & \text{if } C_{i,j} = 1 \\ c_2 & \text{if there exists } k \text{ such that } C_{k,j} = 1 \text{ and } Li_{i,k} + Lo_{i,k} > 0 \\ 0 & \text{otherwise} \end{cases}$$

c_1 and c_2 are constants ($c_1, c_2 > 0$) where $c_1 > c_2$. The algorithm is not sensitive to the values of these two constants. We prove this point empirically in the next section. In the implementation, we use $c_1 = 10$ and $c_2 = 1$.

3.2.2 Most-cited

As with Spread Activation, this algorithm takes advantage of information about hyperlinks between WWW pages. Each page is assigned a relevance score which is the sum of the number of query words contained in other pages citing, or having a hyperlink referring to, the page. More formally, the relevance score of page P_i with respect to query q is defined as:

$$R_{i,q} = \sum_{k=1,k\neq i}^{N} (Li_{i,k} \sum_{j=1}^{M} C_{k,j}) \tag{3}$$

The objective of this algorithm is to assign, among potentially relevant documents, larger scores to the referenced documents than to the referencing documents.

3.2.3 TFxIDF

The TFxIDF algorithm is based on the well known vector space model [14], which typically uses the cosine of the angle between the document and query vectors in a multi-dimensional space as the similarity measure. As described in [15], vector-length normalization can be applied when computing the relevance score, $R_{i,q}$, of page P_i with respect to query q:

$$R_{i,q} = \frac{\sum_{term_j \in q}(0.5 + 0.5\frac{TF_{i,j}}{TFmax_i})IDF_j}{\sqrt{\sum_{term_j \in P_i}(0.5 + 0.5\frac{TF_{i,j}}{TFmax_i})^2 (IDF_j)^2}} \tag{4}$$

where:

$TF_{i,j}$: the term frequency of Q_j in P_i
$TF_{i,max}$: the maximum term frequency of a keyword in P_i
IDF_j : $\log(N/\sum_{i=1}^{N} C_{i,j})$

Generally speaking, the relevance score of a document is the sum of the weights of the query terms that appear in the document, normalized by the Euclidean vector length of the document. The weight of a term is a function of the word's occurrence frequency (also called the term frequency) in the document and the number of documents containing the word in the collection (i.e., the inverse document frequency). This weighting function gives higher weights to terms which occur frequently in a small set of the documents.

The full vector space model is very expensive to implement, because the normalization factor is very expensive to compute. In our TFxIDF algorithm, the normalization factor is not used. That is, the relevance score is computed by:

$$R_{i,q} = \sum_{term_j \in q} (0.5 + 0.5 \frac{TF_{i,j}}{TF_{i,max}}) IDF_j \tag{5}$$

In the next section, we show empirically that this simplified method works better in the WWW environment than the vector space model with vector-length normalization.

3.3 Vector Spread Activation

This algorithm combines the vector space model and spread activation model. In this algorithm, each document is first assigned a relevance score using TFxIDF algorithm, then the score of a document is propagated to the documents it references. More formally, the algorithm assigns a relevance score to page P_i with respect to query q as follows.

$$R_{i,q} = S_{i,q} + \sum_{j=1, j \neq i}^{N} \alpha Li_{i,j} \cdot S_{j,q} \tag{6}$$

where $S_{i,q}$ is the TFxIDF score of P_i as defined in equation 5. α $(0 < \alpha < 1)$ is a constant link weight. Through an experiment (discussed in the next section), we found that 0.2 is the optimal value of α.

4 Retrieval Effectiveness

We evaluated the four algorithms on an index database covering pages at the Chinese University of Hong Kong (`CUHK.HK` domain). CUHK site was chosen because of its reasonable size and it has a diverse collection of information provided by the university's various departments, from the fields of humanity to engineering. We froze the entire WWW collection by copying all of the WWW pages from the site to a local disk. This was done on April 26, 1995. We recorded 2393 WWW pages including 1139 non-HTML pages (non-inline image, sound, video, click-able map, CGI script, and other text files), but excluding pages on the server *http://www.arch.cuhk.hk/* which was out of service that day. We then built an index from the full-text collection as described before.

56 test queries were used (for the list of the queries, see Appendix). The test queries were generated as follows. First, we selected at random 100 WWW pages from the collection.

Of these 100 pages, we removed pages of directory type (indices, catalogs, hotlists, tables of contents, etc.) and non-HTML pages, resulting in 56 pages. Next, for each of these 56 pages, we manually extracted keywords from it, selected keywords which can be used to construct a phrase representing the central concept (topic) of the page, and constructed a query from that phrase. Boolean OR operators, along with scope markers, were used in the queries to specify synonyms.

Given a query, the judgment on whether a WWW page is relevant or not is somewhat ambiguous, as it is very subjective and may vary across users. For this evaluation, we define relevance in the context of resource discovery, i.e., a WWW page is considered relevant to a query if, by accessing the page, the user can find a resource (URL) containing information pertinent to the query, or if the page itself is such a resource. We manually examined the entire collection to identify the relevant WWW pages for each query.

We computed the retrieval precision for each query q and each algorithm a, using the following formula:

$$Prec(a, q) = RetRel(a, q)/Ret(a, q) \tag{7}$$

where:

$Ret(a, q)$: the number of WWW pages retrieved by a on q.
$RetRel(a, q)$: the number of relevant pages retrieved by a on q.

For each query q and algorithm a, we obtained a recall-precision curve using the following procedure which is commonly used in evaluating the precision and recall of ranking algorithms [15]. Let $Rel(q)$ be the number of pages relevant to query q in the collection. First, we rank the pages using algorithm a based on q. We then identify the lowest ranked page which is relevant to q. Let the rank of the page be l. In other words, all relevant pages have ranks higher than or equal to l. If we consider the set of pages up to and including l as the result set, the recall is one since all relevant pages have been retrieved and the precision is $Rel(q)/l$. Next, we consider the next lowest ranked relevant document and recompute the precision and recall at that point. Let the rank be l'. The recall is $Rel(q) - 1/Rel(q)$ and the precision is $(Rel(q) - 1)/l'$. The precisions at recall values up to the recall value of $\frac{1}{Rel(q)}$ can be computed in a similar manner. From these recall-precision data points, we can obtain the precisions at recall values $0.0, 0.1, 0.2, ..., 1.0$ through interpolation.

Taking the average over all of the 56 test queries for each algorithm, we obtain the recall-precision curves as shown in figure 3, which shows that TFxIDF has the best retrieval performance, followed by Vector Spread Activation, Boolean Spread Activation and Most-cited.

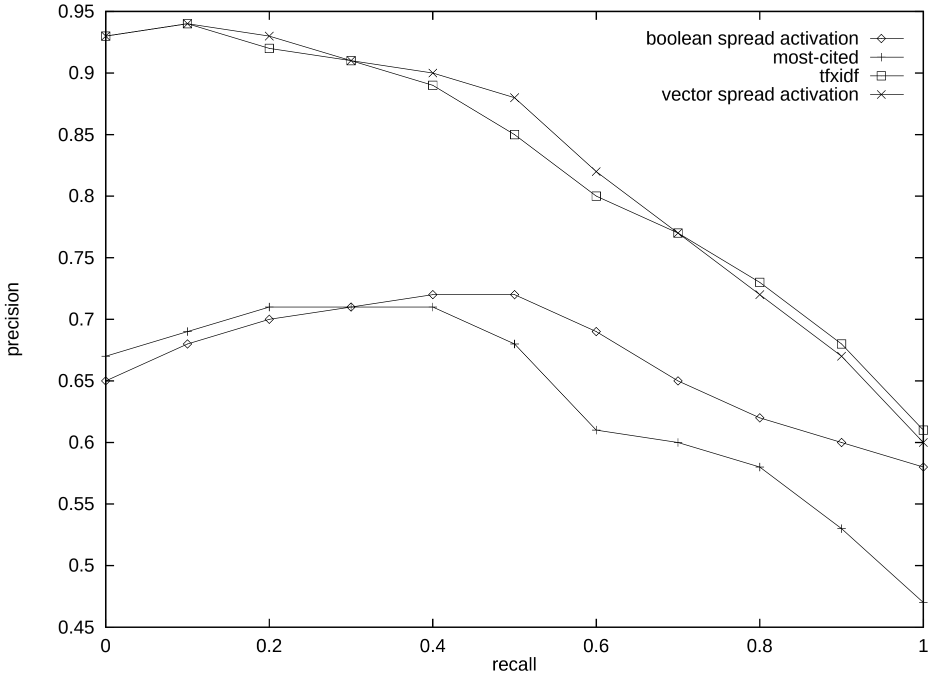

Figure 3: Recall-precision curve for each of the four search algorithms obtained by averaging the curves over the 56 test queries.

	boolean spread activation	most-cited	TFxIDF	vector spread activation
average precision	0.63	0.58	0.75	0.76

Table 1. The average precision across the 56 test queries for each search algorithm.

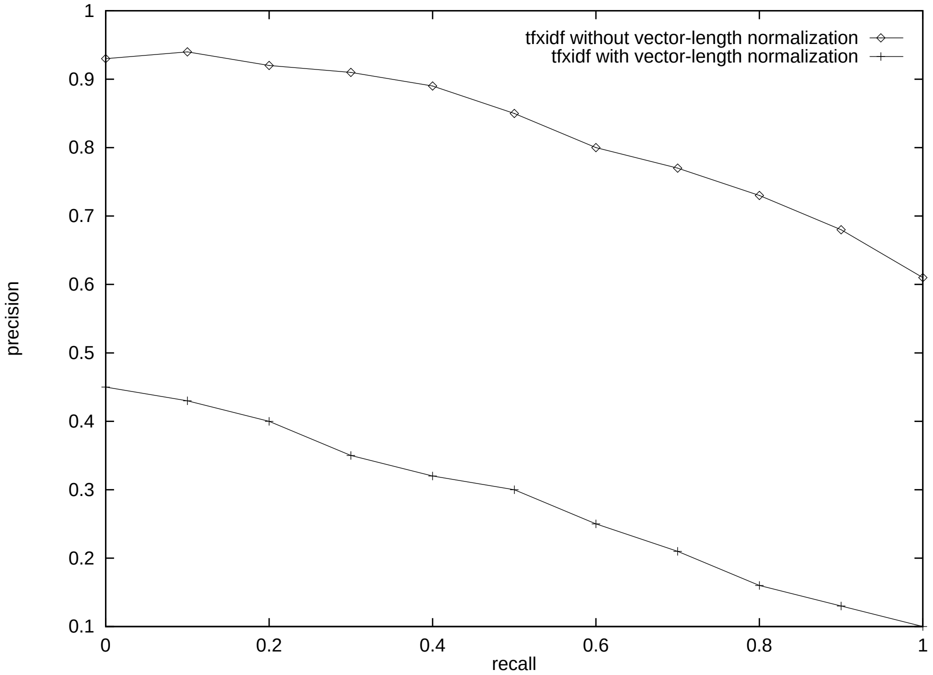

Figure 4: Recall-precision curve for TFxIDF with and without vector-length normalization obtained by averaging the curves over the 56 test queries.

Taking the average over all data points prior to the interpolation for each algorithm, we obtained the average precisions shown in table 1.

As mentioned in the previous section, we also conducted an experiment using the same 56 test queries to see whether vector-length normalization (see equation 4) could improve the retrieval effectiveness of TFxIDF. Figure 4 shows the recall-precision of TFxIDF with and without such a normalization. According to Salton and Buckley [15], vector-length normalization typically does not work well for short documents. This is consistent with our result since the average page length in the test collection is only 48.11 words (not including stop words and other words removed during the indexing process). It may be the case that vector-length normalization, in general, does not work for documents where the size of a segment with a coherent topic is small, e.g., a few sentences. In a WWW environment, it is common that a topic is only represented in a page by a hypertext anchor (click-able phrase).

To evaluate the sensitivity of Boolean Spread Activation and Vector Spread Activation algorithms to the choice of parameter values used, we conducted performance evaluation similar to the above on a range of parameter values.

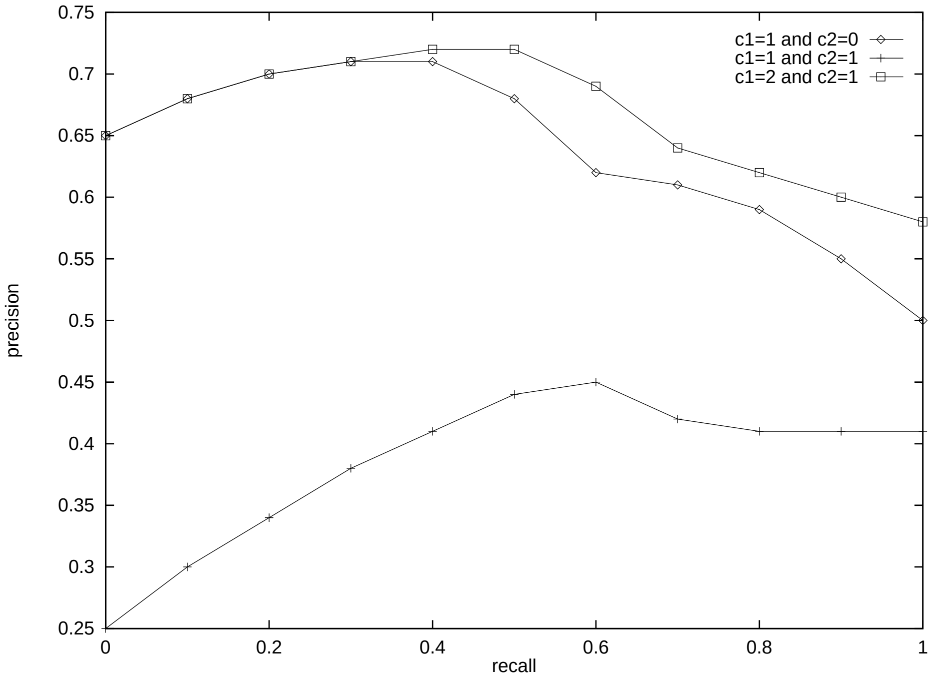

Figure 5: Recall-precision curves of Boolean Spread Activation algorithm on the 56 test queries with (c_1, c_2) value pairs of (1,0), (2,1) and (1,1).

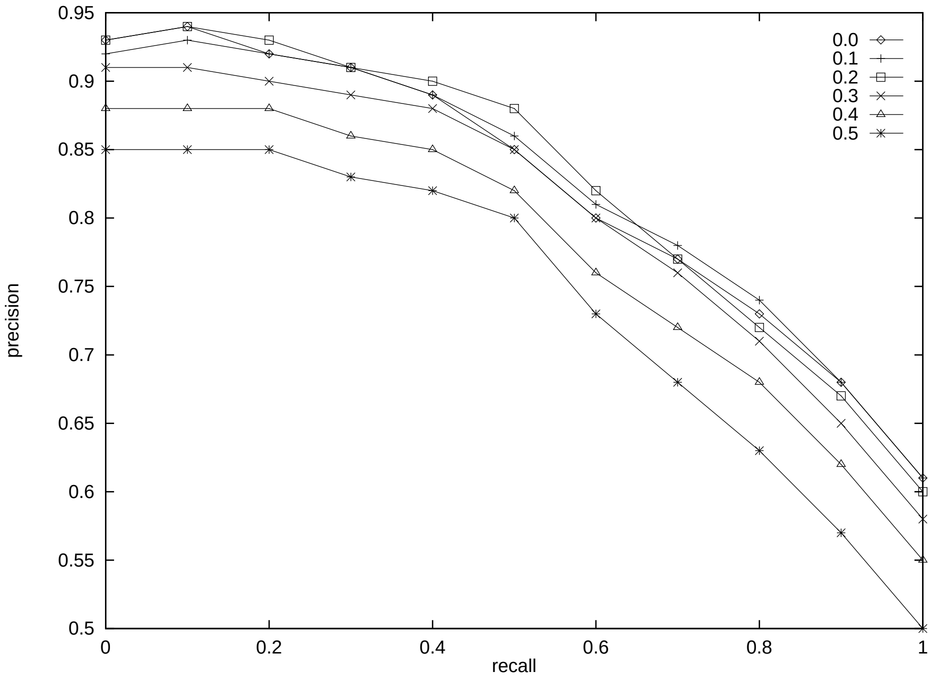

Figure 6: Recall-precision curves of Vector Spread Activation algorithm on the 56 test queries with α value of 0.0, 0.1, 0.2, 0.3, 0.4 and 0.5.

Figure 5 shows the recall-precision curves of Boolean Spread Activation on the 56 test queries using a number of c_1 and c_2 value combinations (see equation 2). By setting c_2 equals 0, that is by disabling the spread activation effect, we obtained the retrieval effectiveness of the fuzzy set retrieval model (see equation 1). Enabling the spread activation effect by setting $0 < c_2 < c_1$ improved the algorithm's recall. The algorithm produced the same recall-precision curve for (c_1, c_2) combinations of (2,1), (5,1), (10,1) and (10,5). Poor retrieval effectiveness resulted when c_1 was set equal to c_2, in which case many pages with various degrees of actual relevance had the same relevance score.

Figure 6 shows the recall-precision curves of Vector Spread Activation on the 56 test queries with α parameter value of 0.0 (TFxIDF without spread activation effect), 0.1, 0.2, 0.3, 0.4 and 0.5 (see equation 6). The best retrieval performance was achieved with α equals 0.2.

5 Discussion and Future Work

5.1 Conclusion

The results of the experiment in Section 4 provide us with some hints on the nature of WWW information retrieval environment. The relatively superior retrieval effectiveness of TFxIDF and Vector Spread Activation search algorithms shows that the concentration or distribution of words in a WWW page and across WWW pages is a good indicator of the page's contents or portions thereof. Algorithms which rely on meta-information such as hyperlinks information, while intuitive, did not perform as well. This indicates that the interconnectivity between WWW pages does not always indicate semantic relationships between the contents of the linked pages. The poor overall performances of these ranking algorithms may also be attributed to the fact that many WWW pages contain many different and unrelated topics. This is true for many home pages, index pages, what's-new pages, hotlist, directory and catalog pages which occur frequently in the WWW environment. It is worth noting that, since most of the test queries are phrases taken out from actual pages with some synonyms added, all of the algorithms showed good recalls.

5.2 Other Index Servers

There are many robot-based WWW index and search services on the Internet today.[10] However, among the well known robots, only a few employ full-text indexing, e.g., WebCrawler[11] [13], the Repository Based Software Engineering Project Spider[12] (RBSE) [5], and Lycos.[13] Other services index only page titles and first level headings (e.g., JumpStation[14]), or titles, headings and anchor hypertexts (e.g., World Wide Web Worm or WWWW[15]). Our index builder takes other HTML tokens such as words in bold-face or italics, the first sentence of every list item, in addition to titles, all-level headings and anchor hypertexts. Our scheme is a balance between full-text and title-only schemes by taking advantage of HTML meta-information as much as possible. On a WWW page containing mostly lists such as an index page, our scheme extracts nearly as much words as a full-text scheme.

Not many index servers use sophisticated information retrieval models beyond a simple Boolean model or pattern matching based on Unix *egrep* (e.g., WWWW), with exception of the RBSE, the WebCrawler, and Lycos. The RBSE uses WAIS search engine which ranks WWW pages based on the occurrence frequencies or term frequency (TF) of the query words [12. The WebCrawler and Lycos, as with our index server, rank the pages based

[10] A list of WWW robots can be found at ⟨http://web.nexor.co.uk/mak/doc/robots/active.html⟩.

[11] ⟨http://webcrawler.cs.washington.edu/WebCrawler/Home.html⟩.

[12] ⟨http://rbse.jsc.nasa.gov/eichmann/urlsearch.html⟩.

[13] ⟨http://lycos.cs.cmu.edu/⟩.

[14] ⟨http://www.stir.ac.uk/jsbin/js⟩.

[15] ⟨http://www.cs.colorado.edu/home/mcbryan/WWWW.html⟩.

on term frequency and inverse document frequency (TFxIDF). As of this writing, we are not aware of any attempt to quantitatively measure the retrieval effectiveness of search algorithms in the WWW environment as in the present work.

5.3 Future Work

As part of an on-going research project, our next step in developing the WWW index server is to study the effectiveness of other information retrieval techniques [14], including relevance feedback, automatic query expansion, and automatic categorization and clustering in the WWW environment. We are using methods based on word distribution statistics, which has been proven useful in this paper, to analyze and group WWW pages as well as keywords into conceptual clusters. By identifying keyword clusters, we can expand a query by adding more keywords belonging to the same cluster as that of the keywords in the original query. Page clusters can be used to identify other pages sharing the same cluster with the pages judged as relevant by the user. We are also exploring other uses of page clustering such as automatic catalog generation, index database reduction/compression and multi-server indexing.

6 References

1. Berners-Lee, T., "Uniform Resource Locators," *Internet Working Draft*, 1 January 1994.

2. Berners-Lee, T., "Hypertext Transfer Protocol," *Internet Working Draft*, 5 November 1993.

3. Berners-Lee, T., and Connolly, D., "Hypertext Markup Language," *Internet Working Draft*, 13 July 1993.

4. Berners-Lee, T., Cailliau, R., Groff, J., and Pollermann, B., "World Wide Web: The Information Universe," *Electronic Networking: Research, Applications and Policy*, 1(2), 1992.

5. Eichmann, D., "The RBSE Spider - Balanching Effective Search against Web Load," In *Proceedings of the First International Conference on the World Wide Web*, Geneva, Switzerland, May 1994.

6. Gravano, L., Tomasic, A., and Garcia-Molina, H., "The Efficacy of GLOSS for the Text Database Discovery Problem," Technical Report STAN-CS-TR-93-2, Stanford University, October 1993.

7. Kahle, B., and Medlar, A., "An Information System for Corporate Users: Wide Area InformatiON Servers," Technical Report TMC-199, Thinking Machines, Inc., April 1991.

8. Koster, M., "ALIWEB: Archie-like Indexing in the Web," *Computer Networks and ISDN Systems*, 27(2), pp. 175-182, 1994.

9. Krol, E., *The Whole Internet User's Guide &Catalog*, O'Reilly &Associates, Sebastopol CA, 1992.

10. Meadow, C., *Text Information Retrieval Systems*, San Diego Academic Press, San Diego CA, 1992.

11. Nelson, P., "GDBM - The GNU Database Manager," online manual pages, Version 1.7.3, 1990.

12. Pfeifer, U., Fuhr, N., and Huynh, T., "Searching Structured Documents with the Enhanced Retrieval Functionality of freeWais-sf and SFgate," *Computer Networks and ISDN Systems*, 27(7), pp. 1027-1036, 1995.

13. Pinkerton, B., "Finding What People Want: Experiences with the WebCrawler," In *Proceedings of the First International Conference on the World Wide Web*, Geneva, Switzerland, May 1994.

14. Salton, G., *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison-Wesley, Reading MA, 1989.

15. Salton, G., and Buckley, C., "Term-Weighting Approaches in Automatic Text Retrieval," *Information Processing &Management*, 24(5), pp. 513-523, 1988.

Appendix
Test Queries

query no.	query
1	computer (data | network | system) security
2	opening hours
3	campus backbone network
4	www |(world wide web) symposium
5	sino plaza
6	victoria harbour
7	times square
8	bank of china
9	victoria park
10	hong kong airport
11	chinese ethnic (culture | cultural | identity)
12	asia pacific studies
13	institute of humanities studies
14	institute of chinese studies
15	(culture | cultural) change
16	(gender | sex) issues
17	marriage family
18	political anthropology
19	chinese (document | information | text)(classification | retrieval)
20	(computer | machine) vision
21	dynamic load balancing
22	(knowledge acquisition)|(machine learning)
23	computer aided (design | engineering | manufacturing)
24	computer chess championships
25	(navigation | navigational) systems
26	(asynchronous transfer mode)| atm
27	(cellular | mobile) communication (network | system)
28	hong kong market (research | analysis | study)
29	marketing theory
30	sociocultural factors of marketing
31	(strategic | strategy) marketing
32	partial differential equations
33	ring theory
34	image deblurring
35	iterative methods

Test Queries
(continued)

query no.	query
36	public lectures
37	chinese music theory
38	music composition
39	(electronic | digital) music (studio | recording | production)
40	chinese (music | musical)(instrument | instrumental)
41	cantonese opera
42	english (spoken | speaking) (learning | course)
43	multimedia production
44	hong kong cultural studies
45	business (policy | strategy)
46	business law
47	dai qing
48	hong kong (public | mass) transportation
49	chinese herbal tea
50	restaurant licence
51	bus fares
52	concubines
53	sun law
54	educational television service
55	english radio programme
56	fm radio station