Searching for Arbitrary Information in the WWW: the Fish−Search for Mosaic

dr. P.M.E. De Bra, debra@win.tue.nl
drs. R.D.J. Post, reinpost@win.tue.nl

Abstract

While the World−Wide Web (WWW, or Web for short) offers an incredibly rich base of information, organized as a hypertext, it does not provide a uniform and efficient way to retrieve specific information, based on user−defined search−criteria.

Two types of search tools have been developed for the WWW:

- Gateways, offering (limited) search operations on small or large parts of the WWW, using a pre−compiled database. These databases are often built by an automated Web scanner (a "robot").
- A client−based search tool that does automated navigation, thereby working more or less like a browsing user, but much faster and following an optimized strategy.

This paper highlights the properties, implementation and possible future developments of a client−based search tool called the *fish−search*, and compares it to other approaches. The fish−search, implemented on top of Mosaic for X, offers an open−ended selection of search criteria. It allows the search to start from the "current" document or from the documents in the user's hotlist.

The future of client−based searching will depend heavily on assistance from WWW servers. The fish−search algorithm is currently being used to search by navigating through individual WWW documents. Substantial speedup and reduction of network resource consumption will become possible when the same search algorithm is converted to navigate through a Web of servers instead of a Web of documents. In order to implement such a search algorithm, WWW servers need to offer a service similar to that provided by GlimpseHTTP.

1. Introduction

The World−Wide Web (WWW) is a loosely organized distributed information base, shared by a rapidly growing number of Internet sites. Becoming a node in this (distributed) database is easy because reliable and ready to use servers are freely available. After installing the server software all one has to do is to provide the information in the *HyperText Markup Language* (HTML). Gaining access to the database is even easier, as a large number of user−interfaces to the WWW are also freely available in both source and binary format for many different platforms.

To some extent, the loose organization of the WWW is both the key to its success and the biggest problem when trying to join it: there is no catalog of what information is available on the WWW and where, and there is no organized way to set up such a catalog if one would want to. There is no way to find information in the WWW if one does not already know where it is. Also, there is no way to let other people know through the WWW that you have information available on your server. You have to distribute pointers to your information by other means, like by using electronic mail.

Documents in the WWW are uniquely identified by means of a *Universal Resource Locator* (URL), which includes the name of the server the documents resides on. Documents may contain pointers (links) to other documents, possibly on other servers. The *Web* structure of documents and pointers makes the WWW a *Hypertext*. Therefore we shall use the terms *node* and *link* in the sequel, as is done in most hypertext literature [SK89].

The only access to information in the WWW is defined by means of the *HyperText Transfer Protocol* (HTTP), which is used by the user's (client) interface to retrieve nodes from a WWW server. There is no protocol for asking a server what nodes it has, and no protocol to find out which servers exist. The access to the WWW is therefore limited to *browsing* (sometimes called *navigation*): you retrieve one node, look at the links it contains to other nodes, then retrieve one of these other nodes, etc.

Browsing is the common interaction paradigm for hypertext, when a user is gathering information. It is very useful for reading and comprehending the contents of a hypertext, but not suitable for locating a specific piece of information. Most hypertext systems offer a *global search* facility that lets you find a specific node based on information you know about its contents, regardless the navigation that would be necessary (the links to follow) to get to that node. In a distributed and loosely organized hypertext like the WWW there is no global search facility. However, two kinds of attempts have been made to provide such a facility:

- By means of browsing, some automated Web scanners (also called *robots*) have been reading all information in the WWW, and have constructed a database which can be searched. These databases have been added to the WWW (by providing an interface that makes them look like a WWW node), and their existence has been made public by means of electronic mail and news. (Remember, there is no way in the WWW to make anything known to all WWW users.) Because the WWW is very large, not necessarily completely connected, and changing all the time, it is impossible to provide databases that truly function as a *catalog* of the WWW. These robots run for weeks, gathering information, forgetting most of it (because there is too much to index everything in the WWW), and failing to reach the newest and still not very well known sites. The information these databases contain is very useful, but unreliable and incomplete. Most organizations offering access to such index databases fail to warn the user about these shortcomings of their systems.
- Our approach is to offer a client−based search tool that does automated navigation, thereby working more or less like a browsing user, but much faster and following an optimized strategy. Like with the robots, the search is always incomplete. Since the search is performed on−line, there is little time to retrieve nodes and check whether they are relevant. Only a tiny fraction of the WWW can be searched at once. But contrary to the index databases, the search uses the entire contents of the nodes (not just a title or header) and your client machine may search for any kind of information you provide a filter for, not just keywords or phrases.

In Section 2 we describe how this client−based search works. The search is currently embedded in Mosaic (but only in the 2.4.2 version of the Univ. of Tübingen). Section 3 briefly presents a Mosaic−independent interface to the fish−search, by means of a *cgi−script*, which must be installed on a local (server) machine.. Finally, in Section 4 we propose a new approach to client−based searching, which assumes there will be some cooperation of the servers in the future. Reliably finding information in the WWW will not be possible in the future if the access to WWW−servers remains limited to the retrieval of (single) nodes. Tools like GlimpseHTTP will improve the possibilities for finding information in the WWW.

2. The Fish−Search Algorithm and Tool

The *fish−search* is completely client−based:

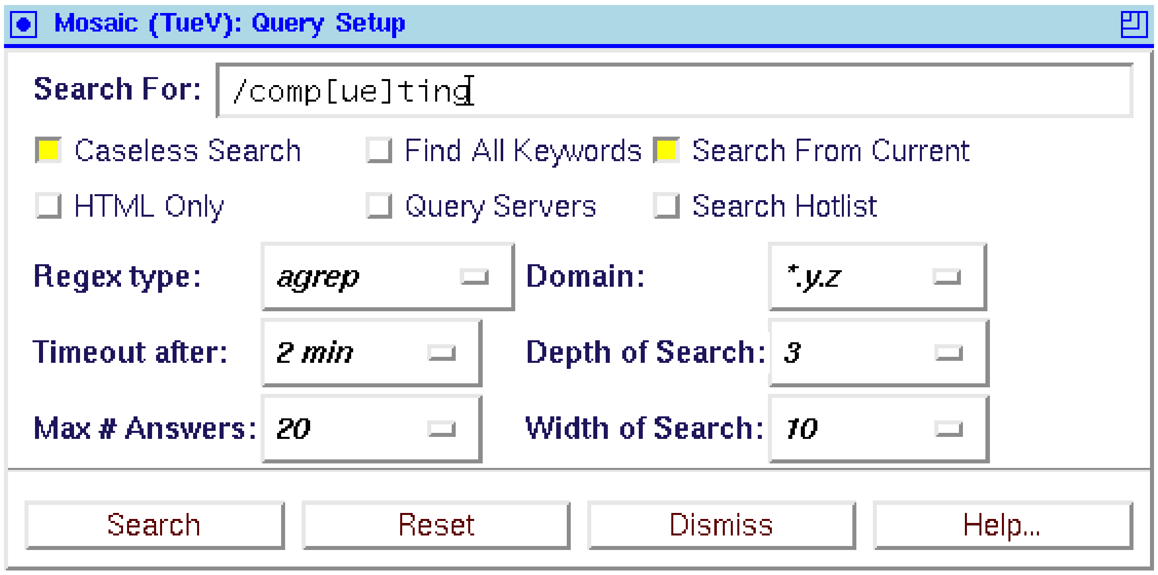
- Every node is retrieved just like a WWW−browser retrieves it from a WWW−server.
- The node is scanned for relevant information. This search process runs on the client (your) machine, so any kind of program on your machine can be used to detect relevant information.
- The node is also scanned for links to other nodes, which are retrieved and scanned later.

In the sequel we shall consistently use the term *relevant node* to indicate nodes that contain the words or expressions the search tries to find.

A number of factors determine how effective and efficient the search−tool can be:

- The most difficult step is the first one: finding a starting node, from which relevant nodes for the search can be found by following only a few links. The robot−based index databases may provide good starting points.
- The algorithm must try to find nodes that are well spread over the WWW. The experiments of [DV94] show that navigation algorithms that resemble depth−first search perform well in this respect.
- The algorithm must try to avoid downloading irrelevant nodes. Since the WWW contains information on many subjects, but many authors of information on a subject know about information others wrote about the same information, it is reasonable to expect relevant nodes to be pointing to other relevant nodes, possibly on another WWW−server. By favoring links to neighbors of relevant nodes, and also favoring links to nodes on different sites, the chance of finding more relevant nodes as well as penetrating into different parts of the WWW increases.
- Retrieving nodes should be fast. Since most users of a WWW−browser (especially users at the same site) are likely to start searching from the same node (or a small set of nodes), they all search through the same nodes many times before finding a cluster of relevant nodes. Using a *cache* as described in [BP94] reduces the time used (or wasted) on these popular nodes. However, it generates some overhead when retrieving a node for the first time.

A detailed description of the fish−search algorithm and its heuristics can be found in [BHKP94]. We concentrate on the search−tool that is part of the Tübingen Mosaic 2.4.2, a derivative of NCSA’s Mosaic for X 2.4. The figure below shows the widget used to specify a search request.



Mosaic (TueV): Query Setup

Search For: /comp[ue]ting

[✓ Caseless Search] Find All Keywords [✓ Search From Current]

HTML Only Query Servers Search Hotlist

Regex type: agrep

Domain: *.y.z

Timeout after: 2 min

Depth of Search: 3

Max # Answers: 20

Width of Search: 10

Figure 1: Widget for specifying a search request.

Apart from the search string, the widget offers two different sets of selectable options and parameters:

- **content search options:**

  search algorithm:

  The current version of the tool offers three different routines for scanning a node: you may search for one or more words, for a regular expression using the GNU regex library, or for approximate matches of a regular expression using the agrep library [WM92]. When the search string (typed in the **Search For** field) starts with a slash (/), a regular expression search is performed, following the syntax indicated by the **Regex type** menu selection. The "agrep" choice activates the approximate search, while all other choices are syntaxes offered by the GNU regex library. When the search string does not start with a slash, a "literal" search is performed for the words you type in the **Search For** field.

  In addition to these three content search routines, the tool also lets you run an external program (filter) in order to determine the relevance of each node. The filter must read *standard input* and must write a number on its *standard output*. An external filter is used when the search string starts with an exclamation mark (!). An example of the use of an external filter would be:

  ```
  !agrep -1 "comp[ue]ting" | wc -l
  ```

  Caseless Search:

  When this toggle is on the search ignores the difference between lowercase and uppercase letters.

  Find All Keywords:

  This toggle only applies to the literal search. When this toggle is on and more than one word is given as the search string, a node is only considered relevant if it contains all given words. When the toggle is off, nodes containing one or more of the words are also considered relevant, though they get a lower relevance rating than nodes containing all the words. The words in the search string must be separated with spaces.

Query Servers:

The WWW contains nodes that are actually interfaces to programs that run on the WWW−server the node resides on. Such *index* nodes are typically used by index databases. Normally, the node would be displayed with an embedded fill−in field. When you type something in that field, the remote program performs some action with what you typed. The typical action is to look your string up in a database. The remote program returns a new node, which may contain links to more relevant nodes. When the "Query Servers" toggle is turned on, the search−tool will pass the search−string to the remote server and hopefully receive a node which contains links to more relevant nodes. There are also WWW−nodes with more complicated fill−in forms. The search tool always ignores such forms, for which the HTTP protocol defines a special (POST) request. Only the simple forms with one text field are used. The WWW also contains *clickable maps*. Clicking inside such a map activates a program on the WWW−server, that usually returns a new node which depends on the coordinates of the click. Even in small images there are far too many possible coordinates to try each of them. The search tool ignores clickable maps, thus possibly missing some links to relevant nodes.

- **navigation options:**

HTML Only:

This toggle (when turned on) restricts the search to nodes retrieved through the WWW's HTTP protocol, and whose name ends with the .HTML suffix (in upper− or lowercase). Although the WWW may contain textual information not satisfying these conditions (like HTML files with no suffix), this is the fastest way to avoid retrieving non−textual information, which cannot be used by the search. By using "HEAD" requests it would be possible to detect whether a node contains text or something else (images, sounds, etc.). However, a "HEAD" request to a remote site still requires quite a bit of time. Since the fish−search does not aim at completeness we accept the fact that the algorithm misses some potentially relevant nodes.

Search From Current:

When this toggle is on, the search starts from the node that is displayed in the document view. This is the default choice. Selecting a relevant or a well connected starting node significantly increases the chance of finding answers to the search.

Search Hotlist:

This toggle was added by popular demand: many users have a large hotlist, and the titles of nodes are often not very meaningful (which is why index databases are often unreliable). You may know that there is a node in your hotlist that deals with a certain topic, but not remember which node it is. When this toggle is on, the documents in the hotlist are searched (and documents connected to them up to a selectable distance).

Domain:

Retrieving nodes from other sites, possibly on another continent, can be slow. By limiting the search to nodes from sites which share part of their domainname with the site of the node the search is started from, a short search can be limited to sites in the neighborhood of a server to which you have a fast connection. The author's "home page", the usual starting point for a search, is located on a machine with domainname `win.tue.nl`. When the domain menu item `*.y.z` is selected (as in the figure), only nodes on servers with a domain name ending in `tue.nl` are searched. A more general way to specify the domains to include or exclude from the search is being considered for future versions of the search tool.

Depth of Search:

The WWW may contain a number of relevant nodes that are not directly linked to each other. It is possible that in order to find a relevant node you must follow links through a number of irrelevant nodes. The maximum number of (forward) links through irrelevant nodes that will be followed before giving up on that direction is selected through this *depth* menu.

Width of Search:

The fish−search selects a number of outgoing links from every retrieved node. In general, not all links are used, to avoid spending too much time in the immediate neighborhood of a single node with many outgoing links. Since we expect relevant nodes to be clustered, we consider more outgoing links of relevant nodes than of irrelevant nodes. The number of outgoing links considered is set by the *width* menu.

A complete search of the WWW is normally impossible, since the WWW is not necessarily completely connected, since the part of the WWW that can be reached depends on how well the starting node is connected, and since some parts of the WWW are "hidden" behind nodes with embedded forms and clickable maps. But most importantly, a complete search would take weeks. The fish−search is intended to run for a limited time, which can be selected, and to find a limited number of relevant nodes, which can also be selected.

While the fish−search can be used as a general−purpose search tool, special care has to be taken to prevent the search from being too slow and ineffective. If all nodes have to be retrieved from remote locations, like different continents, the search will be too slow. Using a cache may significantly improve the search speed. Limiting the search to a local domain may improve the speed even more. The fish−search has been used very successfully as a global search tool for local parts of the WWW. In order to find information in the whole WWW, starting from an answer produced by an index database like the JumpStation or the World Wide Web Worm may increase the chance of finding relevant nodes.

3. The Fish−Search without Mosaic

The implementation of the fish−search embedded in the Tübingen Mosaic for X 2.4.2 suffers from three sources of performance problems:

- The nodes that are scanned are all downloaded into the client's (your) machine. In quite a few cases the network bottleneck is the link between your workstation and the nearest Internet gateway. Using the fish−search at home, on a SLIP or PPP connection through a modem, is close to impossible. Even the speed of an ISDN line is poor compared to the Internet connections between large institutions and companies.
- The computational load of running Mosaic, downloading and scanning the nodes for relevant information is all concentrated in the client. When you are using a low−cost workstation, connected to a high−speed compute−server, it is a shame to leave the server mostly idle while fully loading your slow workstation.
- When you want to search the neighborhood of a remote site (which you can do by restricting the domain as described in the previous section) all nodes have to be transferred through the

Internet to your machine. If you could log on to a workstation at the remote site you could be able to perform the same search an order of magnitude faster. The search−tool should run on a machine with a fast connection to the neighborhood you want to search, not on your own machine.

However, the biggest problem with the fish−search is that users of other WWW browsers (even NCSA Mosaic for X, MS−Windows or Mac) are left out. Luckily, the WWW provides a way to solve this problem: the use of *forms* and *cgi−scripts*.

The "Query Setup" window shown in Figure 1 only contains widgets one can include in WWW forms: a (one−line) text field, some toggles (checkboxes), menus and buttons. Figure 2 shows a form that lets you specify a search request just like the Query Setup of the Mosaic fish−search. When pressing the **Search** button, a POST request is submitted to a WWW−server. That server will execute a program, called a *cgi−script* that performs the actual search. It returns an HTML document containing the answers to the search.

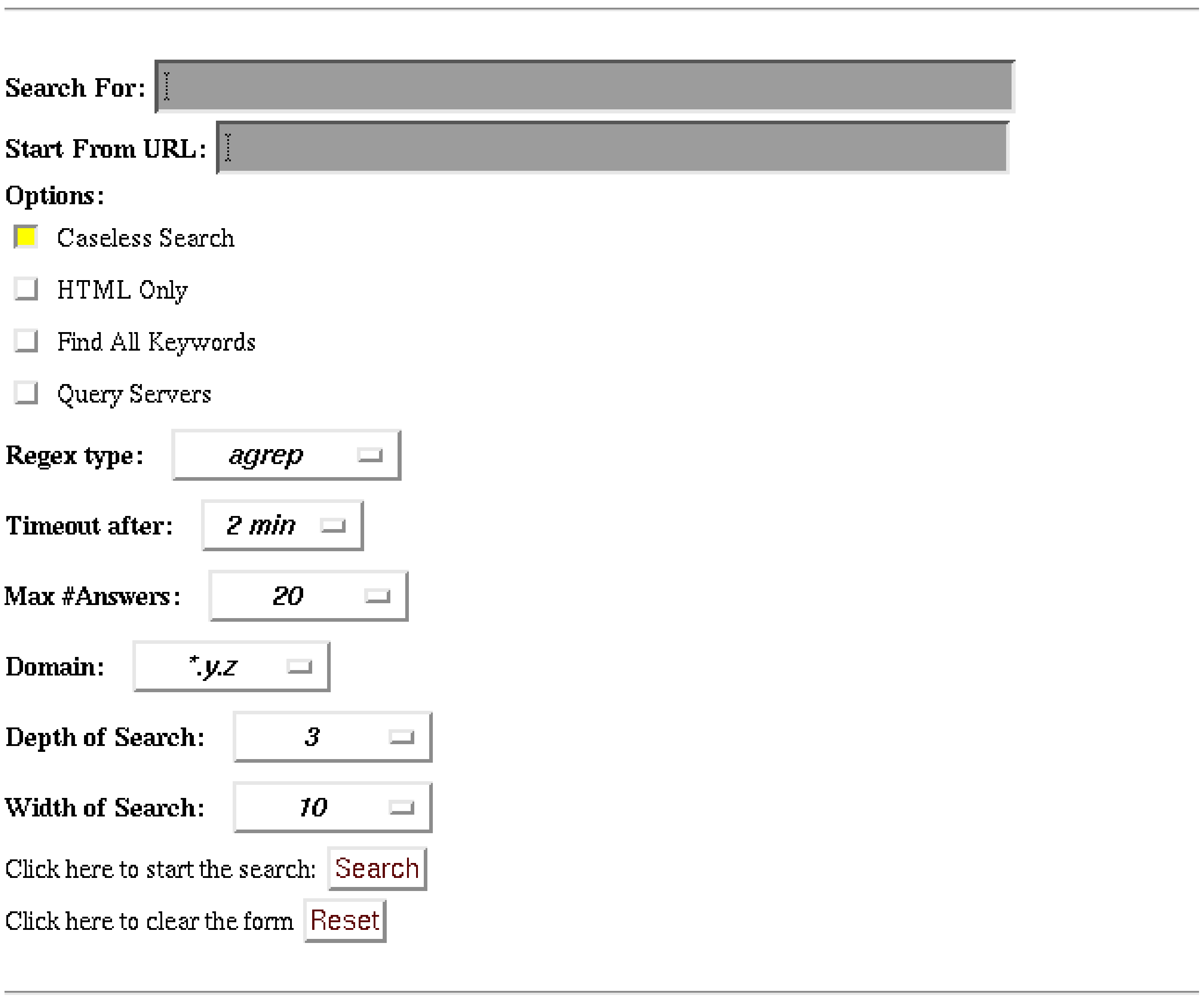
The fish−search form is not identical to the Query Setup window for Mosaic. Since your "current" node is the search form, the "current" node cannot also be the answer from an index database or any other node you would like. Therefore, the form contains a field for specifying the URL of the document to start from. The server also does not know what your hotlist contains. Searching your hotlist through the forms−based search is not possible.

Using the fish−search by means of this form requires you to install the actual search program on a WWW server. Also, you must write an HTML document containing the form. The default selections for the toggles and menus are preset in the form. In order to modify them you must change the form, not a configuration file for your WWW browser.

Compared to the Mosaic version of the fish−search the forms−based version has two more drawbacks: first, the result of the search is returned as an HTML document, after the search is completed. The Mosaic version shows the answers in a separate window as they are being found while searching. Not getting any feedback for a long time can be somewhat disturbing. The second drawback is that since the server executes the search program, you cannot provide your own filter program for determining the relevance of the nodes. Luckily, the combination of literal search, regular expressions and approximate search provides enough possibilities for almost all applications.

In order to be useful for searching the neighborhood of a remote site the fish−search program needs to be installed on a WWW server in *that* neighborhood. Only then are the Internet transfers limited to that neighborhood, and thus faster. So which neighborhoods can be efficiently search depends on where the fish−search programs are installed.

The fish−search cgi−script, when completed, will be made available from `ftp.win.tue.nl` in the directory `pub/infosystems/www`, a directory that also contains a copy of the Tübingen Mosaic for X and the Lagoon cache [BP94].



Search For:

Start From URL:

Options:

[✓ Caseless Search]

HTML Only

Find All Keywords

Query Servers

Regex type: agrep

Timeout after: 2 min

Max #Answers: 20

Domain: *.y.z

Depth of Search: 3

Width of Search: 10

Click here to start the search: Search

Click here to clear the form Reset

Figure 2: Fish−Search Form. (In the on−line version of this paper this is an actual form, so all options can be selected from the menus.)

4. The Future of Searching in the WWW

The World−Wide Web is becoming very large, and still growing at an enormous rate. Some people developing robots that scan the WWW for information to be used in index databases have found around 8000 WWW servers already (although some may be duplicates using different names). At this rate it is simply impossible to build a true catalog of the WWW. It is also becoming more and more difficult for the fish−search to find relevant documents for most search requests, because finding them is beginning to resemble searching for a needle in a haystack.

The problem of finding information, even if one has ample time, like the robots have, is further complicated by the fact that the WWW is not necessarily completely reachable even from a large number of starting nodes (there will always be servers nobody is pointing to yet), and there are parts of the WWW that are hidden behind forms and clickable maps. The number of possible inputs for forms and maps is much too large to try all possibilities in order to find the information behind them.

The *core* of the problem however is the following:
organized (query) access to a server's information should be offered by that server, not by remote databases or search tools.
This idea motivated the designers of GlimpseHTTP to develop a forms−based search interface to the complete information that is stored on a single server (actually, in a directory tree). GlimpseHTTP offers approximate regular expression search (from which the fish−search borrowed the agrep library) on the complete contents of all nodes on a server, also the nodes that are hidden behind forms and maps, and also the nodes to which there is no link in the WWW, not even on that same server.

A database like GlimpseHTTP can be kept up to date by means of a program that is run daily (nightly). A complete and up to date list of all the nodes on a server containing a given string, expression, or approximation, can be obtained within seconds. With the fish−search this would take minutes to hours depending on the connection with that server, and with all existing index−databases put together, this list would still not be obtained because all these databases are always outdated and incomplete.

In order to search the whole WWW (or a large part thereof) for a string or expression, the navigation strategy of the fish−search could be used to jump from server to server, instead of from node to node. To do that, a server should not only provide a list of nodes containing relevant information, but also a list of other servers it knows about. Each of these servers can then be asked the same question: "Give me a list of all nodes containing the search string, and give me a list of all WWW servers you know about."

Providing this kind of information on the whole WWW is not as difficult as it seems. There are only a few popular WWW server packages, most notably the ones offered by CERN and NCSA. If these packages would include such an "extended GlimpseHTTP−like" tool and a standard form to access it, the implementation of an efficient and very effective fish−search would be relatively easy. Index−databases could be replaced by caches containing recent answers to searches for frequently used strings or expressions. They could also provide the names (and addresses) of servers they know about.

For many applications the quick but incomplete and outdated answers one gets from the current generation of index databases will still be sufficient for some time. For up−to−date information on occurrences of strings or expressions in a small part of the WWW the current fish−search will still be sufficient. But for a high−quality information service, the client−based node by node search is too slow, and the second−hand information on the index−databases too unreliable. Furthermore, more and more people (administrators) object to the network resource consumption by the robots that are used by the current generation of search tools. By installing software on each server (or small clusters of servers) that provides up−to−date and complete information on its own part of the WWW, the Web should become searchable again.

References

This section only contains the references to paper documents. References to descriptions of information retrieval tools for the WWW for instance are embedded as links in this paper, and can only be used in the on−line version.

[BHKP94]
De Bra, P., Houben, G.J., Kornatzky Y., Post R., *Information Retrieval in Distributed Hypertexts*, RIAO−94 Conference, New York, October 1994.

[BP94]
De Bra, P., Post, R., *Information Retrieval in the World−Wide Web: Makinc Client−based searching feasible*, First WWW Conference, Geneva, April 1994.

[DV94]
De Vocht, J., *Experiments for the Characterization of Hypertext Structures, Masters Thesis*, Eindhoven Univ. of Technology, April 1994.

[SK89]
Shneiderman, B., Kearsley G., *Hypertext Hands−On!: An Introduction to a New Way of Organizing and Accessing Information*, Addison Wesley, 1989.

[WM92]
Wu, S., Manber, U., *Agrep − A Fast Approximate Pattern−Matching Tool*, Usenix Conference, 1992.

Author biographies and address

dr. P.M.E. De Bra

Paul De Bra is associate professor at the Eindhoven University of Technology, the Netherlands, where he teaches and performs research in databases and hypermedia technology. He is also a part−time professor at the University of Antwerp, Belgium.

Paul De Bra received his ph.d. in computer science from the University of Antwerp, in 1987. His research was on Horizontal Decompositions of Relational Databases. As a post−doctoral researcher he spent one and a half years at AT& Bell Laboratories in Murray Hill, NJ, working on WYSIWYG interfaces for document processing. In december 1989 he joined the Eindhoven University of Technology.

His research interests are models, information retrieval and structure analysis and querying for hypermedia databases.

drs. R. Post

Reinier Post received his masters degree from the University of Nijmegen in 1989. In 1990 he worked on hypertext at the Dutch Open University. He is currently working on a ph.d. in computing science at the Eindhoven University of Technology. Dr. De Bra is his advisor. His research topic is information retrieval and querying in hypertext.

E−mail Addresses

dr. P.M.E. De Bra: debra@win.tue.nl
drs. R. Post: reinpost@win.tue.nl