Connecting Diverse Web Search Facilities

Udi Manber Peter A. Bigot
Department of Computer Science
University of Arizona
Tucson, AZ 85721
{udi,pab}@cs.arizona.edu

Abstract

The World Wide Web is now providing hundreds of powerful search facilities, often for free, enabling people to access an enormous amount of information. The most successful search facilities to date are the global flat databases that attempt to put everything in one place. We believe that this approach will not scale. This paper describes techniques and software that we developed to connect together many different diverse search facilities, allowing people to focus and customize their search much better and increase the precision of the results.

1 Introduction

With the explosion of available information, search is becoming one of the most important computer activities. Getting the right information at the right time with minimal effort is the main competitive edge in many businesses. The most commonly-used search facilities on the web are the global search engines such as Altavista, Infoseek, Excite, HotBot, and Lycos. They collect as much of the web as they can into one large database and provide keyword-based search. They have become amazingly powerful, but they are still lacking. Taking the whole universe as one flat data space and searching it with keywords has inherent limitations. Another even more successful approach is that of Yahoo, which collects and classifies web pages manually with the help of librarians. The noise generated by this approach is much smaller but the coverage is smaller too, and it is still often time consuming to find things. The challenge is to provide users with ways to focus and customize their search better without making it too difficult or too inefficient.

The first part of the paper describes 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 by connecting together many diverse search facilities into one common interface. It also strikes a balance between automated blind collection and manual collection of quality information. We have implemented this method and provide it as a service called the Search Broker [9]. The second part of the paper describes the next generation of this approach, called the "Universal Search Interface" (USI), which is currently work in progress.

Our research emphasizes simplicity suitable for non-specialist users, rather than complexity and generality. We want to provide access through one interface, in many different customizable ways, to hundreds of existing search facilities. Although this problem can be viewed as an instance of database integration, which has been studied extensively, we are not attempting to integrate all the Web's different databases *per se*: for example,

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

we are not attempting to support queries like the join of two separate databases. Instead, we facilitate querying many databases together, either in parallel or sequentially. By limiting ourselves to issues related to straightforward search tasks rather than complex database queries, we believe we can design a system that will be powerful (although not as powerful as an integrated database), and more important, easy to use.

2 The Search Broker

The weaknesses of the global web search engines are most glaring when one looks for answers to specific reference questions, which are hard to answer based on keywords and flat search. For example,

1. How much fat is there in a pepperoni pizza?
2. How do you say "search" in Latin?
3. How do you delete a directory in UNIX?
4. Give me a list of hotels in Phoenix.

The Search Broker 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 desired information is sought within this database. This is not a new approach, of course. It is similar to using the library subject card catalog to find the right shelf, or using Yahoo to find the right category. The novelty of the Search Broker is that it combines 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 otherwise available in any one place on the web.

Consider again the four 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; perhaps not precisely, but often closely. For example, the questions above could be answered more precisely if they were of the form

1. "Subject: nutrition; Query: pizza", or
2. "Subject: english-to-latin; Query: search", or
3. "Subject: unix; Query: delete directory", or
4. "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.

The Search Broker 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. 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 the 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 opposed to the number of web pages or 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,” or “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”. 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; see [9] for details.

3. An HTTP 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 [11], GlimpseHTTP, WebGlimpse [10], and Harvest [2] 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.

polish-english 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 1994 Geo, Prizm gives the blue book value for this model (“,” 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.

demographics health food gives several articles on issues of demographics (and marketing) related to health food.

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. It explores the middle ground between completely automated search systems on the one hand and manual collection of information on the other. It opens the door to a more significant involvement of experts in the organization of search. The web searching problem is too big to be solved by one tool or even one model. 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 most of the time, queries 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 opened for public use on the web in July 1997. It can be found at `http://sb.cs.arizona.edu/sb/index.html`.

3 The Universal Search Interface

The Universal Search Interface (USI) is work in progress, which extends the Search Broker work in two directions. First, it is designed as a client rather than a server tool, allowing users to pick the search facilities they want to use and to customize them. We believe this to be a crucial usability feature: in the course of our daily use of the web, we found ourselves creating personal Search Broker databases that contained the sites of special interest to us, with our own aliases to identify them. Second, it provides tools to connect several search engines together, either in parallel or sequentially, to pipe results between searches, and to link web and local searches. The goal is to allow users to combine all the search facilities available to them —whether they search their own machine, their local network, or the Internet—in a powerful and customizable way via one common interface. The design will allow even inexperienced users to construct "search scenarios"—customized combinations of several search facilities—based on their preferences and needs. Scenarios may also be built by professionals and distributed like any other software. We will provide ways to "publish" search scenarios.

3.1 Examples of search Scenarios

Search for experts: Suppose that you are an editor looking for a referee to a paper about XYZ. You may first search for XYZ in relevant databases (e.g., The Collection of Computer Science Bibliographies at `http://liinwww.ira.uka.de/bibliography/index.html`, which uses our Glimpse, or the DBLP bibliography at `http://sunsite.ust.hk/dblp/db/index.html`) to find who has done work in that area. You then collect all authors' names and search for each one of them in either a special local file (e.g., of previous referees), or through all your files (e.g., using Glimpse). This is an example of piping results of one query into another after appropriate filtering.

Search for citations: Suppose that you want to find articles on the web that cite a given article. You first search one of the global web search engines, possibly asking for more than the usual number of first hits (e.g., 100 instead of 10) by combining together several requests. You then filter the results by extracting the URL from each hit. You fetch each page, and run agrep to find whether the citation is really there and extract the title and URL of that page. This is an example of several pipes involving web search and local search.

These examples involve common tasks that people currently have to do by hand with great effort. A USI will allow users to define these search tasks and then automate them.

3.2 Overall Design

The key to a successful implementation of a USI is the ability to handle three distinct issues:

Generality: A USI must be able to accommodate most search facilities, without changing them or even having access to their source code. The only assumption is that the user has permission and ability to perform the search through some mechanism.

Customizability: A USI must allow users to set their own preferences and customize their own search processes. Such customization must be easy to accomplish; in particular, it should not require any programming or any special knowledge of the search facilities (besides how to use them).

Ease of integration: Adding or extending search facilities should be relatively easy.

As is often the case, it is not likely that all three issues can be solved together perfectly, so making the right tradeoffs is crucial. Our approach to the design, to paraphrase an old saying attributed to Einstein, is to "make it as simple as we can, but not simpler than that." We will provide expansion room for generality and complexity, but the starting point will be as simple as possible. A good test of the simplicity of a USI is that it can be used to maintain hot lists (or bookmarks or favorite lists) as conveniently as they are maintained by the current browsers, but with more features. Next, users will learn to use a USI to build their own Search Broker (by extending items of a hot list to be "active" and trigger a search), and then they will be to able to construct complex scenarios. By concentrating on the most common type of searching interfaces, we hope to make the USI accessible to most users. Our challenge, in a sense, is to resist complexity by emphasizing the special flavor of search interfaces.

The USI is an umbrella encompassing several concepts and tools. One of the most important concepts is that of a "Search Object." Each search engine will have a corresponding search object which will encompass its interface, options, and formatting of results. Search objects will have associated input and output schemas, which will usually be very simple (e.g., a string in and a string out). There will also be schema-converting objects which extract, filter, and reformat the results of intermediate searches. Users can collect search objects, customize them, combine several of them into complex search objects, and build their own "super" interface that will give them easy access to all the scenarios they use in the way they want. We will build tools to allow users to construct search objects easily from given interfaces. In particular, for web-based CGI searches, users will be able to input the URL and fill in default options and/or fields, and the corresponding search object will be built for them (the Search Broker already provides a similar facility for its maintainers).

Selecting search objects and combining them to form scenarios will be done through a GUI which will be the main customization facility for users of a USI. Users will be able easily to pick a scenario from a USI and activate or modify it. The GUI will also provide type checking to guarantee consistent schemas, search capabilities on its own content, and tools for easy modification, import/export, and organization.

3.3 Putting Search Objects Together

The ability to compose search objects into scenarios is one of the most important features of a USI. In many cases, a user may wish to query multiple servers and combine the results, or to use the results of one query (e.g., to a web indexer) as the input to another (e.g., automatically retrieve the pages suggested by the indexer). A user may wish to examine intermediate results before passing them on to another stage, or may allow the system to run automatically. This kind of interaction extends the usual web browsing to include personal web "actions" triggered by the users. By treating stages consistently as implementations of a search object, we can support both arbitrary result reformatting, and complex searches where one output is the input of the next stage. Consistent use of the interface allows complex scenarios of search objects to be encapsulated and treated as search objects in their own right, providing a functionality similar to libraries of subroutines in programming languages.

To facilitate the combination of search objects, we use "translator objects" and "filtering objects". The distinction between search, translator, and filter objects is only for descriptive purposes—they are all objects with specific input and output schemas. A translator object translates between different schemas. For example, a translator may take the results of a particular search engine in HTML and extracts from it a set of fields (e.g., URLs,

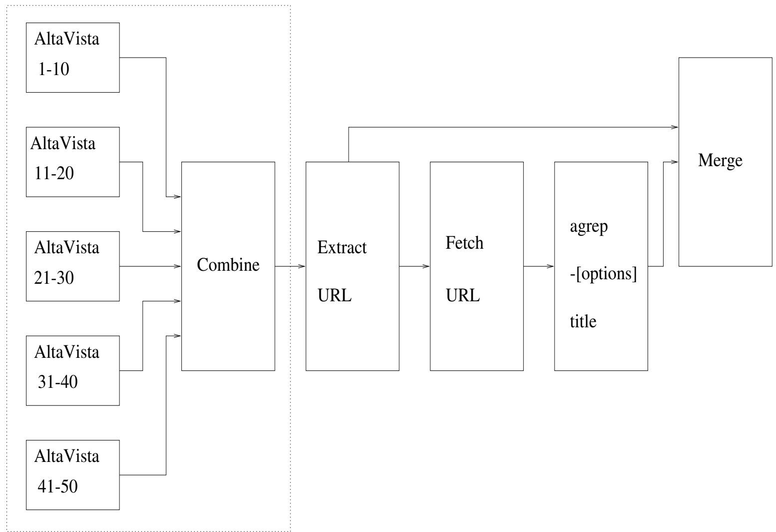

Figure 1: Putting together search objects to find citations

titles, abstracts, dates, scores). We expect this type of translator to be the most common one, and we will implement such translators for all popular search engines. We will also provide tools to allow users to build such translators for other interfaces. Translating from one complex database schema to another is very difficult in general, but we expect most of our search applications to require only simple translations, which is why this approach is feasible.

A filter object extracts some specific information. For example, a URL filter takes as input an HTML page and outputs the list of all URLs contained in that page. Other filters take a list of URLs and remove from it all those that the user has seen before (e.g., by consulting the history.db file), or take only those that appear in the user's bookmark list, or merge two URL lists together. The example of finding citations discussed earlier is shown as a combination of different objects in Figure 1. The area inside the dashed square is an example of part of a scenario that will most likely be encapsulated into a stand-alone search object.

The forms of customization we are discussing are instances of a black-box design [6], where the user is familiar with the interface to and capabilities of each USI object, but is unaware of and unconcerned with the internal implementation by which the object works. This is appropriate to meet the goals of a system used by people who are unfamiliar with programming, but who still want to customize searches to their own areas of interest. The configuration mechanism described above allows for specification of USI object options and supports complex scenarios by linking existing objects that perform searches and filter results. However, it does not provide for the ability to change the implementation of the interface methods. Nor does it allow users to define a search object that is not related to an existing one. Building from scratch a search object for Glimpse, for example, can be done only by a programmer. Selecting the right options can be done by anyone. This is a compromise we believe we must make.

4 Related Work

This work touches on many areas in computer science, including databases, user interfaces, networks, and algorithm design. For lack of space, we'll mention only a few related systems.

The seed of the Search Broker grew out of the Harvest project [2], where we attempted something similar, but ended up concentrating on the actual collection of data rather than the selection of servers. The closest existing search facilities to the Search Broker are the lists of search engines, such as The Internet Sleuth (`http://www.isleuth.com/`) and C/Net Search.com (`http://www.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 USI can be thought of as an example of a *mediator*, a concept introduced by Wiederhold [13], and our search objects are similar to database *wrappers*. The Garlic system from IBM [4] and the use of mediators in TSIMMIS [5] are good examples. The main difference is in our emphasis on simple searches rather than on general database queries. As a result, our system is much simpler, and we can hope to allow non-experts to build specific scenarios, and easily customize them. Levy et al [8] 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).

Software agents, especially web agents, promise some of the features we want a USI to have, and address some of the same issues. Their emphasis, however, is on learning, and search is just one task in the context of many other tasks users perform. Our goal is to build the right infrastructure specifically for search. The two agent projects that are most relevant to this work are WBI from IBM Almaden [3] and LiveAgent [7] from AgentSoft Inc. Both work through HTTP proxies and serve as intermediaries between users and the web. WIDL from WebMethods [1], although not directly a system for end users, offers some of the scripting mechanisms we provide for the USI, and can be used by specialists to build similar search objects for scenarios of interest to businesses and organizations. Another very successful related work in the agents area is the "meta-search" [12] approach, which queries several search engines and combines the results. We believe that a USI will be instrumental in evaluating new meta-search techniques. We expect that a USI will be useful for agents, and hope that some of the agent software will be useful for us.

References

[1] C. A. Allen, "Automating the Web with WIDL," in "XML: Principles, Tools, and Techniques", O'Reilly, 1997.

[2] Bowman C. M., P. B. Danzig, D. R. Hardy, U. Manber, and M. F. Schwartz, "The Harvest Information Discovery and Access System," *Computer Networks and ISDN Systems* **28** (1995) pp. 119-125.

[3] Barrett R., P. P. Maglio, and D. C. Kellem, "How to Personalize the Web," *Proceedings of the 1997 SIGCHI Conference on Human Factors in Computer Systems*, Atlanta, Georgia (March 1997).

[4] Carey M.J. et al, "Towards Heterogeneous Multimedia Information Systems: The Garlic Approach," *Fifth International Workshop on Research Issues in Data Engineering* Taipei, Taiwan (March 1995), pp. 124-131.

[5] Garcia-Molina H., Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, and J. Widom, "The TSIMMIS approach to mediation: Data models and Languages," *Journal of Intelligent Information Systems,* **8**(2) (1997), pp. 117-132.

[6] Johnson R.E., and B. Foote, "Designing Reusable Classes," *The Journal of Object-Oriented Programming* **1**(2) (June/July 1988), pp. 22-35.

[7] Krulwich B., "Automating the Internet – Agents as User Surrogates," *IEEE Internet Computing*, **1** (4) (July 1997).

[8] Alon Y. Levy, Anand Rajaraman and Joann J. Ordille, "Querying Heterogeneous Information Sources Using Source Descriptions," *Proceedings of the 22nd International Conference on Very Large Databases, VLDB-96,* Bombay, India, September, 1996.

[9] Manber U., and P. Bigot, "The Search Broker," *First Usenix Symp. on Internet Technologies and Systems,* Monterey, CA (December 1997), pp. 231-240. `http://sb.cs.arizona.edu/sb`

[10] Manber U., M. Smith, and B. Gopal, "WebGlimpse – Combining Browsing and Searching," *Usenix 1997 Annual Technical Conference,* Anaheim, CA (January 1997), pp. 195-206.

[11] Manber U., and S. Wu, "GLIMPSE: A Tool to Search Through Entire File Systems," *Usenix Winter 1994 Technical Conference,* (best paper award) San Francisco (January 1994), pp. 23-32. (ftp://ftp.cs.arizona.edu/reports/1993/TR93-34.ps.Z)

[12] E. Selberg, and O. Etzioni, "Multi-Service Search and Comparison Using the MetaCrawler," *Proceedings of the 4th International World Wide Web Conference* Boston, MA (December 1995).

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