The Search Broker

Udi Manber and Peter A. Bigot
The University of Arizona

The Search Broker

`http://sb.CS.Arizona.EDU/sb/`

Udi Manber Peter A. Bigot

Department of Computer Science
The University of Arizona
Tucson, AZ 85721-0077
{udi,pab}@CS.Arizona.EDU

Abstract

The current search facilities on the web are amazingly powerful, but they are still lacking. Taking the whole universe as one flat data space and searching it with keywords has inherent limitations of scale. The challenge is to provide users with ways to focus their search better without making it too difficult or too inefficient. We introduce a method of conducting search on the web that is based on a two-level search idea. It strikes a balance between flat global search and specialized databases, and gives users convenient access to vast amounts of information.

1 Introduction

Large scale web search started in two directions, both of which proved very successful. The first is "spider"-based collection of as much of the web as possible, combined with powerful search engines that provide access to all this data. Lycos, WebCrawler, Altavista, and several others are examples of this model. The second is based on manual collection and classification with browsing and search facilities. Yahoo is, of course, the most successful provider of this model. This is not the place for a detailed analysis of these two approaches; let us just mention their most obvious weaknesses: The spider-based approach's main problem (and often main strength as well) is that it is indiscriminatory. It tries to cover everything. It is not uncommon to obtain thousands of hits, most of which are "garbage," and then have to sift through many of them. It's also impossible to expect to have all the world's information in one flat database. All spider-based engines take only a small fraction of most large sites. (E.g., you cannot expect them to collect all 16GB of Medline, and as a result they will not give you all pertinent medical information.) Yahoo's approach guarantees more quality, but browsing is often very time consuming, and of course, coverage is limited.

These weaknesses are most glaring when one looks for answers to specific reference questions, such as:

- how much fat is there in a pepperoni pizza?
- how do you say *search* in Latin?
- how do you delete a directory in UNIX?
- give me a list of hotels in Phoenix.

These are hard questions to answer based on keywords alone and flat search. Using the spider-based services, one will have to think of the right keywords, and if they are too common, a lot of hits will have to be followed with no guarantee of quality of information. Yahoo will probably be more suitable for these questions, not by trying to answer them directly, but by trying to find the right categories (e.g., dictionaries, although Yahoo doesn't know about a Latin one), then following them to hopefully the right places that maintain relevant information. But in any case, it could be quite time consuming (e.g., try to find the list of Phoenix hotels).

The approach we present here is based on the idea of a two-level search. Instead of always searching the same all-encompassing database, imagine having specific databases for specific topics. The search will consist of two phases: In the first phase, the search is after the right database, and in the second phase the relevant information is searched for within this database. This is not a new approach, of course. It is similar, in a sense, to using the library subject card catalog to find the right shelf, or using Yahoo to find the right category. The novelty of our tool is that we combine the two phases into one regular search in a way that makes the process very easy and very powerful for users. The resulting tool provides search features that are not available in any one place on the web.

Consider again the three questions listed above. The first one has to do with nutrition, the second with Latin, the third with UNIX, and the fourth with hotels. These are the most important characteristics of these questions. The person who asks the question can usually pinpoint its subject; not precisely, maybe, but usually close. For example, the questions above could be answered more precisely if they were in the following forms:

- Subject: nutrition; Query: pizza
- Subject: latin; Query: search
- Subject: unix; Query: delete directory
- Subject: hotels; Query: Phoenix

Knowing the right subject may be tricky. Some users may input calories instead of nutrition, or accommodations instead of hotels. We only ask that some information about the subject be included in the query. We also replace the rather complex syntax above with a very simple query, as we will show shortly.

Our approach works as follows. We collected over 400 different search providers that we judged to have reasonable general appeal. (This is an on-going process, of course; we expect a fully-operational system to have thousands of servers.) Each such search server covers a certain subject or category (such as nutrition, latin, unix, or hotels). Each such category is identified by one or two words, and it is also associated with a list of *aliases* that people may think about when searching for that subject. So nutrition can be associated with calories, and hotels with motels, lodging, and accommodations. The collection of search engines and the assignment of the words and aliases that identify them are done manually by a librarian. It can also be customized by end-users (we'll discuss that aspect later on). This is a part that we intentionally do not wish to automate. The role of editors, reviewers, interpreters, and librarians has been rather limited in the web, mainly because of its scale. Finding paradigms that will allow significant librarian input while supporting the scale of the web is increasingly important. The two-level approach is promising because the number of subjects does not grow too fast (as compared to the number of web pages or even the number of web sites).

A user query is made of two parts corresponding to the two phases of the search. In the current implementation both parts are combined into one simple box. To answer the questions above you type:

- nutrition pizza
- latin search
- unix delete directory
- hotel Phoenix

and you get direct results from the appropriate search services. Given a query like hotel phoenix, the Search Broker performs the following steps:

1. It searches its own database for subjects and aliases and finds the search engine corresponding to hotel. With aliases, all subjects allow both plural and singular names (hotels works as well as hotel). In the current implementation, the subject must be the first word in the query, mainly because we want users to identify the subject and think about it. We could easily select any word in the query that matches a subject and try it (or all of them).
2. After identifying the particular search engine, the rest of the query is reformatted to the form expected by that search engine. This step can sometimes be quite complicated, and we discuss it in detail later.
3. An HTTP (Hypertext Transfer Protocol [Fielding]) request is sent to the search engine with the appropriate fields that match the query.
4. The results of the query are sent back to the user.

This simple-minded approach turns out to be extremely powerful. The proliferation of search software, often for free, made it easy to provide search capabilities on many web sites. (We are proud to be partially responsible for that with our glimpse, glimpseHTTP, WebGlimpse, and Harvest systems.) Within the last year thousands of search servers have been added. Most of them deal with very limited specific information (e.g., they search the content of one site), but many provide professional content in areas of general interest. The trend to connect existing databases to the web will continue. There are already so many high quality search facilities that people cannot keep track of them through bookmarks and favorite lists.

The list of currently available subjects is included in the home page of the Search Broker, and there are also facilities to search the Search Broker's own database. Let's see some examples of queries to demonstrate the power of the approach:

stocks ibm gives the current value of IBM's stock plus links to corporate information and news.

patent object oriented gives abstracts of all patents (from 1971 to present) with these keywords.

howto buy a car gives practical advice about buying used and new cars.

fly sfo jfk gives all scheduled flights between San Francisco and New York JFK.

english-polish wonderful tells you that *cudowny*, *zadziwiajacy*, and *godny podziwu* match the adjective "wonderful".

nba-salaries michael gives the salary of Michael Curry (and all other Michaels playing in the NBA).

car-price 1992 Chevrolet, Camero gives the blue book value for this model (the comma (,) is used as a delimiter).

email bill gates gives email addresses for that name (yes, it includes the one you are thinking of).

travel fiji gives a lot of useful information about travel in Fiji.

expert computer algorithm gives a list of experts who put "computer algorithms" in their areas of specialty.

convert 8 liters to pints tells you that there are 16.9 pints in 8 liters.

The Search Broker approach is not a magic bullet, and we do not expect it to replace any of the existing search mechanisms. But we believe that it complements them very well. If one is not sure what one is looking for, browsing works best, and Yahoo presents the right approach. If one is looking for unusual words or names or wants everything known about something, then the spider-based engines cannot be beat. But queries often fall somewhere in between, and the Search Broker can help save time and focus the results. In a sense, it presents a very large live encyclopedia.

The first version of the Search Broker has been operational since October 1996. It was released on the web in July 1997. It can be found at `http://sb.cs.arizona.edu/sb/`.

The rest of the paper is organized as follows. The next section presents the user interface, always an important part of any tool. Then we discuss technical details of how we create and maintain the database of search engines, handle queries, and return results to the user. After that we present a different and maybe even more important use of the Search Broker—as a customizable desktop tool. Related work and conclusions follow.

2 The User Interface

We have experimented with several user interfaces. Our primary interface follows the minimalist approach. It is just one generic search box with a Submit and Reset buttons as shown in Figure 1.

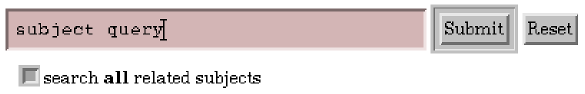
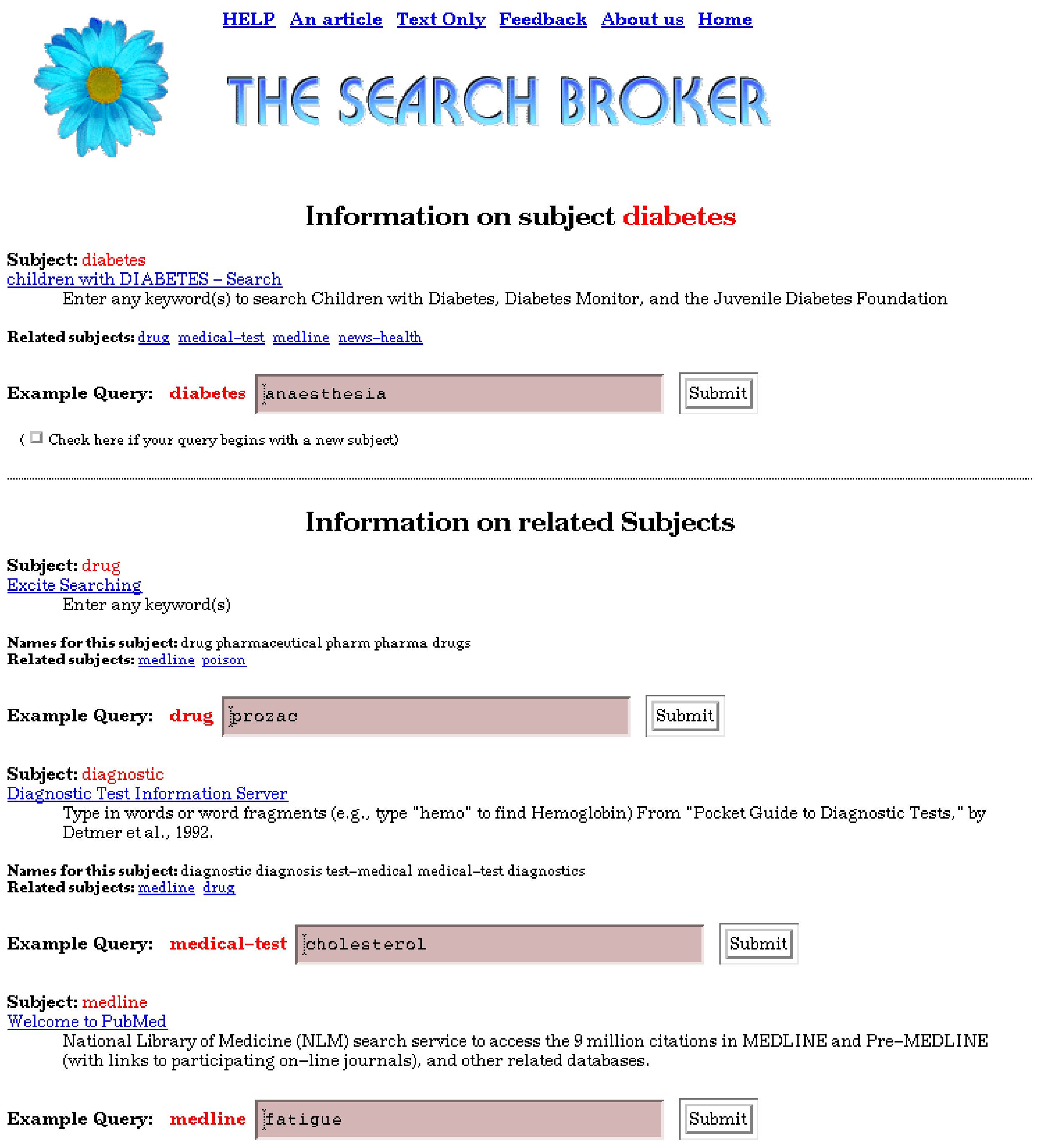


Figure 1: The minimalist user query box

The convention is that the first word is the subject and the rest is the actual query. If the subject requires more than just one word, the words are joined with hyphens; for example, book-kids, tv-guide, or programming-languages. Sometimes the second part is not needed; for example, tv is also an alias for tv-guide, but book is a different subject covering all books rather than just books for kids.

The main problem is how to let users know which subjects exist and what they cover. This is achieved in two ways. First, the list of subjects (divided into several "super" subjects, such as computers, entertainment, and science) is provided below the search form on the Search Broker home page. Second, we provide an extensive help system to search for subjects. For each subject we maintain a list of *related subjects*, again decided manually. If the query contains only one word, then this word is searched for in the list of subjects and information about it and all its related subjects is given. For example, typing *diabetes* will bring the content shown in Figure 2.

If there is no subject matching the first word, the user is presented with the options of forwarding the query to another search engine (using a Search Broker subject like lycos), or revising the query and resubmitting it. Future enhancements may include proposing alternative subjects based on words close to what the user entered (spelling corrections), or by using a more extensive topical-relation database. It is also possible to employ natural language processing techniques to try to automatically infer the subject from the query. We have not tried that yet. Another possibility is the use of pull-



HELP An article Text Only Feedback About us Home

THE SEARCH BROKER

Information on subject diabetes

Subject: diabetes
children with DIABETES – Search
Enter any keyword(s) to search Children with Diabetes, Diabetes Monitor, and the Juvenile Diabetes Foundation

Related subjects: drug medical–test medline news–health

Example Query: **diabetes** anaesthesia Submit

(☐ Check here if your query begins with a new subject)

Information on related Subjects

Subject: drug
Excite Searching
Enter any keyword(s)

Names for this subject: drug pharmaceutical pharm pharma drugs
Related subjects: medline poison

Example Query: **drug** prozac Submit

Subject: diagnostic
Diagnostic Test Information Server
Type in words or word fragments (e.g., type "hemo" to find Hemoglobin) From "Pocket Guide to Diagnostic Tests," by Detmer et al., 1992.

Names for this subject: diagnostic diagnosis test–medical medical–test diagnostics
Related subjects: medline drug

Example Query: **medical–test** cholesterol Submit

Subject: medline
Welcome to PubMed
National Library of Medicine (NLM) search service to access the 9 million citations in MEDLINE and Pre–MEDLINE (with links to participating on–line journals), and other related databases.

Example Query: **medline** fatigue Submit

Figure 2: The results of diabetes

down menus to show the existing subjects so that users do not have to guess them. 400 subjects in one pull-down menu is out of the question, but they could be divided into several categories. From initial experiments, this option does not look attractive to us.

Another feature is an option to search all related subjects with one search. For example, if this option is selected, then the query dictionary bravo (or word bravo) will also search the thesaurus, jargon, phrase, and quotation subjects at the same time.

We place a Search Broker form at the start of the results pages returned from the remote server, so users can make additional queries without having to move to a different page. The initial user interface required the user to have the subject word at the beginning of all queries. By watching how people used the Search Broker, it became clear that users automatically invoked another pattern, where they used the subject in an initial query, but left it off in subsequent ones, assuming searches would continue with the same first-level restriction. To support this usage, we modified the query form on result pages so that the subject would remain the same, and users enter only the second level of the query, unless the user specifically indicated that she wanted a new two-level search.

3 The Search Broker's Database Facilities

The database for the Search Broker contains all the information required to locate an engine that can service a user's query, to rearrange and submit the query in the format expected by the engine, and to provide brief descriptions and examples for each search. This information includes:

- The method and action of the query form (e.g., `GET` and `http://search.yahoo.com/bin/search`);
- A list of form inputs, including their types (text, hidden, checkbox), initial states, and valid states where appropriate (radio and select types);
- A subject name and all its aliases, and a list of related subjects;
- Instructions on how to assign field values from the user's input (query templates; cf. section 4);
- Examples, documentation, and a reference the user can follow back to the original search engine for more refined searching or additional information about the site.

The current database occupies only about 300K with more than 400 subjects.

The accuracy of much of this information is crucial for correct processing of a user's query. For a tool like the Search Broker to be successful, its database must be constantly verified and updated, adding new search engines and removing or modifying entries for ones that have disappeared or changed. We developed a tool which automates much of the tedious portions of database creation and maintenance. Given a URL of a search page, our tool automatically retrieves the HTML search form, translates the action and input fields into a concise format from which the important parts of the original form can be reconstructed, and outputs a database entry template. The librarian then assigns a subject name and aliases, writes a description of the search engine's capabilities, selects which of the form fields to use and sets default values for the rest, and provides query translation patterns. The entry is then ready to add to the database.

In our experience, the "administrative" parts of a subject can typically be added to the database in 20-30 seconds, and therefore do not present a significant burden. The main job of the librarian is to find the right search facilities, give them the most appropriate names and aliases, test them for quality and generality, and write good short descriptions and examples for them. This is in line with the traditional responsibilities and expertise of librarians, which in our opinion the web sorely misses. The Search Broker's design allows librarians to make significant contributions in organizing the web's search facilities.

The same program can also be used for routine maintenance, by querying the entire database on its own. It will go out and verify that each server listed is still there, and that the search form it uses has not changed. If something has changed—the form has moved, or has new fields—the differences are noted for the librarian to review before changing the database.

We are currently extending this program to allow for personal customized use so that people can easily be their own librarians and maintain their own Search Brokers. More on than in the Customization section.

4 Translating Queries

The problem of reformatting queries between different databases is an old problem in the database area. Fortunately, the web makes it simpler, because queries typically use few fields and they typically allow search of the whole database by (non attributed) keywords. HTML forms are often very simple. We built a pattern-matching based scheme to translate the Search Broker queries to different HTML forms.

The Search Broker's database query template consists of a list of form inputs, an extended regular expression against which the user's query is matched, and an optional block of Perl code used to post-process the assignments and set dynamic defaults. The majority of forms have only one field that is filled by the user's query; for example, a text input for keywords. Such a query is defined in the database very simply:

```
keywords = (.*)
```

Currently, about 90% of the subjects use this simple template. Parentheses are used to group regular expressions into a single value that is assigned to an input field. One can have many different fields. For example, some forms require separate fields for city and state. The query template we use in this case is:

```
city state = (.*),\s*(.*)
```

which translates "Tucson, AZ" into "city=Tucson state=AZ" and sends these fields to the search engine.

Sometimes the search form contains several required fields, a few of which are not essential to many queries. For example, in the flight schedule form, there are fields for departure time, departure day, departure month, favorite airline, etc. We picked only the departure and arrival cities as the two necessary fields and use the database query post-processing feature to assign defaults (e.g., two weeks from now, all airlines) to the rest of the fields. We believe that the ability to send a super-simple query

```
fly sfo jfk
```

outweighs the weakness of not including all the details. Of course, those extra fields can be used directly on the original form which the user can obtain from our help system. In a few cases we used several subjects for the same form with different defaults; for example, cd-by-artist and cd-by-title.

Each database entry can have multiple query templates. The templates are matched against the user's input in their database entry order, and the first one that matches the input is chosen. This allows us to support a variety of input formats, rather than force the user to guess which one we can recognize. For example, a server that provides a list of famous people born on a certain date might accept only one format of the date (e.g., in HTTP format: `mon=06&amp;day=25&amp;year=1953`), but templates could be provided to recognize and reformat all of the following:

```
famous-births 6/25/53
famous-births 25 June 1953
famous-births June 25th, 1953
```

The number of recognized formats is limited primarily by the imagination and energy of the librarian.

Because HTML forms often restrict field values through the use of radio or select input types, some search engines are very finicky about the format of queries they accept (for example, requiring the leading zero in the month input example above, something a user would not normally provide). This combination of extended regular expression pattern matching and arbitrary post-processing code has proven to be a very powerful solution to the problem of reformatting queries to meet these requirements.

5 Formatting Responses

After user input has been converted into an HTTP GET or POST request, the Search Broker contacts the remote server and retrieves the response. In the common case, we simply append the body of the response to an introduction that includes a description of and reference to the source search engine, and a form for additional queries. To avoid any appearance of claiming the content of responses as our own, we generally do not modify the retrieved material—inline graphics, scripts, and advertisements are left in their original place.

There are two cases where we must modify the returned material. The first is due to the latitude of HTML, which allows closing tags to be omitted, with an implicit close at the end of the document or entity. When we return results from multiple servers in response to a user's "search all related servers" request, we must append closing tags to the end of each response so that font changes, table layout, and other formatting directives do not affect the subsequent responses.

Modification is also used when a server is known to provide far more information than we want. An example of this is the flag subject. There is no search form interface to this information; the URL points to a long list of links to GIF-format images of flags. The Search Broker database entry provides a response post-processing feature, through which the retrieved material is split into sections based on some pattern (for example, end of line, start of HREF), and only the sections which match the user's query are returned. Although the current database format does not allow for arbitrary response post-processing as it does for query input reformatting, it can easily be extended to support well-defined actions such as filtering by pattern matching, or cutting out particular regions of a response.

6 Customization

So far we discussed the idea of the Search Broker in the context of one central server. But the same method can be applied to personal use, and it can lead to very powerful customizable search facilities on anyone's desktop. The current database, which includes over 400 subjects each with instructions, source, and examples of its use, together with the whole Search Broker software occupy less than half a megabyte. They make use of CGI scripts and assume an HTTP server, but there is no reason they cannot be used on any desktop through direct interaction with the browsers (e.g., on UNIX through Remote Control of UNIX Netscape, or on Windows through ActiveX).

Imagine having a search box somewhere on your desktop to which you can assign your own names to whichever subjects you choose, which sends your simple queries (e.g., fly sfo jfk) directly to the appropriate place, and presents the results automatically on your (favorite) browser. On the one hand, it can be thought of as a more convenient personal "hot list" of search facilities. You don't have to store (and know about) all the different search facilities; you don't have to load the selected search engine's own interface; you don't have to figure out how to fill out the selected form; all you need to do is use your own aliases (or try the standard ones). But it can be even more.

You could customize subjects such as "my-directions" which will give road directions to any address from *your* home. You could search local information. For example, you could have a subject called bookmarks which searches in your own bookmark list using aliases you give to the different pages or any word in their title or URL; you could have a subject called history which searches in the list of all URLs you've ever seen; you could have a subject called file which acts as a "Find File" application, searching for a file name on your local disk; you could have a birthday category searching your own lists of birthdays, a calendar searching your schedule, and so on. You could link some of these subjects so that a search for a person's name will be first conducted on your address book, then on your whole file system, then on your organization's database, then on the whole web. All of these searches could also be done at the same time, giving you the results in the right order. This can become your own personal oracle.

We are currently building a new Java-based version of the Search Broker to allow users to easily set up search scripts, whereever and whichever search engine they want to use.

7 Related Work

The seed of the Search Broker grew out of the Harvest project [Bowman *et al.*], where we attempted something similar, but ended up concentrating on the actual collection of data rather than the selection of servers. Harvest is an integrated system to collect, extract, organize, index, search, cache, and replicate information across the Internet. It is used by hundreds of sites to build "brokers" (our reuse of this term here is coincidental and unrelated) which serve collections of information gathered from many sources. Harvest has a special broker called the Harvest Server Registry (HSR), which maintains information about all brokers. The original intent was that there would be enough Harvest brokers to be used for most purposes, and that the HSR will lead people (by queries) to the right broker. That never happened, and Harvest never extended this idea to facilitate easy selection of servers.

The closest existing search facilities to the Search Broker are the lists of search engines, such as The Internet Sleuth and C/Net Search.com. We believe that our approach is an improvement, and has the potential, especially with personal customization, to be a significant step forward. The Search Broker is easier to use, and it does not require downloading or browsing large pages with forms. IBM InfoMarket also provides source selection through pull-down menus, limited to several publications that offer pay-per-view.

There has been a lot of work in the Information Retrieval area to support natural language queries (e.g., [Salton]). The search engine attempts to "understand" the essence of the query, to figure out which words are more important and which could be substituted more effectively, and how to assign different weights based not only on the query but also on the database. The success of these methods has been mixed. They can lead to unusual findings or embarrassing misses. In the context of the web, where the information is as diverse as possible and as unstructured as possible, it is very difficult to infer structure and patterns. We believe that our approach of handling the selection of search servers and assignment of subjects to them by hand is both feasible and desirable.

There has been a lot of research in the database community related to source selection and interaction between separate databases. Wiederhold [Wiederhold] introduced the general notion of *mediators*. Levy *et al.* [Levy1, Levy2] developed tools (e.g., the Information Manifold) to allow complex queries across different databases. Their most pressing problem is how to negotiate with complex database schemas, a problem we don't have (yet).

The idea of a *MetaCrawler* to combine results from several sources was introduced by Selberg and Etzioni [Selberg]. We currently just concatenate results if they are obtained from several sources. Incorporating this technology would certainly help.

There has also been research in the expert systems area in source selection. For example, the Reference Expert from the University of Houston [Bailey] was developed to help users select the right reference material in the library for their questions, and QPEA (Query Planning Environment Assistant) [Huffman] is being developed at Price Waterhouse to help specialists select and combine data sources.

8 Conclusions

The Search Broker offers a balance of focused search, ease of use, and generality. It opens the door to a more significant involvement of experts in the organization of search. It explores the middle ground between completely automated search systems on the one hand and manual collection of information on the other. The web searching problem is too big a problem to be solved by one tool or even one model. The Search Broker presents a slightly different model than the existing ones, and for some users and some purposes it gives excellent results with much less effort than other approaches. We strongly believe that the Search Broker model can capture an important niche and encourage people to make available more specialized search facilities, which will benefit everyone.

9 Acknowledgements

This work was supported in part by NSF grant CCR-9301129, and by the Advanced Research Projects Agency under contract number DABT63-93-C-0052. The information contained in this paper does not necessarily reflect the position or the policy of the U.S. Government or other sponsors of this research. No official endorsement should be inferred.

References

[Bowman *et al.*] C. Mic Bowman, Peter B. Danzig, Darren R. Hardy, Udi Manber, and Michael F. Schwartz, `<URL:ftp://ftp.cs.colorado.edu/pub/cs/techreports/schwartz/Harvest.Conf.ps.Z>`. The Harvest Information Discovery and Access System, *Computer Networks and ISDN Systems* **28** (1995) pp. 119-125. (An early version appeared in the *Proceedings of the Second International World Wide Web Conference*, pp. 763-771, Chicago, Illinois, October 1994.)

[Salton] Gerard Salton, *Automatic Text Processing*, Addison Wesley, Reading, Mass, 1989.

[Wiederhold] G. Wiederhold, "Mediators in the Architecture of Future Information Systems," *IEEE Computer*, **25** (1992), pp. 38-49.

[Levy1] Alon Y. Levy, Anand Rajaraman and Joann J. Ordille, `<URL:http://www.research.att.com/~levy/vldb96-im.ps.Z>`. Querying Heterogeneous Information Sources Using Source Descriptions, *Proceedings of the 22nd International Conference on Very Large Databases*, VLDB-96, Bombay, India, September, 1996

[Levy2] A.Y. Levy and J.J. Ordille, `<URL:http://cm.bell-labs.com/cm/cs/doc/95/11-01.ps.gz>`. An Experiment in Integrating Internet Information Sources, *AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval*, Cambridge, MA (November 1995).

[Selberg] Erik Selberg, and Oren Etzioni, `<URL:http://www.cs.washington.edu/research/projects/softbots/papers/metacrawler/www4/html/Overview.html>`. Multi-Service Search and Comparison Using the MetaCrawler, Proceedings of the 4th International World Wide Web Conference, 1995.

[Bailey] Charles W. Bailey Jr. and Robin N. Downes, `<URL:http://educom.edu/stories.101/Intelligent-Reference.txt>`. Intelligent Reference Information System (IRIS), in *101 Success Stories of Information Technology in Higher*

Education: The Joe Wyatt Challenge, edited by Judith Boettcher, McGraw-Hill, New York, 1993.

[Huffman] Scott B. Huffman and David Steier, ”A Navigation Assistant for Data Source Selection and Integration,” in *Working Notes of AAAI-95 Fall Symposium Series on AI Applications in Knowledge Navigation and Retrieval*, pp. 72-77, Cambridge, MA, 1995.

[Fielding] Fielding, *et al.* Hypertext Transfer Protocol – HTTP/1.1. Network Working Group Request for Comments 2068.