CWI Quarterly

Volume 8 (4) 1995, pp. 289 — 306

Finding Information on the Web

P.M.E. De Bra

CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands

e-mail: Paul.de.Bra@cwi.nl

Information Systems Section

Department of Computing Science

Eindhoven University of Technology

PO Box 513, 5600 MB Eindhoven, The Netherlands

e-mail: debra@win.tue.nl

The World Wide Web contains a vast amount of information on every imaginable subject. Most of this information is very stable, but some of it is generated dynamically and therefore short-lived. The lack of a complete and usable catalog or index however makes it difficult for users to find the information they want.

This paper gives an overview of the techniques used to find information on the Web, and the search facilities based on them. A better insight in the capabilities and limitations of the different search tools can help users select the appropriate tool for each task.

1. Introduction

Since its conception around 1990 the World Wide Web has been growing at an exponential rate (called the *slow bang* by Tim Berners Lee), to become the largest information space on the Internet, and probably also in the World. The loose network structure that makes it easy for individual organizations to become part of the Web and provide information on their own server is also the source of the largest information nightmare of the world: for many users it has become very difficult, if not impossible, to find information on the Web, even when one knows it exists.

Information retrieval is a research field with a long history. (See [1] and [11] for an overview.) The process of finding information can be divided into three stages:

1. **finding documents:** the Web consists of millions of documents, distributed over tens of thousands of servers; it may be difficult to access all potentially interesting documents.
2. **formulating queries:** the user needs to express exactly what kind of information she is looking for.
3. **determining relevance:** the system must determine whether a document contains the information the user is looking for.

Traditional information retrieval research is primarily concerned with the second and third stage, but for retrieval on the Web the first stage is the largest bottleneck.

In order to answer the user's queries in a timely fashion, most search tools use a specially prepared database instead of searching the documents on the fly. Depending on the technique used for generating this database certain types of questions can or cannot be answered. The search tools that are publicly available on the Internet use different types of databases and different techniques for accessing the documents on the Web. As a result they can answer different questions, and even on the same question they may give different answers. This paper gives an overview of the freely available search tools, the techniques used for accessing the Web and for indexing the information. It is the aim of this paper to provide the reader with a better insight in the information retrieval problem on the Web, and in the merits and limitations of the available search tools.

In this paper we will use the following example of a search: at the Eindhoven University of Technology we have developed a course on hypertext and hypermedia. The complete text of this course has been available on the Web since early 1994. It consists of 163 small documents, with many links between them.[1] We describe our experiences with finding the first page of this course using different search tools on the Web. We also describe our experience in locating the home page of the author (of the course and of this paper), Paul De Bra, and of another researcher, Ad Aerts.

The structure of this paper is as follows: in Section 2 we briefly sketch the overall hypertext structure of the World Wide Web. Section 3 describes how "robots" are used to find some, most or all documents on the Web. Section 4 describes how index databases store (descriptions of) documents and how search engines enable users to find information. This section also includes our experience in locating the hypermedia course and the two people's homepages using the most popular search tools. Section 5 describes Harvest, an example of a novel and distributed approach to solving the main problems with information retrieval on the Web. In Section 6 we give some concluding advice on which tool to use for which kind of search request.

[1] The address is `http://www.win.tue.nl/2L670/`.

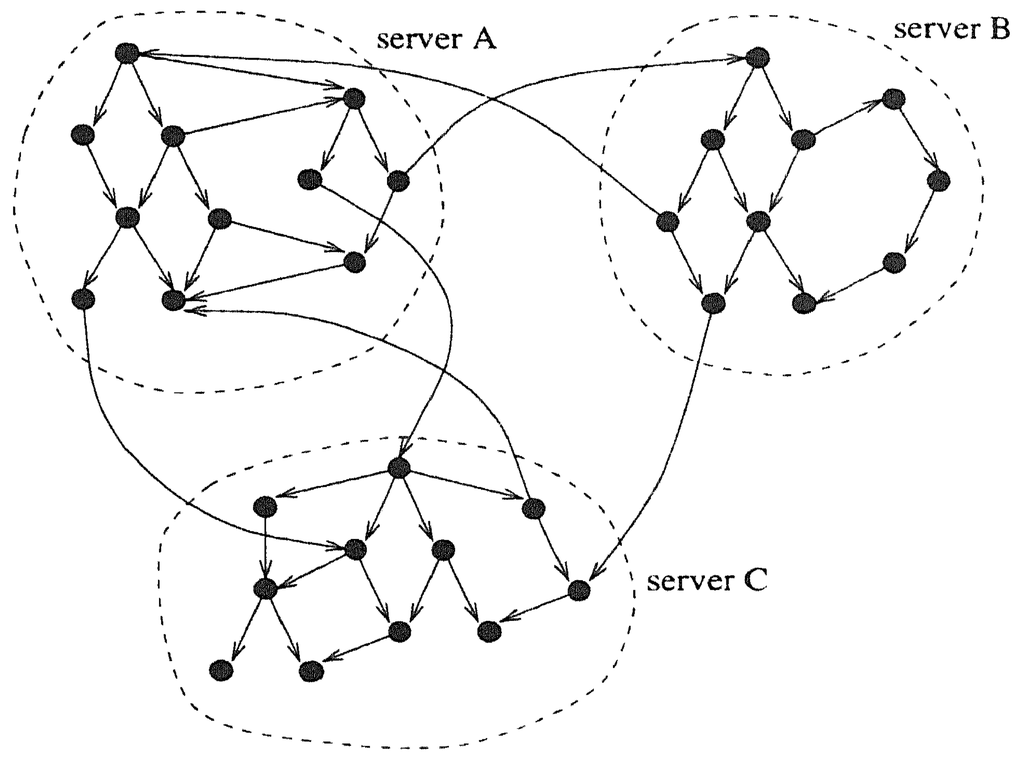

FIGURE 1. Graph structure of the Web.

2. The Hypertext Structure of the Web

Hypertext is defined by Shneiderman and Kearsley [14] as *a database that has active cross-references and allows the reader to "jump" to other parts of the database as desired.* This definition suits the World Wide Web well: documents have active links to each other, meaning that the user can jump from one document to another by following these links. Users can also jump directly to documents by means of the name and location of a document. The Web uses *Universal Resource Locators* (URLs) to address documents. URLs are most useful when the user knows the location of a desired document. When searching for information one normally wants to find out the location of documents containing that information, hence URLs are the answers to queries.

Search tools have to follow links to find where the documents are, by traversing the Web. Figure 1 shows a graph structure like that of the Web. A few servers are shown, having many connections between documents on the same server, and a few connections to each other.

Apart from a rough idea of the graph structure of the Web, Figure 1 also illustrates that although the depicted graph is completely connected it is impossible to reach all the nodes of the graph from a single starting point and by following links forwards. Even when it is possible to jump back to nodes that were visited earlier, a feature offered by most Web browsers, some nodes still remain unreachable. The same problem exists in the "real" Web: the entire Web cannot be reached by following links alone. Different "tricks" are needed in order to find a set of starting points from which the entire Web can be reached.

A lot of information on the Internet is available from ftp servers (ftp means file transfer protocol), and around 1991 and 1992 the Gopher servers also be-

came popular. Ftp servers are strictly hierarchical because they access files directly from the (Unix) file system. Gopher servers are also hierarchical, using a menu system, but they may also contain menu items that point to information on other servers. Finding all documents or files on such hierarchical servers is much easier (once you know where the servers are) than finding the documents on a World Wide Web server.

The RBSE spider project [7], [8] from the University of Houston has investigated the World Wide Web structure by counting the number of links from and to each document. The more links there are to a document the easier it is to find that document by following links. Even when visiting only a small part of the Web, chances are that a pointer to such a document is found. When there are many documents to which there are only a few links, a large part of the Web must be visited before a pointer to such documents is found. The RBSE spider found that for 59 % of the documents on the Web there is only one link pointing to them, and for 96 % there are no more than five links. This means that most documents on the Web are hard to find by means of navigation.

3. World Wide Web Robots or Spiders

Browsing, or "surfing" the Web, consists of starting at a known (URL of a) document and following links to other documents at will. A (graphical) browser program shows where anchors to links are in a document, for instance by underlining them and/or displaying them in a different color than the rest of the document. Documents on the Web are written using HTML, the HyperText Markup Language. Links are embedded in documents using HTML anchor tags that contain the URL of the destination of the link. Users who remember URLs (or put them in a hotlist) can tell the browser to jump directly to a document with a given URL.

In order to search the Web for information, or to simply gather documents to build an index database, one needs to run a program that retrieves documents from the Web in much the same way as a surfing user retrieves documents by means of the browser. These special programs are called *robots* or *spiders.* Martijn Koster (formerly at Nexor, now at WebCrawler) maintains a list of known robots, and a mailing list for robot creators and users.

Although conceptually the robots or spiders appear to wander around the Web, they only wander virtually, because they really remain located on the same computer. Names like "WebWanderer" and "World Wide Web Worm" may suggest programs that invade computers all over the Web to extract information and send it back to their home base, but all they really do is retrieve documents from different locations on the Web to the computer they reside on. Hence they do not impose any danger like the infamous Internet Worm [6] back in 1988.

3.1. Robot Algorithm

All robots use the following algorithm for retrieving documents from the Web:

1. The algorithm uses a list of known URLs. This list contains at least one URL to start with.
2. A URL is taken from the list (using heuristics that are different for each robot), and the corresponding document is retrieved from the Web.
3. The document is parsed to retrieve information for the index database and to extract the embedded links to other documents.
4. The URLs of the links found in the document are added to the list of known URLs. (The order and position in which URLs are added to the list differs between robots.)
5. If the list is empty or some limit is exceeded (number of documents retrieved, size of the index database, time elapsed since startup, etc.) the algorithm stops. Otherwise the algorithm continues at step 2.

3.2. Getting the Robot Started

As Figure 1 shows, no single starting point is sufficient for finding the whole World Wide Web. Therefore, the composition of the initial list of known URLs is an important step towards finding as much of the Web as possible. Also, when using a robot for finding information on a specific topic, an initial list of documents relevant to that topic is a big step forwards.

The World Wide Web organization maintains an official list of Web servers[2]. This list contains pointers to sublists for each country. None of these lists are really complete. The official list for the Netherlands[3] and the unofficial graphical overview, called the "Dutch Home Page"[4] usually contain different server addresses. Furthermore, most organizations list only one server, while they have several others that can be reached through the one listed server. As a starting point for locating most of the Web such lists are extremely valuable. Although they don't provide an address for every server, they bring a robot to the neighborhood of nearly every server on the Web.

New and exciting information often appears on servers that are not yet registered with the official or unofficial lists. In order to find this information some robot sites monitor a number of mailing lists and Usenet Netnews groups. Announcements of new services or reports from interested users often appear on netnews long before there are any links to the documents on known Web servers.

3.3. Navigation Strategies

The processes of taking URLs from the list and adding new URLs to it determine the navigation strategy of the robot. If newly found URLs are always added to the same side of the list as where URLs are picked for retrieving the next document the robot navigates in a depth-first manner. If new URLs are

[2] At `http://www.w3.org/hypertext/DataSources/WWW/Servers.html`
[3] At `http://www.nic.surfnet.nl/nlmenu.eng/w3all.html`
[4] At `http://www.eeb.ele.tue.nl/map/netherlands.html/`

added to one end of the list and URLs are picked from the other end the robot performs breadth-first navigation.

Most robots exhibit a behavior somewhere in between these two extremes, in order to benefit from the advantages from both strategies without suffering from their drawbacks.

- The **depth-first** strategy, as investigated in [15], gives the best overall distribution of URLs over the Web, which is important when only a relatively small part of the Web can be retrieved. Depth-first navigation also generates the danger of leading a robot into infinitely recursive document trees on servers that generate documents on the fly. Many documents contain links to themselves or to different documents that are generated using the same URL. A robot that uses depth-first navigation must take special precautions to avoid entering such infinite loops.
- The **breadth-first** strategy, when used with an initial list like the official registry of servers, yields excellent results at first, because it reaches many different servers. In general however this strategy is less efficient at penetrating the Web deeply, going far beyond the starting points. Also, because links are taken from the list in the same order as they are put in, all the links embedded in a single document are followed consecutively. In many cases these links point to documents on the same server. Thus, breadth-first navigation may lead to periods of putting a heavy load on a single server, which is the single most important reason for server managers (webmasters) disliking robots roaming their site.

Several robot-based search databases use multiple robots (sometimes called *agents*) in parallel, to achieve better overall retrieval performance. Although such actions to receive a larger share of the total network bandwidth are generally considered rude and unacceptable behavior, most robots make sure not to retrieve more than one document from the same server at once, or even consecutively. (The Netscape Navigator exhibits an even more rude behavior by loading in-line images in parallel, from the same server.) Using multiple agents in parallel not only speeds up the overall process, it also avoids blocking when a robot encounteres a very slow link or server.

Web server maintainers can help robot builders to avoid pitfalls like infinite loops or useless documents like the contents of a cache or mirror of another server. Most robots comply to the robot-exclusion scheme[5] by which all documents and/or directories listed in a robot-exclusion file (robots.txt) on the Web server are ignored (i.e. not retrieved) by a visiting robot.

3.4. Robot Performance Limitations

Ideally a robot would load the entire Web in a relatively short period of time, to ensure that the retrieved documents are up to date, and that no recent additions are missed. Given that the Internet network will always be limited

[5] See `http://info.webcrawler.com/mak/projects/robots/norobots.html`

and very busy during the week, reloading the whole World Wide Web over the weekend seems like a reasonable goal. Unfortunately, no robot is able to achieve this, regardless the effort put in by its creators, in terms of computer and network hardware.

The Lycos team (formerly at CMU, the Carnegie Mellon University, but now operating as an independent company) has been retrieving Web documents for over a year, thereby incrementally building and rebuilding an index database. During that period their robots have found an ever increasing number of documents, now getting to about 20 million, for a total of well over one hundred gigabytes of (textual) information. In order to retrieve that much information in a single weekend (of about 50 hours), the robots would need to be able to download at least 100 documents per second, at a sustained rate of over 5.5 Mbps (million bits per second). This transfer rate is currently impossible, both because of the overhead incurred by the TCP/IP protocol used on the Internet, the speed of light which limits round-trip packet times on intercontinental connections, and the numerous Web servers connected through slow data lines. Assuming that the average transfer rate obtained over the Web is around 1 Kbps per connection, a total of 5.500 simultaneous connections are needed to achieve the 5.5 Mbps transfer rate, but currently most network switches cannot handle 5.500 simultaneous TCP/IP connections.

Because of these limitations robot maintainers have taken different approaches: Lycos tries to download and reload the entire Web as often as possible, meaning every once in a few months up to about a year. WebCrawler on the other hand tries to download as much as possible, from as many different servers as possible, in a weekend's time, thereby achieving a more limited coverage, but with more up to date information.

4. Index Databases and Search Engines

The Web contains over one hundred gigabytes of textual information. Assuming one has that much diskspace available, it would still be impractical to search sequentially through that many gigabytes for documents about a certain topic. Therefore, index databases are built, that link topics or words directly to relevant documents.

An index database works somewhat like an inverted file. A normal text file contains lines of text. Given a line number one has easy and direct access to the words on that line. In an inverted file, given a word one has direct access to the numbers of the lines containing that word. The Web is a text database that provides easy access to the contents of a document, given its name. An index database tries to provide access to (names or URLs of) documents, given a description of their contents, e.g. a few words that must occur in the documents.

Generating index databases, both in general and for the Web, is difficult for a number of reasons:

– The description of the contents of documents must be such that it is easy

for the user to give these descriptions, and easy for the system to match documents and descriptions. Most systems use logical combinations of words that should or should not occur in documents. Often a subject cannot be described by a single word or a combination of words. When a phrase (consisting of several words) is needed the system must be able to decide whether the individual words of the phrase occur in the right order, and adjacent to each other. Sometimes a subject is best described by means of an example article or abstract. Some information retrieval programs can match documents for similarity. Infoseek is a publicly available service that offers this facility for the Web.

- When using words to describe subjects or topics, the system must account for synonyms, words which are almost always related, and with stemming and stripping of suffixes. This is difficult to do correctly in general. Words like "fact" and "factual" describe the same topic, but that doesn't mean that the suffix "ual" can always be deleted. When deleting "ual" from "equal" no English word is left. Stemming is even more difficult: the system must know that "absorb" and "absorpt" have the same stem ("absorb") although the stem need not be part of the word. Synonyms are the most difficult to handle because context may be needed in order to determine whether they are synonyms in a given document or not. The words "cat" and "pussy" are not synonymous in all contexts for instance.
- Because of the gigantic size of the Web the number of words or terms that are used to describe a document must be limited. Words that occur in very many documents are not useful, and neither are words that occur too infrequently (like only once). The Lycos system appears to use only about 20 words per document to identify each of 20 million Web documents. This has already lead to a database of close to 10 gigabytes.

Four of the most popular search tools on the Web use very different index databases which, combined with different techniques for filling them using robots, lead to different strengths and weaknesses:

- ALIWEB [9] uses an approach similar to the Archie search tool for FTP servers. Each webmaster must create a file containing a description of what information can be found on the Web server. A robot retrieves these files once every so often (daily) and rebuilds its database. This scheme relies on man-made summary files, and therefore has gained only limited acceptance. However, because it generates a small database, it is easy to mirror onto different sites, thus distributing the load of search operations. There are about ten mirror sites for ALIWEB.
- Yahoo started as a university project (at Stanford) for creating a subject catalog of the Web. The database is accessible through a hierarchical menu system of topics and subtopics. The database can also be searched by means of (topic) keywords. Most of the database is generated manually. Webmasters who wish to have some of their URLs listed by Yahoo can send

a request with a description of the documents and the subject category they belong in.

— WebCrawler [10] started as a small university project of a graduate student (Brian Pinkerton) at the Washington University. WebCrawler retrieves as many documents as possible, from as many different servers as possible, during the weekend. The documents are indexed completely, using NextStep's Indexing kit. WebCrawler works well for searching for topics that are typical for many documents on the same server. Because the WebCrawler database may contain only a few documents from a single server, it is likely to contain information that is very typical for that server, but unlikely to contain information that occurs in only one or a few documents on that server. WebCrawler may not work well for finding personal homepages for instance, but is very useful for locating departmental servers based on the name of a faculty and institute. WebCrawler uses the breadth-first navigation strategy for finding documents. Therefore, documents that are many links away from popular or starting pages of a Web server are unlikely to be found by WebCrawler. The hypermedia course, 2L670, is mentioned on several pages on our department's Web server. Therefore we expect WebCrawler to be able to find this document, while we expect WebCrawler to be unable to find the home page of some of our employees.
— Lycos started as a project at CMU. Lycos tries to index all documents in the entire World Wide Web, and also the documents reachable through Gopher and ftp servers. To keep the total size of the index database within reasonable limits the Lycos index database contains only a few (about 20) words per document. Hence, the fact that Lycos has indexed almost all documents on the Web does not mean that it is easy to find a specific document using Lycos. Lycos will only find documents for which the given words are typical. Words that occur only once in a document, and which may characterize that document exactly, may not have been selected by Lycos for inclusion in its database. Given a few good keywords, Lycos will be able to find many documents about these topics. We expect Lycos to be able to find both the hypermedia course 2L670 and the home pages of our employees. Because Lycos has been indexing documents for a long time, many of the URLs returned by Lycos may point to documents that no longer exist or that have moved. Also, because many Web sites include gateways to common databases like Unix manual pages, Lycos has indexed thousands of copies of these popular documents.

There are many other search tools like the four mentioned above.

— The JumpStation maintains an index database of only headers and titles of documents. The B94] maintains an index database of (3 million) titles, URLs and text used for links. Like ALIWEB, these tools do not index the contents of Web documents.
— The TradeWave Galaxy (Formerly EINet Galaxy), and the Global Network Navigator are menu-based catalogs, much like Yahoo. (The Global

Network Navigator however does not offer searching for keywords, while The TradeWave Galaxy and Yahoo do.) City Net and the Virtual Tourist provide an index based on geographic location instead on subject. They are great to find a server if you know where it is but not how it is called.

- RBSE's Url Database [7] provides searching in 36.000 documents indexed using WAIS. This makes it comparable to WebCrawler, but WebCrawler is better maintained. Infoseek offers searching in over one million documents. It also offers commercial search services. It aims at completeness, like Lycos, but must contain more information per document in order to perform its similarity search. Alta Vista is the most recent major search database, and is offered as a free service by Digital. It claims to contain a full-text index of over 16 million documents.

4.1. Boolean Operators

Sometimes a document can be described perfectly by giving a few words that must occur in them. Sometimes a number of words are used to describe a topic, and a relevant document may contain some, but perhaps not all of them. Sometimes one also knows words that should not occur in relevant documents. Search tools offer boolean combinations of words to enable the user to describe the documents she is looking for.

The search tools for the Web are quite primitive: they ask for a number of keywords, and let the user select whether some or all should occur, i.e. they offer a choice of boolean *and* and *or*. Lycos, the TradeWave Galaxy, WebCrawler and the World Wide Web Worm all offer this choice. With Infoseek one can prepend a + or - sign to words to indicate that they must or must not occur in documents in order to consider them relevant.

Lycos offers more options than just *and* and *or*. A parameter can be selected to tell Lycos to match at least a certain number (between 2 and 7) of terms. With other search tools one may search for documents containing any or all of the hypermedia course's codes 2L670, INF725 and INF706, but with Lycos one can search for documents containing at least two of these three codes.

Note that none of the above systems offer a complete range of boolean formulae. Only Alta Vista lets you combine *and*, *or* and *not* and use parentheses to generate any boolean combination you want, for instance "(A or B) and not (C or D)".

4.2. Proximity

When given several search terms (words), none of the available search tools offer the user complete control over the importance of words occurring in a certain order, or near each other, or in the same structural element of the document (except for searching within titles or URLs). However, this does not mean that the proximity of words has no influence on the documents that are found or on their ranking. Not many details about this are made public.

In order to take proximity of words into account an index database needs to know the location of words in each document. This makes the database a lot larger than without proximity. Instead of really using proximity, the Lycos system simply gives preference to discriminating words that occur close to the beginning of a document. This not only provides for a limited form of proximity, but also increases the importance given to the title of a document, without the need for analyzing the structure of the document to find out what the title is.

Some search tools provide *adjacency* search by using the given words as one large search term. When searching for the title of the hypermedia course this is useful, but it is not a complete replacement for proximity in general.

Infoseek offers the best possibilities to control proximity and word order. The search term `"user interface"` (with quotes) means that the two words must be adjacent to each other in the given order. The term `user-interface` means that the two words should appear in the given order, and near each other. `[user interface]` means that the two words should appear near each other, but in any order.

Alta Vista uses the keyword `near` to indicate that two terms should not be more than ten words apart in the document. (Alta Vista is the only tool that actually explains what the exact meaning of its proximity operator is.)

4.3. Weighing of Terms

Not all words are equally important in a document. Even when the user provides several (four or more) meaningful words, a search engine may find many documents containing these words. In order to rank documents (from most to least relevant) the engine needs to guess which words are more important than others, both in the documents and in the user's search request.

Figure 2 shows how words (or terms) should be weighed. Words that occur very frequently are not useful for finding relevant documents, and words that occur very infrequently may not be typical for the document either.

It is important to consider both the frequency of word occurrence in a single document and in the entire document database. For the hypermedia course, the codes 2L670, INF725 and INF706 are not more or less common than many others, but in the entire Web these words are ideal for discriminating between the course and other documents. When selecting a few words to characterize this document in the Web these terms are ideal, whereas to characterize this document in general (not knowing which words are typical in the Web) these words may fall below the lower cut-off because they occur too infrequently in the document. Lycos for instance has rejected the codes 2L670 and INF725 which occur only once in the first page of the course, but has kept INF706 which occurs twice.

Ideally the user would be given the option to modify the system's default weight for each of the search terms. When searching for the "Unix command cat", the word "cat" is very important because the user does not want informa-

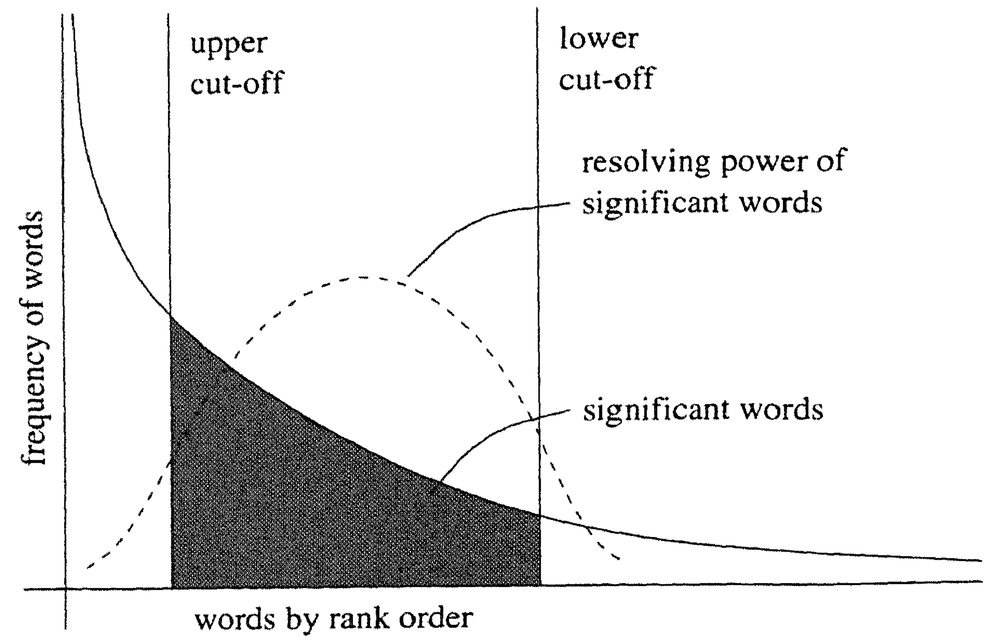

FIGURE 2. Resolving power of words.

tion on other Unix commands. The system however may decide that "cat" is not an important word, because there are many documents about cats. None of the search tools for the Web offer the choice of altering the weights of search terms. But on some systems, including WebCrawler, the user can use an undocumented feature that gives a higher weight to words that are repeated in the request. So the request for the "Unix command cat" can be modified to "Unix command cat cat cat" to indicate that finding "cat" is most important.

4.4. Searching for Parts or Expressions

Most search engines on the Web are not very intelligent about stripping and stemming of words, or about analyzing the structure of documents. They offer an option to find exact word matches or just substrings, putting most of this burden on the user. ALIWEB is a search engine that allows regular expressions in addition to just words. This means that ALIWEB must contain the complete text of the summaries it contains. Harvest [12] also has the ability to perform regular expression searching, and even approximate searching, because its *Glimpse* index database format supports those. The on-line Fish-Search [3],[4],[5] offers the same regular and approximate expression search, as well as a number of others.

In order to search for structural parts of documents an index database would need to analyze the HTML syntax. A few search engines, like the World Wide Web Worm and the TradeWave Galaxy, offer the option of searching document titles only, or link text only. The title is easy to detect (looking for the <;TITLE>; tag), and the links must be found in order to be able to find more Web documents anyway. Searching for other structural parts, like figure captions or subsection titles is not possible with any of the publicly available search engines.

4.5. Example Search Results

As an experiment we tried to locate the hypermedia course in four different ways: using its title "Hypermedia Structures and Systems", and using three codes that exist for it: 2L670, INF725 and INF706. We also searched for the names "Paul De Bra" and "Ad Aerts".

- ALIWEB cannot find any of the requested documents because the Web server containing them isn't registered explicitly with ALIWEB.
- Yahoo is only able to find the course using the code INF706. Since the first page of the course contains the code INF706 twice and INF725 and 2L670 only once, this suggests that Yahoo selects a few words for its index database, based on the frequency of their occurrence. The names "Paul De Bra" and "Ad Aerts" cannot be found. The hypermedia course was submitted to Yahoo for inclusion a few months ago, while the home pages were not.
- WebCrawler is able to find the course using the title and ranks it as first. Using either of the codes the course cannot be found. This suggests that the indexer may not like words containing a combination of letters and numbers. When using logical *or* instead of *and* the course is still found, using the title, but no longer ranked first. WebCrawler finds the name "Paul De Bra", but not the home page, and it does not find "Ad Aerts".
- Lycos cannot find the complete title (with logical *and*). It ignores the stopword "and", but still looks for four matching terms (while only three terms remain). When searching for "Hypermedia Structures Systems" it finds the course, and ranks it first. With logical *or* Lycos still ranks the course first. Lycos cannot find the code 2L670, but indicates it looked for L670. This means it wants words to start with a letter. It also cannot find code INF725, but does find the course using code INF706 because that code occurs twice in the first page of the course. This search demonstrates that Lycos does not index the entire text of the documents, but only keeps a small number of words it considers meaningful and discriminative. Lycos finds the home page of "Paul De Bra" when matching 2 terms (not 3). It's not ranked first however, but a still very acceptable second. The abstract it gives of this home pages describes a version which is about two months old. This demonstrates that Lycos indeed does not update its database very frequently. Lycos also finds the home page of "Ad Aerts", demonstrating its completeness.
- Infoseek does not find the course, but the page of the research group, containing a link to it, using both the code 2L670 and the title. Not surprisingly, a search for 2L670 gives the same result, while INF725 and INF706 are not found. The home page of "Paul De Bra" is found, but that of "Ad Aerts" is not, although the page of the research group is, which contains a pointer to it. These results suggest that Infoseek indeed contains much more information about each document than Lycos, but it does not contain nearly as many documents.

- Alta Vista can find the course using either of the codes, but cannot find it using the complete title. This strongly suggests that not all words from documents are used in the index database (contrary to what Alta Vista claims). Alta Vista can find both home pages, using the people's names. Despite the large number of documents Alta Vista has indexed, a mirror of the hypermedia course, installed at the University of Antwerp, cannot be found.
- The World Wide Web Worm is unable to find either the course or the home pages, although it concentrates on document titles. (The name of the course and of the people occur in the title of the corresponding documents.)
- The TradeWave Galaxy has the option of searching entire documents or only titles. Either way the neither the course nor the home pages can be found.
- RBSE's URL Database cannot find the course or the home pages, but suffers from the typical WAIS problem of returning many answers it thinks are similar to the search terms, but that do not actually contain the given terms.

5. Search Tools without their own Database

There are two approaches to improving the results obtained from the index databases and their search tools:

- Tools like the Savvy Search (Colorado State Univ.), the MetaCrawler [13] (Univ. of Washington) and IBM's Infomarket Search distribute a search request to several known index databases, and combine their answers (indicating the source of each answer).
- When one assumes that authors of documents on a specific topic are often aware of each other, and provide links to each other's documents, the result of a database search may often be improved upon by performing an on-line search, starting from the answer of a major search engine such as WebCrawler or Lycos. The Fish-Search [3],[4],[5] is such an on-line search tool.

Combining the forces of several search engines does not necessarily provide better results. Because each index database offers different facilities, like proximity searching, weighing keywords, specifying boolean combinations of words, or selecting exact word matching, tools that distribute queries over such databases cannot take advantage of the strength of each of these systems. However, having a single access point for all major search engines (and avoiding their commercials along the way) can be a benefit to many users.

On-line searching in a Web of 20 million documents may seem futile at first. During an on-line search each document that is examined needs to be downloaded to the site at which the search engine runs. Even for relatively well accessible servers, a transmission rate of about one small document per second is about all one can hope for. However, when starting a search from a list of

URLs of documents given by one of the well known search tools, chances are that some more relevant documents can be found by examining only a few other documents, say 20 to 100. When starting with the result from a WebCrawler search one may find more documents because WebCrawler only knows about a small fraction of the Web. Furthermore, the depth-first search strategy of the fish-search complements the breadth-first strategy of WebCrawler nicely. When starting with a result from Lycos one may find more documents because Lycos may not have used the right words for indexing some other relevant documents. From Infoseek's answer to the search for the hypermedia course, listing the research group's home page with a link to the course, an on-line search using fish-search will easily find the course. From Lycos' answer which contains the first page of the course one can easily find any desired page from that course. (Pages from the course are difficult to find using Lycos, because of Lycos' intricate selection of words from documents. Pages from the course are impossible to find using Infoseek because its database does not contain them.)

The fish-search has been integrated into the Tübingen version of the Mosaic browser. As such it works like a robot in the user's machine. The fish-search also exists as a separate search tool (a CGI-script that can be installed on a Web server). When used this way the robot actually runs at the server site where it is installed, and not in the user's machine. This is most useful for users on a slow modem connection, who would rather perform the on-line search on a server machine with a fast network connection.

WebCrawler also has an experimental on-line search, which combines searching with additions to the index database. This search however is not available to the public.

6. A Distributed Approach

The Lycos project demonstrates that even with a considerable investment in computer and network hardware it is impossible to download the entire World Wide Web onto a single site in a reasonably short period of time. Given the rate at which the Web changes and grows, and the temporary nature of a lot of information on the Web, the usability of an index database decreases significantly with each day it isn't updated. In view of the saturation of the Internet during weekdays the WebCrawler approach of downloading as much as possible during the weekend seems the least unacceptable compromise.

Distributing the download process over multiple sites, located in different parts of the world may provide a solution to the index generation problem:

- When each participating site indexes Web servers in their vincinity (not necessarily geographically close, but with fast network connections to the participating sites) the number of documents that can be downloaded on a weekend is larger than when a single site tries to download documents from all over the world.
- Unless there is a large (and undesirable) overlap in the sets of servers indexed by each participant the separate download actions of each site do not

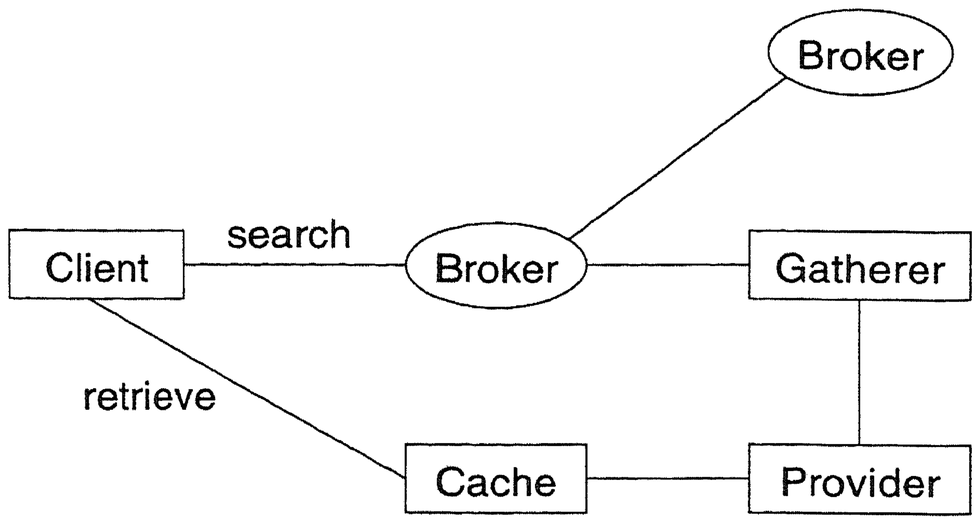

FIGURE 3. Simplified Harvest Architecture.

share network connections with each other, and can thus achieve a much larger total throughput.

Researchers from a number of institutes have initiated the *Harvest* project [12]. Figure 3 shows a simplified view of the architecture of Harvest. Harvest consists of the following types of parts:

- A *provider* is a server, e.g. a Web server.
- A *gatherer* collects information from one or more providers and generates an index database. Normally a gatherer resides on the same machine as the provider, but a large institute may use one gatherer for all of its departmental Web servers together. The Harvest gatherer uses *Essence* to recognize structural elements (extract files from tar archives, find the title and author in HTML and Latex documents, etc.) and *Glimpse* for the index databases, offering regular- and approximate expression matching.
- A *broker* offers a search interface. It uses index databases from gatherers and also forwards queries to other brokers.
- An efficient *object cache* delivers the requested objects (e.g. HTML documents) using the shortest path to the user.

It is hard to predict how many gatherers and brokers the Web will need in order to provide a decent search facility with complete coverage and good performance. A lot depends on the evolution of network bandwidth. But the ratio between the size of index databases built by gatherers to that of the complete texts is also important. A Glimpse index database as used by Harvest requires about 3 to 7 % of the space used by the indexed documents. Thus, exchanging information between gatherers and brokers is much more efficient than building centralized index databases by downloading the complete documents (and one by one).

7. Conclusions

There are many freely available search tools for the Web. No single tool provides a solution for every request from every user. The main names to remember, and their strongest features are:

- WebCrawler is a search engine with a broad but far from complete coverage of the Web. It is updated frequently (weekly) and it indexes the complete contents of documents. Information which occurs in several documents on a Web site is typically found by WebCrawler, while very specific information hidden deep in a Web server's document hierarchy is typically not found.
- Lycos is a search engine with almost complete coverage of the Web. It is continually updated but because it takes a long time to go through the entire Web it contains a lot of outdated information. Its index database only contains a small number of words per document, making it difficult to guess good search keywords to find a document. The possibility to require a search to match (at least) a certain number of the given keywords is very helpful to narrow down a search.
- Infoseek offers a reasonable coverage (about 10 %) and indexes the complete text of documents. Documents can be found using any set of words that occur in them. Infoseek can also search for documents *not* containing certain words. Also, when words need to appear near to each other, one can tell Infoseek to consider or ignore the order in which the words need appear. After finding some documents, Infoseek offers the possibility to search for documents that are "similar" to a given document.
- Alta Vista is a very complete index database, and offers rich boolean operators to search for documents containing some or none of a given set of terms. This tool is especially useful for people who are used to dealing with boolean formulae, but also offers good results when just given a set of keywords.
- Meta search tools such as the MetaCrawler are very useful because they forward search requests to WebCrawler, Lycos and Infoseek in parallel. When one needs specific features of one of these search tools, such as matching at least a certain number of words, a meta tool cannot be used.
- On-line search tools such as the Fish-Search are useful to perform a limited deep search, starting from an answer given by a search engine such as Lycos, Infoseek or Alta Vista. Such a search may lead to information that is either still very recent, or that is many links away from the root of a server document tree.
- Tools that do not index the contents of documents (but only URLs or titles) are not very effective for locating information.

Although the combined set of available search tools for the Web does not come close to the state of the art in information retrieval, it is fairly easy to find most information on the Web by using the appropriate selection of tools and a combination of keywords describing the user's information request.

REFERENCES

1. BÄRTSCHI, M.. (1985). An overview of information retrieval subjects, *IEEE Computer* **18**:5, pp 67–84.
2. MC BRIAN, O.A., (1994). GenVL and WWWW: Tools for Taming the Web, *First WWW Conference*,Geneva.
3. DE BRA, P., POST, R., (1994). Information Retrieval in the World-Wide Web: Making Client-Based Searching Feasible , First WWW Conference, Geneva,*Journal on Computer Networks and ISDN Systems* **27**,pp 183–192, Elsevier Science BV.
4. DE BRA, P., POST, R., (1994). Searching for Arbitrary Information in the World-Wide Web: the Fish-Search for Mosaic, *Second WWW Conference, Chicago.*
5. DE BRA, P., HOUBEN, G.J., KORNATZKY, Y., POST, R., (1994) Information Retrieval in Distributed Hypertexts, *Proc. RIAO-94 Conference*,pp 481–492, New York.
6. DENNING, E., (1989). The Internet Worm, *American Scientist*,March-April 89, pp. 126-128.
7. EICHMANN, D., (1994). The RBSE Spider - Balancing Effective Search Against Web Load, *First WWW Conference*, Geneva, pp 113–120.
8. EICHMANN, D., (1994). Ethical Web Agents, *Second WWW Conference*,Chicago.
9. KOSTER, M., (1994). ALIWEB – Archie-Like Indexing in the WEB, *First WWW Conference*,Geneva.
10. PINKERTON, B., (1994). Finding What People Want: Experiences with the WebCrawler, *Second WWW Conference*,Chicago.
11. SALTON, G. (1989). *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison-Wesley.
12. SCHWARTZ, M.F, BOWMAN, C.M., DANZIG, P.B., HARDY, D.R., MANBER, U., (1994). The Harvest Information Discovery and Access System, *Second WWW Conference*,Chicago.
13. SELBERG, E., ETZIONI, O., (1995). Multi-Engine Search and Comparison Using the MetaCrawler, *Fourth WWW Conference*,Boston, pp 195–208.
14. SHNEIDERMAN, B., KEARSLEY G. (1989). *Hypertext Hands-On!: An Introduction to a New Way of Organizing and Accessing Information*, Addison Wesley.
15. DE VOCHT, J. (1994). *Experiments for the Characterization of Hypertext Structure*,Masters Thesis, Eindhoven Univ. of Technology.