Multi-Service Search and Comparison Using the MetaCrawler

Erik Selberg Oren Etzioni
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195
{selberg, etzioni}@cs.washington.edu
http://www.cs.washington.edu/homes/{selberg, etzioni}
http://www.cs.washington.edu/research/metacrawler

October 9, 1995

Abstract

Standard Web search services, though useful, are far from ideal. There are over a dozen different search services currently in existence, each with a unique interface and a database covering a different portion of the Web. As a result, users are forced to repeatedly try and retry their queries across different services. Furthermore, the services return many responses that are irrelevant, outdated, or unavailable, forcing the user to manually sift through the responses searching for useful information.

This paper presents the MetaCrawler, a fielded Web service that represents the next level up in the information "food chain." The MetaCrawler provides a single, central interface for Web document searching. Upon receiving a query, the MetaCrawler posts the query to multiple search services in parallel, collates the returned references, and loads those references to verify their existence and to ensure that they contain relevant information. The MetaCrawler is sufficiently lightweight to reside on a user's machine, which facilitates customization, privacy, sophisticated filtering of references, and more.

The MetaCrawler also serves as a tool for comparison of diverse search services. Using the MetaCrawler's data, we present a "Consumer Reports" evaluation of six Web search services: Galaxy[5], InfoSeek[1], Lycos[15], Open Text[20], WebCrawler[22], and Yahoo[9]. In addition, we also report on the most commonly submitted queries to the MetaCrawler.

Keywords: MetaCrawler, WWW, World Wide Web, search, multi-service, multi-threaded, parallel, comparison

This paper appears in the Proceedings of the 1995 World Wide Web Conference

1 Introduction

Web search services such as Lycos and WebCrawler have proven both useful and popular. As the Web grows, the number and variety of search services is increasing as well. Examples include: the Yahoo "net directory"; the Harvest home page search service[7]; the Query By Image Content service[12]; the Virtual Tourist[24], a directory organized by geographic regions; and more. Since each service provides an incomplete snapshot of the Web, users are forced to try and retry their queries across different indices until they find appropriate responses. The process of querying multiple services is quite tedious. Each service has its own idiosyncratic interface which the user is forced to learn. Further, the services return many responses that are irrelevant, outdated, or unavailable, forcing the user to manually sift through the responses searching for useful information.

This paper presents the MetaCrawler, a search service that attempts to address the problems outlined above. The premises underlying the MetaCrawler are the following:

- No single search service is sufficient. Table 2 shows that no single service is able to return more then 45% of the references followed by users.
- Many references returned by services are irrelevant and can be removed if the user is better able to express the query. Table 3 shows that up to 75% of the references returned can be removed if the user supplies a more expressive query.
- Low-quality references can be detected and removed fairly quickly. Table 4 shows that an average of about 100 references can be verified in well under 2.5 minutes, while simple collation and ranking takes under 30 seconds.
- These features will be used by the Web's population. The MetaCrawler is receiving over 7000 queries per week, and that number is growing, as shown in Figure 1.
- The MetaCrawler log facilitates an objective evaluation and comparison of the underlying search services. Tables 5-8 detail trade-offs between the services. For example, Lycos returns over 5% more followed references than any other service, yet WebCrawler is the fastest, taking an average of 9.64 seconds to return answers to queries.

The MetaCrawler logs also reveal that people search for a wide variety of information, from "A. H. Robins" to "zyx music." While the most common queries are related to sex and pornography, these only account for under 4% of the total queries submitted to the Metacrawler as shown in Table 1. Nearly half of all queries submitted are unique.

The remainder of this paper is organized as follows: the design and implementation of the MetaCrawler are described in Section 2. Experiments to validate the above premises are described in Section 3. We discuss related work in Section 4, and our ideas for future work and potential impact appear in Section 5. We conclude with Section 6.

2 The MetaCrawler

The MetaCrawler[1] is a free search service used for locating information available on the World Wide Web. The MetaCrawler has an interface similar to WebCrawler and Open Text in that it allows users to enter a search string, or *query*, and returns a page with click-able references or *hits* to pages available on the Web. However, the internal architecture of the MetaCrawler is radically different from the other search services.

Standard Web searching consists of three activities:

- *Indexing* the web for new and updated pages, a process that demands substantial CPU and network resources.
- *Storage* of the Web pages retrieved into an index, which typically requires a large amount of disk space.
- *Retrieval* of pages matching user queries. For most services, this amounts to returning a ranked list of page references from the stored index.

Standard search services create and store an index of the Web as well as retrieve information from that index. Unlike these services, the MetaCrawler is a *meta-service* which uses no internal database of its own; instead, it relies on other external search services to provide the information necessary to fulfill user queries. The insight here is that by separating the retrieval of pages from indexing and storing them, a lightweight application such as the MetaCrawler can access multiple databases and thus provide a larger number of potentially higher quality references than any search service tied to a single database.

Another advantage of the MetaCrawler is that it does not depend upon the implementation or existence of any one search service. Some indexing mechanism is necessary for the Web. Typically, this is done using automated robots or spiders, which may not necessarily be the best choice[13]. However, the underlying architecture of the search services used by the MetaCrawler is unimportant. As long as there is no central complete search service and several partial search services exist, the MetaCrawler can provide the benefit of accessing them simultaneously and collating the results.

The MetaCrawler prototype has been publicly accessible since July 7, 1995. It has been steadily growing in popularity, logging upwards of 7000 queries per week and increasing. The MetaCrawler currently accesses six services: Galaxy, InfoSeek, Lycos, Open Text, WebCrawler, and Yahoo. It works as follows: given a query, the MetaCrawler will submit that query to every search service it knows in parallel. These services then return a list of references to WWW pages, or hits. Upon receiving the hits from every service, the MetaCrawler *collates* the results by merging all hits returned. Duplicate hits are listed only once, but each service that returned a hit is acknowledged. Expert user-supplied sorting options are applied at this time. Optionally, the MetaCrawler will *verify* the information's existence by loading the reference. When the MetaCrawler has loaded a reference, it is then able to re-score the page using supplementary query syntax supplied by the user.

When the MetaCrawler has finished processing all of the hits, the user is presented with a page consisting of a sorted list of references. Each reference contains a click-able hypertext link to the reference, followed by local page context (if available), a confidence score, verified keywords, and the actual URL of the reference.

[1] http://metacrawler.cs.washington.edu:8080

Each word in the search query is automatically boldfaced. So that we may determine which references are followed, each click-able link returned to the user points not to the reference, but to a script which logs that the reference was followed and then refers the user's browser to the correct URL.

Querying many services and simply collating results will return more results than any one service, but at the cost of presenting the user with more irrelevant references. The MetaCrawler is designed to increase *both* the number of hits and relevance of hits returned. The MetaCrawler yields a higher proportion of *relevant* hits by using both a powerful query syntax as well as expert options so that users can more easily instruct the MetaCrawler how to determine the quality of the returned references. The query syntax used specifies required and non-desired words, as well as words that should appear as a phrase. The expert options allow users to rank hits by physical location, such as the user's country, as well as logical locality, such as their Internet domain.

2.1 User Interface

While giving the user a Web form with added expressive power was easy, presenting the user with a form that would facilitate actually using the novel features of the MetaCrawler proved to be a challenge. We strove for a balance between a simple search form and an expressive one, keeping in mind interface issues mentioned by service providers[23].

In our early designs, we focused on syntax for queries with several additional options for improving the result. This syntax was similar to InfoSeek's query syntax: parentheses were used to define phrases, a plus sign designated a required word, and a minus designated a non-desired word. For example, to search for "John Cleese," naturally requiring that both "John" and "Cleese" appear together, the syntax required was the unwieldy `(+John +Cleese)`. Not surprisingly, we discovered that while most users attempted to use the syntax, they often introduced subtle syntactical errors causing the resulting search to produce an entirely irrelevant set of hits.

In our current design, we have reduced the need for extra syntax, and instead ask the user to select the type of search. The three options are:

- *Search for words as a phrase:*
  Treat the query text as a single phrase, and attempt to match the phrase in pages retrieved, e.g. "Four score and seven years ago."
- *Search for all words:*
  Attempt to find each word of the query text somewhere in the retrieved pages. This is the equivalent of logical "and."
- *Search for any words:*
  Attempt to find any word of the query text in the retrieved pages. This is the equivalent of logical "or."

The older syntax is still supported, although it is not advertised prominently on the main search page, save for the minus sign, which was the most widely used element of the query syntax. Since we changed the search page to this new design, the number of malformed requests has dropped significantly.

In addition to the query entry box, we maintain various expert options which can be activated via menus. The MetaCrawler currently uses two menus to provide extra expressiveness. The first describes a coarse grain Locality, with options for the user's continent, country, and Internet domain, as well as options to select a specific continent. The second menu describes the sundry Internet domain types, e.g. `.edu`, `.com`, etc. These options allow users to better describe what they are looking for in terms of where they believe the relevant information will be.

2.2 Client-Server Design

Current search services amortize the cost of indexing and storing pages over hundreds of thousands of retrievals per day. In order to field the maximal number of retrievals, services devote minimal effort to responding to each individual query. Increases in server capacity are quickly gobbled up by increases in pages indexed and queries per day. As a result, there is little time for more sophisticated analysis, filtering, and post-processing of responses to queries.

By decoupling the retrieval of pages from indexing and storage, the MetaCrawler is able to spend time performing sophisticated analysis on pages returned. The MetaCrawler just retrieves data, spending no time indexing or storing it. Thus, the MetaCrawler is relatively lightweight. The prototype, written in C++, comes to only 3985 lines of code including comments. It does not need the massive disk storage to maintain an index nor does it need the massive CPU and network resources that other services require. Consequently, a MetaCrawler client could reside comfortably on an individual user's machine.

An individualized MetaCrawler *client* that accesses multiple Web search services has a number of advantages. First, the user's machine bears the load of the post-processing and analysis of the returned references. Given extra time, post-processing can be quite sophisticated. For example, the MetaCrawler could divide references into clusters based on their similarity to each other, or it could engage in *secondary search* by following references to related pages to determine potential interest. Second, the processing can be customized to the user's taste and needs. For example, the user may choose to filter advertisements or parents may try to block X-rated pages. Third, the MetaCrawler could support scheduled queries (e.g., what's new today about the Seattle Mariners?). By storing the results of previous queries on the user's machine, the MetaCrawler can focus its output on new or updated pages. Finally, for pay-per-query services, the MetaCrawler can be programmed with selective query policies (e.g., "go to the cheapest service first" or even "compute the optimal service querying sequence").

Organizations may choose to have an institutional MetaCrawler with enhanced caching capabilities, on the presumption that people within an organization will want to examine many of the same pages. The cache could also facilitate local annotations, creating a collaborative filtering and information exchange environment of the sort described elsewhere [17].

Finally, while our prototype MetaCrawler depends on the good will of the underlying services, a MetaCrawler client would not. In the future, an underlying service may choose to block MetaCrawler requests, which are easily identified. However, it would be nearly impossible to distinguish queries issued by a MetaCrawler client versus queries made directly by a person.

No.	Query	Times Issued	No.	Query	Times Issued
1	sex	533	6	pornography	112
2	erotica	219	7	erotic	105
3	nude	217	8	porno	89
4	porn	158	9	adult	89
5	penthouse	127	10	playboy	67

Table 1: Top Ten Queries Issued to the MetaCrawler

"Times Issued" lists the number of times the corresponding query was issued from July 7 through Sept. 30, 1995. Note that while each query is sexually related, the combined total amounts to less than 4% of the total queries processed by the MetaCrawler. Also, 46% of the queries issued were unique.

2.3 Common Usage

One of the frequently asked questions regarding search on the Internet is "What are people searching for?" Table 1 summarizes the top ten repeated queries out of a total of 50,878 queries made from July 7 through September 30. Each query in the top ten is related to sex. However, the combined top ten queries amount only to 1716 queries out of 50,878 total queries, or 3.37%. Further, 24,253 queries, or 46.67%, were not repeated.

3 Evaluation

The MetaCrawler was released publicly on July 7, 1995. Averages and percentages presented in this paper are based on the 50,878 completed queries, starting on July 7 and ending September 30, except those in reference to Open Text, which are based on 19,951 completed queries starting September 8, when Open Text was added to the MetaCrawler's repertoire. The log results from seven days were omitted due to a service changing its output format, causing that service to return no references to the MetaCrawler even though the service was available. The MetaCrawler is currently running on a DEC Alpha 3000/400 under OSF 3.2.

The first hypothesis we confirmed after we deployed the MetaCrawler was that sending queries in parallel and collating the results was useful. To confirm this, we used the metric that references followed from the page of hits returned by the MetaCrawler contained relevant information. We calculated the percentage of references followed by users for each of the search services. Table 2 demonstrates the need for using multiple services; while Lycos did return the plurality of the hits that were followed, with a 35.43% share (42.17% in the last month recorded), slightly under 65% of the followed references came from the other five services. Skeptical readers may argue that service providers could invest in more resources and provide more comprehensive indices to the web. However, recent studies indicate the rate of Web expansion and change makes a complete index virtually impossible[16].

	% followed Jul. 7 - Sept. 30	% followed Sept. 8 - 30
Lycos	35.43	42.17
WebCrawler	30.76	25.74
InfoSeek	18.55	15.70
Galaxy	17.10	15.60
Open Text	n/a	14.70
Yahoo	10.67	6.59

Table 2: Market Shares of Followed References

This table shows the percentage each service has of the total followed references. References returned by two or more services are included under each service, which is why the columns sum to over 100%. This demonstrates that a user who restricts his or her queries to a single service will miss most of the relevant references.

We then analyzed the data to determine which, if any, of the added features of the MetaCrawler were helping users. The metric we used was the number of references pruned. Table 3 shows the average number of references removed for each advanced option.

Using syntax for required or non-desired words typically reduces the number of returned results by 40%. Detecting dead pages allowed the removal of another 15%. Finally, the expert options were very successful in removing unwanted references. When all of these features are used in conjunction, up to 75% of the returned references can be removed.

3.1 MetaCrawler Benchmarks

We have shown that the MetaCrawler improves the quality of results returned to the user. But what is the performance cost? Table 4 shows the average times per query, differentiating between having the MetaCrawler simply collate the results or verify them as well.

Table 4 shows that the MetaCrawler finished relatively quickly. The average time to return collated results is a little over 5 seconds longer than the slowest service as shown by Table 8. This is to be expected given the percentage of the time a service times out, which causes the MetaCrawler to wait for a full minute before returning all the results. We are pleased with the times reported for verification. Our initial prototype typically took five minutes to perform verification. We recently began caching retrieved pages for three hours, and have found that caching reduces the average verification time by nearly one-half. We are confident that this time can be further reduced by more aggressive caching as well as improvements in the thread management used by the MetaCrawler.

Since the MetaCrawler was publicly announced, the daily access count has been growing at a linear rate. We are also pleased with an increased use of the user options. Figure 1 plots the data points for the weeks

Feature	% of Hits Removed
Syntax	39.79
Dead	14.88
Expert	21.49

Table 3: Effect of Features Removing Irrelevant Hits

This table shows the percentage of hits removed when a particular feature was used. "Syntax" refers to queries that were removed due to sophisticated query syntax (e.g., minus for undesired words); "Dead" refers to unavailable or inaccessible pages, and "Expert" refers to hits removed due to restriction on the references' origins.

	Wall Clock Time	User Time	System Time	Lag Time
Collated	25.70	0.32	1.87	23.51
Verified	139.30	22.72	4.50	112.08

Table 4: Average Time for MetaCrawler Return Results

All times are measured in seconds. "Wall Clock Time" is the total time taken for an average query, and is broken down into User, System, and Lag Time. "User Time" is the time taken by the MetaCrawler program, "System Time" the time taken by the operating system, and "Lag Time" the time taken for pages to be downloaded off the network.

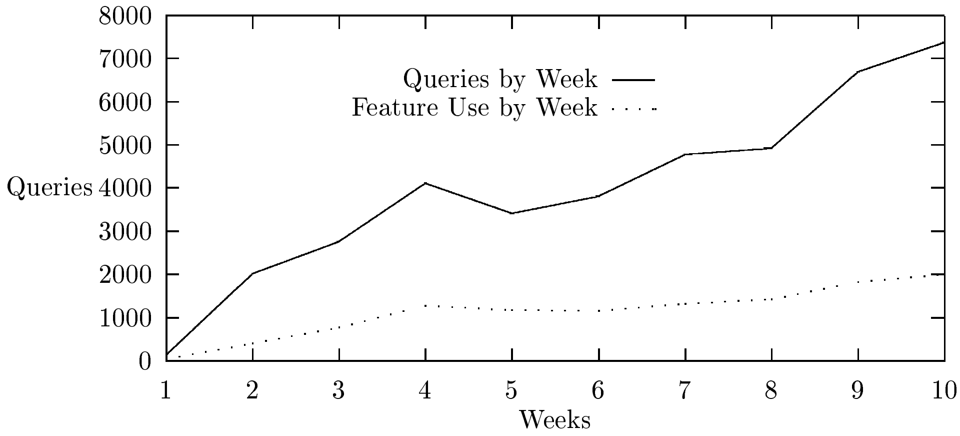

Figure 1: Queries per week from July 7 - Sept. 30

beginning July 7 until September 30. "Feature Use by Week" shows the number of queries where any of the MetaCrawler's advanced features, such as verification, were used.

3.2 Search Service Comparison

In addition to validating our claims, the MetaCrawler's logs also allow us to present a "Consumer Reports" style comparison of the search services. We evaluate each service using three metrics:

- *Coverage:* How many hits will be returned on average?
- *Relevance:* Are hits returned actually followed by users?
- *Performance:* How long does each service take, and how often does it time out?

3.2.1 Coverage

Given a pre-set maximum on the number of hits returned by each service, we measured both the percentage of references returned as well as references unique to the service that returned them. Thus, 75% returned with 70% unique shows that on average a service returns 75% of its maximum allowed, with 70% of those hits being unique.

Tables 5 and 6 details our findings in terms of raw content. It shows that with default parameters, Open Text returns 80% of the maximum hits allowed, with nearly 89% of those hits being unique. Lycos and WebCrawler follow, also returning over 70%, with slightly over 90% of those hits being unique. Yahoo has particularly poor performance on the total hits metric, but this was not surprising to us. We included

	% of Max Hits Returned	Ave. Hits Returned / Maximum Allowed
Open Text	80.0%	8.0 / 10
Lycos	76.3%	19.1 / 25
WebCrawler	70.2%	17.6 / 25
Galaxy	56.9%	28.4 / 50
InfoSeek	43.5%	4.4 / 10
Yahoo	11.1%	11.1 / 100

Table 5: Returned References by Service

The first column shows the percentage of the maximum hits allowed that each service returned. Each percentage was calculated by dividing the average hits returned by the maximum allowed for that service, as shown in the second column. This percentage is a measure of how many hits a service will provide given a pre-set maximum. The MetaCrawler used different maximum values for services, as some had internal maximum values, and others would either accept only certain maximums or none at all.

Yahoo on the hypothesis that people search for subjects, such as "Mariners Baseball," which Yahoo excels at. However, it turned out this hypothesis was incorrect, as people tended to use the MetaCrawler to search for nuggets of information, such as "Ken Griffey hand injury." Yahoo does not index this type information, and thus shows poor content. Presumably, most topic searches go to Yahoo directly.

Although each service returns mostly unique references, it is not clear whether those references are useful. Further, unique references are not necessarily unique to a service's database, as another service could return that reference given a different query string.

3.2.2 Relevance

To measure relevance, two metrics are used. The first is which service is returning the most references that people follow. This is shown by Table 2. The second metric is what percent of references returned by each service are people following. Table 7 summarizes these calculations. It shows that nearly 6% of all references returned by InfoSeek were followed. Lycos, Open Text, and WebCrawler follow, each having about 2.5% of their hits followed.

This data has two caveats. The first is that the relevant information for people may be the list of references itself. For example, people who wish to see how many links there are to their home page may search on their own name just to calculate this number. The second caveat is that these numbers may be skewed by the number of hits returned by each service. Thus, while InfoSeek has nearly 6% of its results followed, only a total of 13,045 references returned by InfoSeek were followed, compared with the 24913 references followed that were contributed by Lycos, which on average had only 2.5% of its 19 hits examined.

	% of Unique Hits Returned
Galaxy	99.6%
Yahoo	92.8%
Lycos	90.6%
WebCrawler	90.3%
Open Text	88.8%
InfoSeek	79.5%

Table 6: Unique References by Service

This table shows the percentage of references each service returned that were unique to that service.

	% of hits returned that are followed	Total Hits Followed
InfoSeek	5.89	13,045
Lycos	2.56	24,913
Open Text	2.51	4,025
WebCrawler	2.42	21,631
Yahoo	1.33	7,503
Galaxy	0.83	12,022

Table 7: Relevance of Returned Hits by Service

This table shows the percentage of followed hits for each service. References returned by multiple services are counted multiple times. Column 2 shows the actual number of references followed for each service. These numbers are out of 50,878 queries, except Open Text which is out of 19,951 queries.

	Ave. Time (sec)	% Timed Out
WebCrawler	9.64	2.30
InfoSeek	12.30	3.01
Open Text	16.26	14.13
Yahoo	18.32	2.28
Lycos	18.99	4.87
Galaxy	19.52	3.10

Table 8: Performance of Services

This table shows the average time in seconds taken by each service to fulfill a query. The second column gives the percent of time that the service would time out, or fail to return any hits under one minute.

3.2.3 Performance

Finally, we measure each service's response time. Table 8 summarizes our findings. It is not surprising, although disappointing, to find that average times vary from just under 10 seconds to just under 20. The percent of time the services timed out is under 5% for all services except Open Text, which is the newest and presumably still going through some growing pains. One explanation for the length of times taken by these services is that the majority of requests are during peak hours. Thus, results are naturally skewed towards the times when the services are most loaded. Times during non-peak hours are much lower.

4 Related Work

Unifying several databases under one interface is far from novel. Many companies, such as PLS[21], Lexis-Nexis[14], and Verity[30] have invested several years and substantial capital creating systems that can handle and integrate heterogeneous databases. Likewise, with the emergence of many Internet search services, there have been many different efforts to create single interfaces to the sundry databases. Perhaps the most widely distributed is CUSI[18], the Configurable Unified Search Index, which is a large form which allows users to select one service at a time and query that service. There are also several other services much like CUSI, such as the All-in-One Search Page[2], or W3 Search Engine list[19]. Unfortunately, while the user has many services on these lists to choose from, there is no parallelism or collation. The user must submit queries to each service individually, although this task is made easier by having form interfaces to the various services on one page.

The Harvest system[6] has many similarities to the MetaCrawler; however, rather than using existing databases as they are and post-processing the information returned, Harvest uses "Gatherers" to index information and "Brokers" to provide different interfaces to extract this information. However, while Harvest may have many different interfaces to many different internal services, it is still a search service like

Lycos and WebCrawler, instead of a meta-service like MetaCrawler.

There are also other parallel Web search services. Sun Microsystems supports a very primitive service[27], and IBM has recently announced infoMarket[11] which, rather than integrating similar services with different coverage, integrates quite different services, such as DejaNews[26], a USENET news search service, McKinley[29], a clone of Yahoo with some editorial ratings on various pages, in addition to Open Text and Yahoo.

The closest work to the MetaCrawler is SavvySearch[3], an independently created multi-threaded search service released in May 1995. SavvySearch has a larger repertoire of search services, although some are not WWW resource services, such as Roget's Thesaurus. SavvySearch's main focus is categorizing users' queries, and sending them to the most appropriate subset of its known services[4].

Like the MetaCrawler, the Internet Softbot[8] is a meta-service that leverages existing services and collates their results. The Softbot enables a human user to state what he or she wants accomplished. The Softbot attempts to disambiguate the request and to dynamically determine how and where to satisfy it, utilizing a wide range of Internet services. Unlike the MetaCrawler, which focuses on indices and keyword queries, the Softbot accesses structured services such as Netfind and databases such as Inspec. The Softbot explicitly represents the semantics of the services, enabling it to chain together multiple services in response to a user request. The Softbot utilizes automatic planning technology to dynamically generate the appropriate sequence of service accesses. While the MetaCrawler and the Softbot rely on radically different technologies, the vision driving both systems is the same. Both seek to provide an expressive and integrated interface to Internet services.

5 Future Work

We are investigating how the MetaCrawler will scale to use new services. Of particular importance is how to collate results returned from different types of Internet indices, such as USENET news and Web pages. Also important is determining useful methods for interacting with specialized databases, such as the Internet Movie Database[28]. If the information requested is obviously located on some special purpose databases, than it does not make sense to query each and every service. We are investigating methods that will enable the MetaCrawler to determine which services will return relevant data based solely on the query text and other data provided by the user.

5.1 Future Design

The existing MetaCrawler prototype can cause a substantial network load when it attempts to verify a large number of pages. While one query by itself is no problem, multiple queries occurring at the same time can cause the system and network to bog down. However, with the emergence of universally portable Internet-friendly languages, such as Java[10] or Magic Cap[25], load problems can be lessened by having users' machines take on the workload needed to perform their individual query, as discussed in Section 2.2. The JavaCrawler, a prototype next generation MetaCrawler written in Java, supports most of the features already present in the MetaCrawler. However, instead of users running queries on one central service, each user has a local copy of the JavaCrawler and uses that copy to directly send queries to services. The load caused by verification will be taken by the user's machine, rather than the central server. This has the added benefit of inserting downloaded pages into the local cache, making retrieval of those pages nearly instantaneous. The JavaCrawler is loaded automatically from the MetaCrawler home page when visited with a Java-compatible browser.

5.2 Impact on Search Service Providers

We anticipate that a wide range of meta-services like the MetaCrawler will emerge over the next few years. However, it is uncertain what the relationship between these meta-services and search service providers will be. We envision that this relationship will hinge on what form the "information economy" used by service providers takes. We discuss two different models.

5.2.1 Charge-per-Access

In the charge-per-access model, service providers benefit from any access to their database. InfoSeek has already taken this model with their commercial service. InfoSeek is financially rewarded regardless of who or what sends a query to their commercial database. Many other databases, both on and off the Web, also use this model.

The MetaCrawler fits in well with this model. Since service providers benefit from any access, the added exposure generated by the MetaCrawler is to their advantage. Further, this model creates an implicit sanity check on the claims this paper makes on the use of its features. In order for the MetaCrawler, or any meta-service, to survive in such an economy, it must charge more per transaction than the underlying services, as the MetaCrawler will in turn have to pay each service for its information. Thus, users must be willing to pay the premium for the service. By voting with their pocketbook, they can determine if those features are truly desirable.

5.2.2 Advertising

In the advertising model, service providers benefit from sponsors who in turn gain benefit from exposure provided by the service. Nearly all major search services that do not charge users directly have adopted this model, as have many other unrelated services which are heavily accessed.

Under this model, the providers' relationship with the MetaCrawler can become problematic as the MetaCrawler filters away superfluous information such as advertisements. One promising method to ensure profitable co-existence is to use provider-created interfaces. Providers could create an interface for the MetaCrawler to access their service which, in addition to returning relevant hits, also returns the appropriate advertisement. Another solution involves the MetaCrawler accepting advertisements, and forming a profit-sharing relationship with the service providers. We are currently investigating these and other methods of mutually beneficial co-existence with service providers.

6 Conclusions

In this paper we have presented the MetaCrawler, a meta-service for Web searching with additional features designed to return more references of higher quality than standard search services. We demonstrated that users follow references reported by a variety of different search services, confirming that a single service is not sufficient (Table 2). Further, due to the expressive power of the MetaCrawler's interface, the MetaCrawler was able to automatically determine that up to 75% of the hits returned can be discarded. Finally, the performance benchmarks and usage logs also show that the features provided by the MetaCrawler are both reasonably fast and actually used in practice.

The MetaCrawler provides a "Consumer Reports" of sorts for Web searchers. The individual service data extracted from the MetaCrawler's logs is compelling evidence concerning the quality of each service. By comparing services using the same query text and recording what links users follow, we are able to evaluate the services from a user's point of view. As far as we know, we are the first to quantitatively compare the search services used by MetaCrawler on a large sample of authentic user queries.

While it is possible that some MetaCrawler features could be integrated into the search services, others are intrinsic to meta-services. By definition, only a meta-service can provide the coverage gained by using multiple services. Also, as argued earlier, client-side meta-services can offer user and site customizations, and absorb the load caused by post-processing of search results. Finally, there are some features that do not belong under control of search services for purely pragmatic reasons. For example, as more commercial search services become available, tools will emerge that select which services to use on the basis of cost. An impartial meta-service such as the MetaCrawler avoids the conflict of interest that would arise if such a tool were offered by one of the commercial services.

New Web services are constantly being created. As the number and variety of services grows, it is natural to group existing services under one umbrella. The MetaCrawler goes further than merely organizing services by creating an integrated *meta-service* that moves the interface (and the associated computational load) closer to the user. We believe that this trend of moving up the information "food chain" will continue. The MetaCrawler is one of the first popular meta-services, but many more will follow.

7 Acknowledgments

The research presented in this paper could not have been accomplished without the help of many individuals. We would like to thank Mary Kaye Rodgers, for editing assistance and for putting up with late nights. Ruth Etzioni and Ellen Spertus provided comments on an earlier draft. Dan Weld, Rich Segal, Keith Golden, George Forman, and Donald Chinn were very vocal and active in testing the early prototypes of the MetaCrawler, and Craig Horman and Nancy Johnson Burr were extremely helpful and patient in dealing with it when it ran amok. Lara Lewis was very helpful in finding references upon demand. The Internet Softbot group provided early insight into desirable features of the MetaCrawler, and Brian Bershad and Hank Levy contributed ideas relating to the impact the MetaCrawler could have on the Web. Ken Waln aided in early development for his form patches to the WWW C library, and Lou Montulli helped in later development by unlocking the secrets of `nph`-scripts and Netscape caching. MetaCrawler development was supported by gifts from US West and Rockwell International Palo Alto Research. Etzioni's Softbot research is supported by Office of Naval Research grant 92-J-1946 and by National Science Foundation grant IRI-9357772.

8 About the Authors

Erik Selberg, *selberg@cs.washington.edu*, http://www.cs.washington.edu/homes/selberg
Department of Computer Science and Engineering
Box 352350
University of Washington
Seattle, WA 98195

Erik Selberg is pursuing his Ph.D. in computer science at the University of Washington. His primary research area involves World Wide Web search, although he also has interests regarding system performance and security as well as multi-agent coordination and planning. In April, 1995 he created the MetaCrawler, a parallel Web search meta-service. He graduated from Carnegie Mellon University in 1993 with a double major in computer science and logic, and received the first Allen Newell Award for Excellence in Undergraduate Research.

Oren Etzioni, *etzioni@cs.washington.edu*, http://www.cs.washington.edu/homes/etzioni
Department of Computer Science and Engineering
Box 352350
University of Washington
Seattle, WA 98195

Oren Etzioni received his bachelor's degree in computer science from Harvard University in June 1986, and his Ph.D. from Carnegie Mellon University in January 1991. He joined the University of Washington as assistant professor of computer science and engineering in February 1991. In the fall of 1991, he launched the Internet Softbots project. In 1993, Etzioni received an NSF Young Investigator Award. In 1995, Etzioni was chosen as one of 5 finalists in the Discover Awards for Technological Innovation in Computer Software for his work on Internet Softbots.

His research interests include: software agents, machine learning, and human-computer interaction.

References

[1] InfoSeek Corporation. InfoSeek Home Page.
URL: `http://www.infoseek.com`.

[2] William Cross. All-In-One Internet Search Page.
URL: `http://www.albany.net/~wcross/all1srch.html`.

[3] Daniel Dreilinger. Savvy Search Home Page.
URL: `http://www.cs.colostate.edu/~dreiling/smartform.html`.

[4] Daniel Dreilinger. Integrating Heterogeneous WWW Search Engines.
URL: `ftp://132.239.54.5/savvy/report.ps.gz`, May 1995.

[5] EINet. Galaxy Home Page.
URL: `http://galaxy.einet.net/galaxy.html`.

[6] C. Mic Bowman et al. Harvest: A Scalable, Customizable Discovery and Access System. Technical Report CU-CS-732-94, Department of Computer Science, University of Colorado, Boulder, Colorado, March 1995.
URL: `http://harvest.cs.colorado.edu/harvest/papers.html`.

[7] Michael Schwartz et al. WWW Home Pages Harvest Broker.
URL: `http://town.hall.org/Harvest/brokers/www-home-pages/`.

[8] O. Etzioni and D. Weld. A softbot-based interface to the internet. *CACM*, 37(7):72–76, July 1994.
URL: `http://www.cs.washington.edu/research/softbots`.

[9] David Filo and Jerry Yang. Yahoo Home Page.
URL: `http://www.yahoo.com`.

[10] James Gosling and Henry McGilton. The Java Language Environment: A White Paper.
URL: `http://java.sun.com/whitePaper/javawhitepaper_1.html`.

[11] IBM, Inc. infoMarket Search Home Page.
URL: `http://www.infomkt.ibm.com`.

[12] IBM, Inc. Query By Image Content Home Page.
URL: `http://wwwqbic.almaden.ibm.com/~qbic/qbic.html`.

[13] Martijn Koster. Robots in the Web: threat or treat? *ConneXions*, 9(4), April 1995.

[14] LEXIS-NEXIS. LEXIS-NEXIS Communication Center.
URL: `http://www.lexis-nexis.com`.

[15] Michael Mauldin. Lycos Home Page.
URL: `http://lycos.cs.cmu.edu`.

[16] Michael L. Mauldin and John R. R. Leavitt. Web Agent Related Research at the Center for Machine Translation. In *Proceedings of SIGNIDR V*, McLean, Virginia, August 1994.

[17] Max Metral. Helpful Online Music Recommendation Service.
URL: `http://rg.media.mit.edu/ringo/ringo.html`.

[18] Nexor. CUSI (Configurable Universal Search Interface).
URL: `http://pubweb.nexor.co.uk/public/cusi/cusi.html`.

[19] University of Geneva. W3 Search Engines.
URL: `http://cuiwww.unige.ch/meta-index.html`.

[20] Open Text, Inc. Open Text Web Index Home Page.
URL: `http://www.opentext.com:8080/omw/f-omw.html`.

[21] Personal Library Software, Inc. Personal Library Software Home Page.
URL: `http://www.pls.com`.

[22] Brian Pinkerton. WebCrawler Home Page.
URL: `http://webcrawler.com`.

[23] Brian Pinkerton. Finding What People Want: Experiences with the WebCrawler. In *Proceedings of the Second World Wide Web Conference '94: Mosaic and the Web*, Chicago IL USA, October 1993.

[24] Brandon Plewe. The Virtual Tourist Home Page.
URL: `http://wings.buffalo.edu/world`.

[25] Daniel Sears. Guide to CodeWarrior Magic/MPW. Development Release 1
URL: `http://www.genmagic.com/MagicCapDocs/CodeWarriorMagic/introduction.html`, May 1995.

[26] DejaNews Research Service. DejaNews Home Page.
URL: `http://www.dejanews.com`.

[27] Sun Microsystems, Inc. Multithreaded Query Page.
URL: `http://www.sun.com/cgi-bin/show?search/mtquery/index.body`.

[28] The Internet Movie Database Team. The Internet Movie Database.
URL: `http://www.msstate.edu`.

[29] The McKinley Group, Inc. Magellan: McKinley's Internet Directory.
URL: `http://www.mckinley.com`.

[30] Verity, Inc. Verity Home Page.
URL: `http://www.verity.com`.