A World Wide Web Meta Search Engine
Using an Automatic Query Routing Algorithm

Submitted to Committee Members

Dr. Wen-Chen Hu
Dr. Kai Chang
Dr. David Umphress

By
Ruslana Svidzinska

In Partial Fulfillment of the
Requirements for the
Degree of
Master of Computer Science and Software Engineering

May 12th, 2001
# TABLE OF CONTENTS

1. INTRODUCTION................................................................. 6

2. LITERATURE REVIEW.......................................................... 9
   2.1 Overview of conventional search techniques...................... 9
   2.2 Conventional query routing systems................................. 11
       2.2.1 Manual query routing services................................. 11
       2.2.2 Automated query routing systems based on centroids...... 12
       2.2.3 Automated query routing systems without centroids...... 12

3. SYSTEM STRUCTURE.......................................................... 14
   3.1 System overview............................................................ 14
   3.2 Off-line operations....................................................... 17
       3.2.1 Neighborhood-based topic identification.................... 18
       3.2.2 Search engine selection index database...................... 20
   3.3 Major tasks performed on-line........................................ 21
       3.3.1 Query expansion...................................................... 21
       3.3.2 Clustering............................................................... 21
       3.3.3 Search engine ranking.............................................. 22

4. MAJOR OFF-LINE TECHNOLOGIES USED IN THIS PROJECT ......... 23
   4.1 Neighborhood-based topic identification.......................... 23
       4.1.1 The algorithm........................................................ 23
4.1.2 Examples ................................................................. 25
4.2 Search engine selection index database ......................... 33
  4.2.1 The algorithm ....................................................... 33
  4.2.2 Examples ............................................................... 35

5. MAJOR ON-LINE TECHNOLOGIES USED IN THIS PROJECT .......... 40
  5.1 Query expansion and Clustering ................................. 40
    5.1.1 The query expansion algorithm .......................... 41
    5.1.2 The clustering algorithm .................................. 44
    5.1.3 Examples of query expansion and document clustering .... 46
  5.2 Search engine ranking .............................................. 51
    5.2.1 The algorithm ................................................... 51
    5.2.2 Examples .......................................................... 53

6. EXPERIMENTAL RESULTS ............................................. 55
  6.1 System interface .................................................... 55
  6.2 Experiments ........................................................... 58

7. CONCLUSIONS AND FUTURE WORK ............................... 63

8. REFERENCES ............................................................... 64
List of Figures

**Figure 1.** System architecture........................................................................................................ 15

**Figure 2.** Programming languages and methods used in this project........................................... 16

**Figure 3.** Using CGI to connect the Web Client and Server.......................................................... 17

**Figure 4.** Terms collected by the back-link method......................................................................... 20

**Figure 5.** Query results for “link:www.vortex.com”....................................................................... 26

**Figure 6.** HTTP addresses extracted from the HTML page of Figure 5........................................ 27

**Figure 7.** Information obtained from the downloaded back link sources of Figure 6................. 28

**Figure 8.** List of distinct words in the set of documents in Figure 7.............................................. 29

**Figure 9.** Terms collected by the Neighborhood-based identification algorithm........................ 29

**Figure 10.** Topic words of the selected topic-specific search engines.......................................... 30-31

**Figure 11.** Comparison analysis between WebPage Theme Inspector and the proposed system....................................................................................................................... 32

**Figure 12.** Connection of the Web interface and database......................................................... 34

**Figure 13.** JDBC architecture........................................................................................................ 35

**Figure 14.** Title and search format attributes for some selected specialized search engines........... 36
Figure 15. Topic words for the specialized search engine http://www.vortex.com/...... 38

Figure 16. Query expansion procedure................................................................. 41

Figure 17. Search results for the query “python”.................................................. 47

Figure 18. Rating results for the documents in Figure 17..................................... 48

Figure 19. Rating results....................................................................................... 49

Figure 20. Rating results....................................................................................... 49

Figure 21. Results of the query expansion algorithm......................................... 51

Figure 22. Results of Search Engine Ranking Algorithm.................................... 53-54

Figure 23. Web system interface.......................................................................... 55

Figure 24. Routing results for query “python”................................................. 56

Figure 25. CNET.com: Shareware from Go to of Figure 24............................... 57

Figure 26. Search results of “python programming” on CNET.com: Shareware from Search of Figure 24................................................................. 58

Figure 27. Routing results for the query “palm os”.......................................... 59

Figure 28. Routing results for the query “auburn”.......................................... 61

Figure 29. Routing results for the query “safari”............................................. 62
1. Introduction

Searching the Internet has never been an easy task, even though it is one of the most common tasks performed on the Web. This task is becoming even more difficult with the continued growth of the amount of information posted on the World Wide Web. Not only is there an enormous amount of information, but also the lack of structure makes the search for relevant information a difficult and sometimes very time-consuming procedure. Traditional Web search engines return the user query results by displaying a long list of documents without any process of data classification or clustering. Considering the variations among search engines, the efficient integration of the results from several different search engines for the same query is an important but difficult technique that can dramatically improve Web search technology.

Search engines, such as Yahoo! [24] and Alta Vista [2], are the most widely used tools for finding information on the WWW. However, these general-purpose search engines are subject to low precision and/or low coverage. Manually generated directories such as Yahoo! provide high-quality references, but cannot keep up with the Web's explosive growth. Although crawler-based search engines such as AltaVista cover a larger fraction of the Web, their automatic indexing mechanisms often cause search results to be imprecise. It is thus difficult for a single search engine to offer both high coverage and high precision. This problem is exacerbated by the growth in Web size and by the increasing number of naive users of the Web who typically issue short, often single word queries to search engines.
On the other hand, the recent growth in both the number and variety of specialized topic-specific search engines, from VacationSpot.com [19] to KidsHealth.org [12] or epicurious.com [5], suggests a possible approach to this problem: searching the topic-specific search engines. Topic-specific search engines often return higher-quality references than broad, general-purpose search engines for several reasons. First, specialized engines are often a front-end to a database of authoritative information that search engine spiders, which index the Web's HTML pages, cannot access. Second, specialized search engines often reflect the efforts of organizations, communities, or individual enthusiasts that are committed to providing and updating high-quality information. Third, because of their narrow focus and smaller size, word-sense ambiguities and other linguistic obstacles to a high-precision search are ameliorated.

The proposed system takes a different approach to the search process. As a metasearch engine, it uses other search engines (in our case topic-specific search engines) to obtain results relevant to the user query. After a user query is submitted, the system first tries to find some additional information on the user query itself and proposes to the user different areas of knowledge, where his/her query terms are found, and also different search engines, which would facilitate further exploration of the particular topic. In addition, the proposed system maintains a small database to help it decide which topic-specific engine is most likely to be able to deal with the user query in relation to the chosen topic.

The proposed system not only incorporates all the advantages of the topic-specific search engines mentioned above, but also addresses the issue of finding the best search engine for the query. This helps to overcome a major disadvantage of traditional search
systems: the uniform treatment of query results. It significantly minimizes the human effort needed in the search procedure.

Two techniques are presented in this project: (i) neighborhood-based topic identification and (ii) query expansion. The first technique is used to specify which topic-specific engine supports which “topic” (actually set of topics). The second one helps the user to narrow down the query to a more specific search and suggests which of the specific-search engines are most likely to contain information on the subject. The proposed system obtains all the necessary information from the Web, which makes it a somewhat intelligent agent in the sense that it autonomously learns from the Web what it does not know.

The report is about a prototype metasearch engine with a built-in automated query routing mechanism and it is divided into four major sections: (i) a brief description of the existing search facilities, (ii) an overview of the proposed system, (iii) a detailed analysis of the algorithms used to implement the system, and (iv) the system interface and experimental results obtained by the new system.
2. Literature Review

Because of the great demand, there are a lot of different search facilities represented on the WWW. This chapter provides an overview of existing search tools, with a brief analysis of their strengths and weakness.

2.1 Overview of conventional search techniques

Web search services have very different implementation methods and search strategies from each other. Over the years, search technology has improved significantly. Different search engines with different search depths, behavior, and speed have emerged. However, all the available search engines can be categorized in two basic architectures: (i) genuine search engines and (ii) directories [22]. The difference is in how the listings are compiled.

- Search engines, such as HotBot [9], create their listings automatically. Search engines crawl the Web gathering information, then users search through the database they create.

- A directory such as Yahoo [24] depends on humans for its listings. It is similar to newspaper-classified advertisements. The Web author submits a short description of the site to the directory, or editors write descriptions for sites they review. A search looks for matches only in the descriptions submitted.

The disadvantages of these two systems are obvious, including:
• Most directories are out of date, are full of dead links, have access only to a limited number of Web documents, and have to be constantly maintained and updated

• Search engines are usually slow and sometimes “biased,” because they rely heavily on the seed URL, not to mention uniform treatment of search results.

• Both directories and search engines usually result in hundreds or thousands of search results, most of which are completely irrelevant to the user query but are included in the set because of the appearance of the search keyword in either the Web document or in the description. This problem makes searches for common terms or one- or two-word searches extremely inefficient and annoying, discouraging the user from searching further.

In general, the “silver bullet” approach – one search engine fits all - does not work anymore, because of the enormous size and fast, dynamic changes on the World Wide Web. This project aimed to look at the search process from another perspective with this thought in mind: “Keep it small and keep it simple.”

The whole idea to somehow bring some structure to the search results is not new and has been under investigation for a long time. Topic-specific search engines keep their directories up to date most of the time, so it should be possible to use their search capabilities in order to provide users with a more efficient and precise search mechanism. The next section will discuss conventional query routing systems that have been studied, some of which have been successfully implemented and have already demonstrated their efficiency.
2.2 Conventional query routing systems

Conventional query routing systems and services, some of which are currently available on the Web, can be classified into three groups. Even though they use different kinds of techniques, they all serve the same purpose: to make search results more accurate and help users to find relevant information in less time and with less effort. All of the query routing systems are metasearch engines (i.e. they depend on other existing search engines to perform the searches).

2.2.1 Manual query routing services

Some query routing services have recently become available on the Web. However, each requires that some aspect of the query routing be performed manually by the user. AllSearchEngines.com [1], SEARCH.COM [16], InvisibleWeb.com [11], and The Search Broker [13] provide a categorized list of specialized search engines, but these services basically require the users themselves to choose engines from the list. Although they provide keyword search interfaces to find desired search engines, the terms that can be accepted as the keywords are limited to abstract category names (such as “sports”). The users are required to make connections between their specific queries (such as “Michael Jordan”) to the relevant categories. This approach only works if the user is actually quite sure what he or she has in mind (in other words, what this term means or which specific area he/she wants to look into), because a topic-specific search engine can actually be very narrow. If the user has directed the query to the wrong search engine, the chances are that he or she will get no relevant search results as the specialized search engine will not be able to address their query.
2.2.2 Automated query routing systems based on centroids

Some systems perform automated query routing and a centroid-based technique is widely used by these kinds of systems. This generates “centroids” (summaries) of databases, each of which typically consists of a complete list of terms in that database and their frequencies, and decides which databases are relevant to a user query by comparing the query with each centroid. CORI [4] is a centroid-based system. GLOSS [8] is also based on the same idea, although it does not explicitly generate the centroid. STARTS [7] and WHOIS++ [21] propose standard architectures and protocols for the distributed information retrieval using centroids provided by information sources.

An advantage of the centroid-based technique is that it is able to handle a wide variety of search keywords by using the large number of terms obtained from databases. However, this technique cannot be applied to most of the topic-specific search engines provided on the Web because of the restricted access to their internal databases.

2.2.3 Automated query routing systems without centroids

There are some automated query routing systems that do not generate centroids. However, these systems have strict limitations on acceptable search keywords. Query routing in Discover [17] relies on short texts, associated with WAIS databases, to explain the contents of databases given by service providers. Discover can operate only when some of the search keywords are contained in the short texts. Although Discover helps users refine their queries so that it is able to select topic-specific search engines, this effort is inadequate for handling a wide variety of search keywords. ProFusion [10] routes queries in thirteen predefined categories to six search engines. It posts sample
queries from each category to the search engines and examines which engine is best for that category by checking the relevance of the returned documents. ProFusion has a dictionary to determine which categories the given user queries are relevant to. Since, however, this dictionary is created by looking at the names of newsgroups (a term “movie” can be categorized into a recreation category from “rec.movie.reviews”), the result is that ProFusion can accept only limited types of queries.

One exceptional system that cannot be classified into any of these three groups is Ask Jeeves [3], which performs automated routing of queries to a limited set of Web sites that contain “answers” to users’ “questions.” Since Ask Jeeves is a proprietary commercial system, little is known about its internal routing algorithm, its degree of automation, or its ability to scale.
3. System Structure

The proposed system is a metasearch engine. Atsushi Sugiura and Oren Etzioni [18] proposed the specification for the following system in the article “Query routing for Web search engines”, where they describe the outline of the system and provide some experimental data with comparison analysis of the different methods.

3.1 System overview

The main purpose of the proposed system is to provide users with efficient and effective Web information retrieval. As an input it takes the user query (the set of words, the user would like to get information on), and as an output the user receives an expanded query (the original query plus three proposed topics, which the user can explore) and suggested topic specific search engines, where relevant information is most likely to be located. The mechanism which is involved is often addressed as a query routing problem, which helps to resolve ambiguities when the user query is too broad (for instance, the word “python” can be related to entertainment, programming or animals), and not only narrow down the search, but also point the user to the most appropriate search engines, where he/she can explore the particular topic with some level of guarantee for success.

The system is a two-tier metasearch engine. The first tier involves collecting the required information about topic-specific search engines. The second uses a routing system to expand and route the user queries to a subset of the selected topic-specific search engines. An overall schema of the system is shown in Figure 1.
Time-consuming tasks are concentrated in the white blocks and lightweight on-line tools are shown in gray.

It is worth pointing out that even though off-line and on-line tasks are performed in different periods of time, they use similar techniques. The ranking algorithm, for instance, is used both off-line and on-line as a part of an automated query routing algorithm. The only difference is the amount of information that has been processed.

To implement such a system, several programming languages (Java, Perl, and Shell) and a DBMS (Oracle 8) are used. Figure 2 shows the components of the system and which programming language is used for each.
Figure 2. Programming languages and methods used in this project

Although not shown in the diagram, the documents downloaded from the Web and the interface provided to the user are written using HTML, which is recognized by all Web browsers.

Due to the College of Engineering network setup, implementation of the tasks which must be performed on-line use the Client-Server architecture schematically illustrated in Figure 3. Using this type of architecture allows us to build an on-line Web based prototype of the system, which can be accessed from any computer in the Department of Engineering.
Figure 3. Using CGI to connect the Web Client and Server

The main problem that arises with this design is the time-consuming nature of client–server communication, which makes the system slower than if the program were to reside on one computer.

In the next two sections, the system components are examined, starting with the off-line tasks and ending with the on-line tasks.

3.2 Off-line operations

Two tasks of the system are performed off-line: Neighborhood-based topic identification (collecting information about topic-specific search engines) and creation of the database with the information gathered. The first operation is used to retrieve relevant terms for each topic-specific engine and some criteria that will be used to identify appropriate topic-specific search engines. The second is simply the operation of populating a permanent database with the information obtained in the previous task.
It is worth mentioning that performing those tasks is done off line because of the enormous amount of time they require for execution; on average it takes up to one hour to run them both. Even though adding a new topic-specific search engine is not yet fully automated, the time needed for this procedure can be up to two hours in the worst case.

3.2.1 Neighborhood-based topic identification

The main purpose of this operation is to identify a group of terms that describe the particular topic-specific search engine. The algorithm will be explained later in details; this section provides comparison analysis between different methods and approaches.

The idea of exploiting the “reputation” of a Web page has attracted research attention recently and has even been incorporated into some search engines. Recent work analyzing the link structure of the Web suggests that hyperlinks between pages often represent relevance. An important key point in this research is to use Neighborhood-based identification of the search engines' topics. The Neighborhood-based method does not collect terms relevant to the search engine's topics from their internal databases, but collects them from the limited “neighborhood” of Web pages. Therefore, only a small number of abstract terms (some of them representing the topics of a search engine) can be obtained.

There are three methods of Neighborhood-based topic identification to collect terms relevant to a search engine from existing and static Web documents.
o **The front-page method**

Every search engine has a page providing a query interface (we call this page a *front page*), and the front page usually contains terms explaining the topic of that search engine. In the front-page method, all terms in the front page and their frequencies are registered to a search engine selection index.

---

o **The back-link method**

Web pages that have links pointing to a search engine's front page (we call these pages *back-link pages*) often contain good explanations of that search engine. The back-link method first finds multiple back-link pages for a target search engine $e_i$, then extracts from all the back-link pages only the terms that are in the lines of the links to $e_i$. Finally, it stores all the extracted terms and their document frequencies in the search engine selection index.

---

o **The database sampling method**

A database sampling method obtains a part of the search engine’s database and generates a kind of incomplete centroid. That is, it submits training queries to the search engine and stores in the search engine selection index all the terms in the returned documents and frequencies of those terms. A similar method has been proposed by Xu and Callan[23].

Due to the nature of the sources used for the term collection, most of the terms collected by the front-page and back-link methods are abstract general terms like “hotel” and “travel.” Relatively few specific terms, such as the proper noun “Hilton,” are
obtained. In contrast, the database sampling method is quite successful at collecting specific terms. In the proposed system, the back-link method was implemented and will be described in more detail in Chapter 4.

### 3.2.2 Search engine selection index database

The purpose of the *Search engine selection index database* is to store ranked terms and their normalized frequencies with their relevant search engines. We call high-frequency terms, which appear in the search engine selection index, *topic terms*. Specifically, a set of topic terms \( T_{OPIC_i} \) for the search engine \( e_i \) is defined as follows:

\[
T_{OPIC_i} = \{ w_{ij} | f_{ij} > f_{max} \times 0.8 \}
\]

where \( w_{ij} (1 \leq j \leq m) \) is a term in the index for \( e_i \), \( f_{ij} \) is its frequency, and \( f_{max} \) is the highest frequency observed in the index.

Figure 4 shows examples of terms obtained by the back-link method. The abstract general terms would usually be topic terms.

<table>
<thead>
<tr>
<th>Computer security search engine (<a href="http://www.securityfocus.com">www.securityfocus.com</a>)</th>
<th>Hotel search engine (hotel search under <a href="http://www.travelweb.com">www.travelweb.com</a>)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Topic term</strong></td>
<td><strong>Topic term</strong></td>
</tr>
<tr>
<td>security</td>
<td>hotel</td>
</tr>
<tr>
<td>bug</td>
<td>travel</td>
</tr>
<tr>
<td>bugtraq</td>
<td>airline</td>
</tr>
<tr>
<td>unix</td>
<td>reservation</td>
</tr>
<tr>
<td>exploit</td>
<td>restaurants</td>
</tr>
<tr>
<td>attack</td>
<td>map</td>
</tr>
<tr>
<td>password</td>
<td>book</td>
</tr>
<tr>
<td>mail</td>
<td>flights</td>
</tr>
</tbody>
</table>

**Figure 4.** Terms collected by the back-link method.
3.3 Major tasks performed on-line

Major on-line tasks for this project are: (i) receiving the user query, (ii) expanding the original query, (iii) identifying the relevant topic-specific engines, (iv) sending the expanded user query to the topic-specific search engine, and (v) displaying the results. The main characteristics of the tasks performed on-line are that they need to be lightweight and highly optimized, so the system will give a response to the user query in a reasonable time.

3.3.1 Query expansion

Query expansion is a mechanism that transforms the original query into an expanded query by adding words which would suggest to the user different subjects or areas for further search. In other words, the purpose of the query expansion mechanism is to provide the user with a set of words, which would help formulate a more specific and narrow query. Usually, the query expansion mechanism provides the user with several alternatives and is very helpful, especially for inexperienced users.

An important benefit of the query expansion process implemented in the proposed system is the ability to automatically obtain terms relevant to a query from the Web. This allows the system to identify topics covering great variety of queries without having to maintain a massive dictionary or database of terms in a wide range of fields.

3.3.2 Clustering

Clustering is a process grouping similar or related terms into so-called “clusters.” In the proposed system, clustering is performed as a part of the query expansion process. There are some key requirements for document clustering [26]:
1. **Coherent Clusters**: The clustering algorithm groups similar documents together. As documents often have multiple topics, they should not be confined to a single cluster; the clustering algorithm generates *overlapping clusters* when appropriate.

2. **Efficiently Browsable**: The user needs to determine at a glance whether the contents of a cluster are of interest. Therefore, the system has to provide concise and accurate cluster descriptions.

3. **Speed**: The clustering system should not introduce a substantial delay before displaying the results. This suggests two additional requirements. *Algorithmic Speed*: The clustering algorithm used must be able to cluster hundreds or even thousands of snippets in a few seconds. *Snippet-Tolerance*: As there is no time to download the original documents off the web, the system should produce high quality clusters even when it only has access to the snippets returned by the search engines.

### 3.3.3 Search engine ranking

The final operation of the proposed system is the ranking of the topic-specific search engines. This procedure determines which search engines are most likely to have relevant results for the expanded user query. In this process, the system performs a calculation of the *goodness*\(^1\) of each topic-specific search engine for any given query by comparing the terms obtained through query expansion with terms stored in the search engine selection index.

---

\(^1\) Goodness is a metric that indicates relevancy of the particular search engine to the query.
4. Major Off-Line Technologies Used in this Project

Off-line tasks performed by the proposed system provide the essential information and are the “heart” of the system. The accuracy and precision of the functional metasearch engine depend solely on the accuracy and precision of the topic identification algorithm and the resulting topic-specific search engine selection index database. In this section, detailed algorithms are provided for both the Neighborhood-based topic identification algorithm and the search engine selection index database.

4.1 Neighborhood-based topic identification

The method applied for the Neighborhood-based identification of search engines’ topics is the so-called back-link method. This method is more complex and time consuming than the alternatives, but has shown excellent results [18].

4.1.1 The algorithm

The algorithm works as follows:

1. Web pages that have links to a search engine’s front page (the page that provides the query interface to the search engine) are identified.

2. Terms that are in the line where the link to the search engine occurred are then extracted and ranked.

3. The results obtained in step 2 of the current list are normalized and stored along with their frequency in the database.
Back links are obtained through sending a query²

\[\text{link:www.searchengineURL.com}\]

to the general-purpose search engine “Google” [6] (instead of “searchengineURL” we insert the actual URL of the search engine that we are interested in).

However, a problem arises when the number of search results that are returned from Google exceeds several hundred; due to the limitations of time and space, there is no way to process this number of full-text Web documents. The easiest solution is to somehow limit the number of URLs that are taken into consideration. Experimental results show [18], that after 50 Web back-linked documents, the number of distinct terms with frequencies exceeding a certain threshold significantly drops. This justifies reducing the number of back-links to 75 (we just take the first 75 from the top of the query results list), leaving a small margin to assure the accuracy of the ranking results.

One very important step after Web documents have been downloaded is to make sure that we consider only terms that co-occur with our search engine URL and are also in the same paragraph as a hyperlink. By doing so, we eliminate the possibility of irrelevant terms creeping into our “characteristics list.”

For some highly commercial search engines, this approach does not work simply because their links appear as banners without any context. This situation requires special attention and is a good topic for future research.

² Perl has a facility for retrieving Web content known as LWP, for Light Weight Processes. This particular method was used to fetch Web documents. Most of the word processing also has been done in Perl.
The problems mentioned earlier are not the only problems that occur during ranking. Another problem is the so-called “stop words” problem. In any natural language (and English is no exception) some words occur more frequently than others. Words like “in,” “and,” “an,” or “about” do not carry much meaning by themselves, but in combination with other words they make language more smooth and understandable, acting as a glue which binds all the other, less frequent (and most of the time more meaningful) words together. Unfortunately, these words usually have the highest frequencies in Web documents. To eliminate the presence of these words in our database, we subtract the set of predefined stop words from the set of distinct words, extracted from the back-link pages. This is done before the terms are actually ranked, to improve the performance of the algorithm.

After the stop words have been removed from the list of terms obtained from the back link pages, a calculation of ranks is performed.

4.1.2 Examples

To illustrate the proposed algorithm we would like to give a short example to show how it actually works. To reduce the size of the documents provided, let us calculate the rank of the topic-specific search engine http://www.vortex.com/, based on five back link pages. The small number of back link pages is given for demonstration purposes only.

- **Step 1.** Using Google [6], obtain HTTP addresses of the Web pages that contain links to http://www.vortex.com/. First we download the Google search results page for the query: “link:www.vortex.com.” The results page is shown in Figure 5 in HTML format and has been edited to save space.
The HTTP addresses, which are shown in bold in Figure 5, will be extracted from the downloaded HTML page. The results are shown in Figure 6.
http://www.epic.org/privacy/privacy_resources_faq.html
http://www.csl.sri.com/neumann/neumann.html
http://www.pfir.org/
http://www.cpsr.org/cpsr/privacy/epic/privacy_resources_faq.html

**Figure 6.** HTTP addresses extracted from the HTML page of Figure 5

- **Step 2.** Download Web pages using the HTTP addresses from Step 1 (because in this step the algorithm produces a large amount of information, the result of it is not shown)

- **Step 3.** Extract paragraphs (portions of text between `<p>` and `</p>` tags) from the documents obtained in Step 2, which contain hyperlinks to http://www.vortex.com/. The results of this step are shown in Figure 7.
In a related effort that is supported in part by the ACM Committee on Computers and Public Policy, Lauren Weinstein moderates the Privacy Forum Digest. He is providing a superb service for those of you who are deeply concerned about privacy issues. You may subscribe or request information via privacy-request@vortex.com. Check out the Privacy Forum on-line.

Peter is the chairman of the ACM (Association for Computing Machinery) Committee on Computers and Public Policy, and the creator and moderator of the Internet RISKS Forum. Lauren is a member of that same committee, and he is the creator and moderator of the Internet PRIVACY Forum. PRIMING Forum.

Movies, Television, Privacy, Computer History, Improbable Research, and more hit the Web as Vortex Technology's WWW server opens for public access. Internet PRIVACY Forum digest information and archive materials are available for those interested in privacy issues. Film reviews, news about film and television releases, publicity photos, audio/video clips, broadcast and cable TV industry info and much more can be found in "Professor Neon's TV & Movie Mania." Learn about the history of computing from the Computer History Association of California. And visit the renowned motley crew of scientists, doctors, Nobel laureates and a host of others, all of whom formerly populated the pages of "The Journal of Irreproducible Results," over in "The Annals of Improbable Research."

Figure 7. Information obtained from the downloaded back link sources of Figure 6

- **Step 4.** Remove all HTML tags and punctuation marks.
- **Step 5.** Identify a list of the distinct words in the set of documents shown in Figure 7.
• **Step 6.** Remove the stop words from the list of words obtained in Step 5. The result of this procedure is shown in Figure 8.

professor, access, acm, annals, archive, association, audio/video, available, back, broadcast, cable, california, chairman, check, clips, committee, computer, computers, computing, concerned, creator, crew, deeply, digest, doctors, effort, email, film, formerly, forum, history, hit, host, http, improbable, industry, interested, irreproducible, issues, journal, laureates, lauren, learn, machinery, mania, materials, member, moderated, moderates, moderator, motley, movie, movies, neon’s, nobel, on-line, opens, others, out, over, part, partially, peter, photos, policy, populated, privacy, providing, public, publicity, related, releases, renowned, request, research, results, reviews, risks, scientists, server, sponsored, subscribe, superb, supported, technology's, television, tv, via, visit, weinstein

**Figure 8.** List of distinct words in the set of documents in Figure 7

• **Step 7.** Count the number of times the words from the list of distinct words shown in Figure 8 occur. The normalized frequency is the “rate” of this particular word. Figure 9 shows the results of the Neighborhood-based topic identification algorithm for the top ten words.

<table>
<thead>
<tr>
<th>Topic term</th>
<th>Frequency</th>
<th>Normalized Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>privacy</td>
<td>12</td>
<td>1.00</td>
</tr>
<tr>
<td>forum</td>
<td>7</td>
<td>0.58</td>
</tr>
<tr>
<td>acm</td>
<td>4</td>
<td>0.33</td>
</tr>
<tr>
<td>issues</td>
<td>4</td>
<td>0.33</td>
</tr>
<tr>
<td>lauren</td>
<td>4</td>
<td>0.33</td>
</tr>
<tr>
<td>moderator</td>
<td>4</td>
<td>0.33</td>
</tr>
<tr>
<td>digest</td>
<td>4</td>
<td>0.33</td>
</tr>
<tr>
<td>committee</td>
<td>3</td>
<td>0.25</td>
</tr>
<tr>
<td>history</td>
<td>3</td>
<td>0.25</td>
</tr>
<tr>
<td>public</td>
<td>3</td>
<td>0.25</td>
</tr>
</tbody>
</table>

**Figure 9.** Terms collected by the Neighborhood-based identification algorithm
Even though only five back links have been used in the illustration, it is obvious that the specialized search engine http://www.vortex.com/ specializes in privacy issues.

Figure 10 shows the “topic” words for some of the topic-specific search engines included in our study.

<table>
<thead>
<tr>
<th>Root address (HTTP)</th>
<th>Terms with high frequencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>ciac.llnl.gov</td>
<td>security, vulnerability, team, ciac, computer, ftp, e-mail, access, solaris, membership, available, incident, secure, bulletins, type, tools, virus, first, machine, program, sgi, hp-ux, buffer, response, ast, system, vulnerabilities, http, server.</td>
</tr>
<tr>
<td>hyperarchive.lcs.mit.edu</td>
<td>mac, program, filemaker, source, coding, unix, image, archive, system, code, info-mac, windows, utility, demo, faq, subject, hyperarchive, available, compression, plug-in, macintosh, files, software.</td>
</tr>
<tr>
<td>shareware.cnet.com</td>
<td>software, document, medical, browser, server, engines, science, current, australia, global, frames, time, engineering, health, content, university, documents, frameset, multimedia, table, national, form, image, world, law, browsers.</td>
</tr>
<tr>
<td>shopper.cnet.com</td>
<td>mac, hardware, auctions, shopper, winzip, newsletters, software, shopping, netscape, platform, related, dispatch, popular, pcs, business, join, javascript, shareware, linux, products, prices, jobs, icq, tech, help, games.</td>
</tr>
<tr>
<td><a href="http://www.adobe.com">www.adobe.com</a></td>
<td>adobe, windows, product, reader, software, document, products, names, acrobat, pdf</td>
</tr>
</tbody>
</table>

**Figure 10.** Topic words of the selected topic-specific search engines
<table>
<thead>
<tr>
<th>URL</th>
<th>Topic Words</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.apple.com">www.apple.com</a></td>
<td>mac, software, license, us, power, g4, pro, system, tips, server, subscreen, video, options, reserved, notice, rights, privacy, product, macintosh, copyright, airport, contact, products, inc, computer, quicktime, apple, os</td>
</tr>
<tr>
<td><a href="http://www.cs.indiana.edu">www.cs.indiana.edu</a></td>
<td>university, one-line, technology, computing, message, languages, college, lab, laboratory, institute, automated, intelligence, group, indiana, robotics, artificial, cognitive, systems, psychology, research, hyplan, subscribe, programming, computer, science</td>
</tr>
<tr>
<td><a href="http://www.cwru.edu">www.cwru.edu</a></td>
<td>university, muse, institute, technology, school, server, system, health, medical, research, college, usa</td>
</tr>
<tr>
<td><a href="http://www.ibm.com">www.ibm.com</a></td>
<td>ibm, global, worldwide, management, systems, e-business, microsoft, corporation, technology, solution, computer, companies, enterprise, technologies, privacy, company, e-commerce, business, inc, products, server, research, software, solutions</td>
</tr>
<tr>
<td><a href="http://www.vortex.com">www.vortex.com</a></td>
<td>privacy, computer, risks, system, software, security, us, problems, systems, technology, forum, issues, computers, report, problem, time, phone, power, e-mail, y2k, people.</td>
</tr>
</tbody>
</table>

**Figure 10.** Topic words of the selected topic-specific search engines (cont.)

To justify these results, we ran the same root addresses through another similar system, WebPage Theme Inspector [20], which has been implemented by PHD Software Systems and uses all of the available backlinks provided by several search engines. Instead of using a full-text search, they study title, keyword section, and the first paragraph of the backlink and sometimes include the description. Figure 11 shows us a
A clear picture of the similarities and differences between our system and the “Theme Inspector.”

<table>
<thead>
<tr>
<th>HTTP OF THE SEARCH ENGINE</th>
<th># OF WEBPAGES</th>
<th># OF TERMS</th>
<th>N2/N1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cached by the proposed system</td>
<td>collected by Theme Inspector</td>
<td>collected by the proposed system (N1)</td>
</tr>
<tr>
<td>shopper.cnet.com*</td>
<td>75</td>
<td>50973</td>
<td>26</td>
</tr>
<tr>
<td><a href="http://www.adobe.com">www.adobe.com</a></td>
<td>75</td>
<td>1064139</td>
<td>10</td>
</tr>
<tr>
<td><a href="http://www.cs.indiana.edu">www.cs.indiana.edu</a></td>
<td>75</td>
<td>38134</td>
<td>25</td>
</tr>
<tr>
<td><a href="http://www.cwru.edu">www.cwru.edu</a></td>
<td>75</td>
<td>22255</td>
<td>12</td>
</tr>
<tr>
<td>travel.excite.com</td>
<td>75</td>
<td>16865</td>
<td>26</td>
</tr>
<tr>
<td><a href="http://www.findtravelto.com">www.findtravelto.com</a></td>
<td>75</td>
<td>27</td>
<td>26</td>
</tr>
<tr>
<td><a href="http://www.ibm.com">www.ibm.com</a>*</td>
<td>75</td>
<td>4043853</td>
<td>23</td>
</tr>
<tr>
<td>travel.state.gov</td>
<td>75</td>
<td>53201</td>
<td>29</td>
</tr>
<tr>
<td><a href="http://www.vortex.com">www.vortex.com</a>*</td>
<td>75</td>
<td>2216</td>
<td>21</td>
</tr>
<tr>
<td>hyperarchive.lcs.mit.edu</td>
<td>75</td>
<td>16</td>
<td>23</td>
</tr>
<tr>
<td><a href="http://www.apple.com">www.apple.com</a></td>
<td>75</td>
<td>342055</td>
<td>28</td>
</tr>
<tr>
<td>ciac.llnl.gov*</td>
<td>75</td>
<td>67</td>
<td>29</td>
</tr>
<tr>
<td><a href="http://www.railserve.com">www.railserve.com</a></td>
<td>75</td>
<td>2766</td>
<td>29</td>
</tr>
<tr>
<td>shareware.cnet.com</td>
<td>75</td>
<td>4428</td>
<td>26</td>
</tr>
</tbody>
</table>

Figure 11. Comparison analysis between WebPage Theme Inspector and the proposed system

The first column contains the search engine addresses that were entered in Google (a Neighborhood-based topic identification algorithm) and WebPage Theme Inspector. A star beside the search engine address indicates that the first term picked by the prototype system and the Theme Inspector are the same. The second and third column show how
many links were explored by each system. The fourth and fifth columns show how many “topic” terms were identified. The sixth column contains the number of common terms and the last column contains the ratio between the number of common terms and the number of terms picked by the system.

There are numerous reasons why the ratio of similar terms varies so much. These include:

- Differences in the list of the “stop words”
- Differences in indexing and ranking methods
- The use of different search engines to obtain back links

These results do not prove that one system is better than the other, but make an interesting comparison.

4.2 Search engine selection index database

The purpose of the Search engine selection index database is to store ranked terms and their normalized frequencies. This information is provided by a Neighborhood-based topic identification algorithm.

4.2.1 The algorithm

The database stores the characteristics of the search engines. It is a permanent storage medium and is not affected by on-line tasks, but can only be modified off-line. An important question would be how many of the terms extracted from the back-link pages should be stored in the database? It is obvious that we cannot limit ourselves to just topic
terms, because there are usually no more than one or two of them, but there should be a reasonable number in order to have a database of reasonable size. For this project the following approach was taken: first, elements are sorted by their normalized frequencies; second, the frequency of the 30th element of the sorted list is noted; and finally, only elements with this frequency or above are included in the database.

The database was implemented using an Oracle 8 database system. Once a table has been created with an SQL statement

```
create table table_name (field_name field_type) as object;
```

it can be used for permanent storage of information. In order to insert information into the database, the statement `Insert` is used:

```
insert into table_name(field1_name, field2_name, ...) values
    (value1, value2, ...);
```

After the database has been created and populated with records, it can be accessed and searched for relevant information. Because information retrieval from the database takes place on-line, we had to establish a connection between the Web interface and the database. A conceptual representation of this chain is shown in Figure 12.

**Figure 12.** Connection of the Web interface and database
The connection between DBMS and the Web was established using CGI and JDBC (Java Database Connectivity). The architecture of Java Database Connectivity itself is illustrated in Figure 13.

![JDBC Architecture Diagram](image)

**Figure 13.** JDBC architecture

### 4.2.2 Examples

Information about the topic-specific search engines is saved into two tables to reduce space complexity. The first table contains information like HTTP address, title, and search format. This table is created using the following SQL statement:

```sql
create table prob (address varchar2(50) unique, title varchar2(70), searchstring varchar2(250));
```
Figure 14 shows the actual entries for the selected topic-specific search engines used in our study.

<table>
<thead>
<tr>
<th>HTTP Address</th>
<th>Title</th>
<th>Search Format</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.filmbug.com">www.filmbug.com</a></td>
<td>Filmbug Movie and Movie Star Search</td>
<td><a href="http://www.filmbug.com/f/index.php3?query=">http://www.filmbug.com/f/index.php3?query=</a>&lt;search&gt;&amp;type=m&amp;x=20&amp;y=8</td>
</tr>
<tr>
<td>travel.excite.com</td>
<td>Excite Travel</td>
<td><a href="http://travel.excite.com/results/?keywords=">http://travel.excite.com/results/?keywords=</a>&lt;search&gt;</td>
</tr>
<tr>
<td>travel.state.gov</td>
<td>US State Department - Services - Consular Affairs</td>
<td><a href="http://travel.state.gov/cgi-bin/search.swish.pl?query=">http://travel.state.gov/cgi-bin/search.swish.pl?query=</a>&lt;search&gt;&amp;results=0</td>
</tr>
</tbody>
</table>

**Figure 14.** Title and search format attributes for some selected specialized search engines

The purpose of this table is to provide all the essential information about the search engine. One of the most important parts is Search Format, which allows us to automatically search each individual search engine. Expression “<search>” is embedded into each record and has to be substituted with query words delimited by “+.” Therefore, to search for “python programming” using the topic-specific search engine provided by Indiana University, the query will look like:


and can be embedded into the results interface as a link, using `<A HREF=..> </A>` HTML tags.
The second table contains the HTTP address, topic words and normalized frequency for each search engine. HTTP addresses in this table corresponds to the HTTP addresses in the first table. The second table is created using the following SQL statement:

```
create table enginfo (address varchar2(50), dword varchar2(70),
    rate real);
```

This table is populated automatically after the results of the topic identification algorithm are obtained. Figure 15 illustrates the table entries for the specialized search engine http://www.vortex.com/.
Figure 15. Topic words for the specialized search engine http://www.vortex.com/

To improve efficiency and save space, a database view has been used to combine the information from both tables. To create a view that would hold the data from both tables, SQL statement have been used:
create table combination as select prob.address, title, searchstring, dword, rate from prob, enginfo where prob.address=enginfo.address;

This View does not require additional disk space because it is a soft link to already existing data, but can be queried as a static table in order to retrieve all the information about specialized search engines. Population of the database with consistent data is the final off-line operation, after which we can proceed with on-line tasks.
5. Major On-line Technologies Used in this Project

Several algorithms are performed on-line in the proposed system. Those algorithms are: (i) the query expansion algorithm, (ii) the clustering algorithm, and (iii) the search engine ranking algorithm. On-line tasks can be categorized as lightweight and highly optimized. While off-line operations run on the full-text Web pages, on-line operations try to download as little information as possible, attempting to retain high precision and relevancy of the results.

5.1 Query expansion and clustering

The two most important on-line operations, the query expansion and clustering algorithms, work together to expand the original query and provide several alternatives in order to help the user in further exploration of the topic. The clustering algorithm is a part of the query expansion algorithm. These algorithms will be explained in detail separately, but an illustration of the clustering algorithm will be combined with an illustration of the query expansion algorithm because they function cooperatively.

An important benefit of the query expansion process is the ability to automatically obtain terms relevant to a query from the Web. This allows the system to identify topics in many kinds of queries without having to maintain a massive dictionary or database of terms covering a wide range of fields. In this project, we assume the user query consists of one or two words, even though that is not necessary. If the query gets more specific, then a smaller number of relevant topics can be identified. Based on the assumption
mentioned above, one of the first and most important steps in the routing process is to expand the user query.

5.1.1 The query expansion algorithm

The outline of the mechanism involved in the query expansion is shown in Figure 16.

1. Get a document set \( D_0 \) relevant to a user query \( Q_0 \), where search keywords are \( w_{01}, ..., w_{0n} \), by sending \( Q_0 \) to a general search engine.
2. Count co-occurrences of search keywords and other terms in the document set \( D_0 \).
3. Let \( WH_0 \) and \( WL_0 \) be a set of terms whose co-occurrences exceed a certain threshold and a set of the other terms, respectively. \( WH_0 \) is considered relevant to the query \( Q_0 \) and will be a part of the query expansion result.
4. Pick up at most four topic terms \( wt_1-wt_4 \) from \( WL_0 \).
5. Formulate four queries \( QT_1-QT_4 \) by combining \( wt_1-wt_4 \) with \( Q_0 \) (for example, \( QT_1=\text{"w}_{01} ... \text{w}_{0n} \text{wt}_1\text{"} \)).
6. Cluster all terms in \( D_0 \) into at most three clusters: \( W_1=\{w_{11}, ..., w_{1m}\}, W_2=\{w_{21}, ..., w_{2k}\} \) and \( W_3=\{w_{31}, ..., w_{3j}\} \).
7. Formulate three queries \( Q_1-Q_3 \) by combining \( W_1-W_3 \) with \( Q_0 \) (for example, \( Q_1=\text{"w}_{01} ... \text{w}_{0n} \text{w}_{11} ... \text{w}_{1m}\text{"} \)).
8. Get document sets \( DT_1-DT_4 \) and \( D_1-D_3 \) by sending \( QT_1-QT_4 \) and \( Q_1-Q_3 \) independently to a general search engine.
9. Count co-occurrences in \( DT_1-DT_4 \) and \( D_1-D_3 \). Sets of high co-occurrence terms \( WTH_1-WTH_4 \) and \( WH_1-WH_3 \), as well as \( WH_0 \) in step 3, are query expansion results.

Figure 16. Query expansion procedure

The query expansion process consists of searching for the relevant terms by running three different algorithms on the sets of Web documents retrieved by sending the user query.
and the sub-expanded user query to the general-purpose search engine\(^3\). This procedure is fairly straightforward, and can be described as follows:

- **Getting relevant terms from the Web dynamically**

  As was mentioned earlier, the proposed system does not use special dictionaries for query expansion, but uses existing Web documents to search for relevant terms. The user query is submitted to the general-purpose search engine twice: the first time by itself and the second time paired with terms suspected to be relevant to the query. Using the Web as a source for relevant terms makes the system more dynamic and up-to-date, reflecting which topics are more popular at the present time.

- **Co-occurrence based evaluation of term relevance**

  The mutual relevance of terms is evaluated on the basis of their co-occurrence with the original search keywords in the documents. To make query routing more efficient, in our algorithm we do not follow the links provided by a general-purpose search engine, but explore the titles of the returned pages and short descriptions that are found in the search results. If we were to download full size documents, as we did in the off-line steps of our algorithm, it would be likely to increase the precision, but it would also increase the running time enormously.

  It is very important to count only the co-occurrence of the terms with the original query keywords, instead of just counting how many times the terms simply

---

\(^3\) The query expansion step of the algorithm uses MetaCrawler.com as its primary source of the relevant to user query terms.
“occur” in the Web document. As a metasearch engine, MetaCrawler [14] uses other search engines and directories to retrieve the results. It is well known that in some directory-based search engines, the top positions are just paid-for advertisements that can affect the results of ranking.

- **Using a pseudo-feedback technique**

  It is difficult to determine the term relevance from only the results of a single document search on the general search engine. The proposed system, therefore, re-evaluates low occurrence terms through sending those terms, combined with the original query keywords, to the general-purpose search engine for a second time, and performing the co-occurrence-based evaluation again. Such automatic query refinement is called pseudo-feedback.

In our implementation of the mechanism for counting co-occurrences of the terms and the original query, recognizing similar words (often different forms of the same word) is a very important step towards a more accurate and precise algorithm. The algorithm mentioned here targets two things: merging the plural form of nouns with the singular form and the simple past tense of the verbs with the simple present tense if such a form of the verb is present in the document. To accomplish this, the following approach was taken:

- Pick a term that belongs to the list of distinct terms prepared for rating ($w_i$) and check with every term that remains in the list
• If $w_j$ (some term from the list) starts with $w_i$ and has $\text{length}(w_j) - \text{length}(w_i) \leq 2$ then remove $w_j$ from the list and during the rating process combine the number of co-occurrences of $w_j$ and $w_i$, treating them all as co-occurrences of the term $w_i$

The other precaution was taken to make the system less biased and more accurate: Steps 1 and 8 in the algorithm shown in Figure 6 use different general-purpose search engines - MetaCrawler [14] and Google [6] respectively.

5.1.2 The clustering algorithm

Clustering, as shown in Figure 16, is a part of the query expansion process and takes place in Step 6. Clustering allows us to get additional relevant terms. It uses a simple “ad hoc” method to reduce total computation time.

The clustering algorithm of the proposed system generates three clusters that are mutually exclusive. The algorithm is as follows:

1. Pick up the term $w_{max_1}$ with the highest co-occurrence in the document set $D_0$ obtained in step 1 of the query expansion algorithm in Figure 5. Let $D_{01}$ be a set of documents containing $w_{max_1}$.

2. Pick up the highest co-occurrence term $w_{max_2}$ in a set of documents not containing $w_{max_1}$. Let $D_{02}$ be a set of documents containing $w_{max_2}$ and not containing $w_{max_1}$.

3. Let $D_{03}$ be a set of the documents that contain $w_{max_3}$, which is the term with the highest co-occurrence among the documents that do not contain $w_{max_2}$ and $w_{max_1}$. It is very tempting to accept as $D_{03}$ the rest of the documents which were
excluded in Step 2 and 3 of the clustering algorithm, but experiments shows that by doing it this way we often bring together unrelated terms, which makes the search in Step 9 of the query expansion algorithm impossible.

4. Terms that appear in $D_{01}$, $D_{02}$, and $D_{03}$ would be the clusters of terms $W_1$, $W_2$, and $W_3$ in step 6 of Figure 5, respectively.

After the expansion process, there are at most eight clusters of terms: $WH_0$ (Step 3), $WTH_1$-$WTH_4$, and $WH_1$-$WH_3$ (Step 9). Since, however, different clusters are often related to the same topic, the system merges them to eliminate the duplicates. Basically, it merges clusters that contain the same topic term. It also merges clusters that have many common terms.

Merging is an iterative process. First of all, the two closest clusters are determined and merged (related documents obtained in Step 9 are joined and the list of terms are obtained). Then counting of co-occurrences takes place in the new cluster, and the process of merging is repeated again until the intersection of any two clusters reaches a certain threshold (in our case 20%). First of all, smaller clusters are merged with the bigger clusters, which will later define main topics. After that, large clusters are checked for overlap and merged by joining the document sets obtained in Step 9.

Merging is a very time-consuming process and, therefore, to reduce complexity it is performed only one time, after all the clusters have been determined. In this work another approach to merging was used. Instead of the continuous iteration, “ad hoc” clustering is done for the second time after all the documents: $DT_1$-$DT_4$, $D_1$-$D_3$ and $H_0$ have been retrieved. This is beneficial for the program in two ways:
Better efficiency, as the iterative algorithm of complexity $O(n^2)$ was substituted by an algorithm with linear complexity $O(n)$

Stability of the results, as experiments show that quality of the first two clusters increased by 25%.

However, the major disadvantage of this method is that the third cluster is sometimes irrelevant or tightly bounded with one of the first two clusters, and this cannot be successfully detected unless we employ some kind of heuristic algorithm which would be able to handle both of these cases in a reasonable amount of time.

### 5.1.3 Examples of query expansion and document clustering

The query expansion algorithm is reasonably complex and an illustration of its operation would greatly aid in understanding it. The outline of the algorithm is schematically represented in Figure 16. In our example, the original query “python” will be expanded to “python programming” using the query expansion algorithm. All the examples are simplified for illustrative purpose. Assume the initial query is

$$Q_0=“python”$$

**Step 1.** Use a general-purpose search engine to get documents $D_0$ resulting from the original query $Q_0$. This step uses Perl to fetch result documents for the query “python.”

The query string is constructed using the following Perl statement:

```
$fgsearch="http://search.metacrawler.com/crawler?general=python&method=0&redirect=web&rpp=50&hpe=10&region=0&timeout=0&sort=0&format=beta99&theme=classic&refer=mc-search";
```
Then Web documents are retrieved:

```
use LWP::Simple;
$pageoutput = get($fgsearch);
```

As a result $D_0$ is assigned an HTML document. After all HTML tags are removed the result is shown in the Figure 54 (only ten results are shown).

**d_1:** Pair Networks - World Class Web Hosting pair Networks offers a complete Web development environment, with full CGI privileges and a true virtual domain, for only $29.95/month.

**d_2:** Monty Python at TVvideos.com Monty Python's Flying Circus and more than 100 other TV shows on video for secure online purchase at great prices in association with Amazon.com! Affiliate.

**d_3:** Monty Python Monty Python: Homepage of Monty Python Online, includes a tour, the spam club, plugs, new stuff, games, chat, and shopping. Click on this Internet Keyword to go directly to the Monty Python Web site.

**d_4:** Python Software Activity null

**d_5:** Boas and Pythons Get the specifics on various snakes in the Boid family. Find out average size, color and potential dangers.

**d_6:** Lycos Music: Monty Python's Flying Circus - Biography Biography - As any seventh grade dork can tell you, Monty Python's Flying Circus is inarguably the greatest Sketch Comedy group ever. With skits such as "The Argument Clinic" and "The Pet Shop" and films such as

**d_7:** Python Compared to Other Languages Many links to comparisons involving Python.

**d_8:** Python Language Website Home page for Python, an interpreted, interactive, object-oriented, extensible programming language. It provides an extraordinary combination of clarity and versatility, and is

**d_9:** manny juan's python scripts for cgi these are my first cgi scripts in python and i'm sharing them with the web community, especially with web authors who have calendars to maintain on their sites. i hope these are

**d_10:** Pet Pythons Includes resources and personal Web pages of interest to owners of pet python snakes.

**Figure 17.** Search results for the query “python”
Step 2. Count co-occurrences of the key words and other terms. In this process, the stop words are removed and frequency of all other terms is calculated. Because the original query appears in every single document, frequency is counted as the number of documents where each distinct term is present. Thus, if a document contains several identical terms they will be counted only once. Different forms of the same word are also merged at this point. The results of this process are shown in Figure 18. Only the words with co-occurrence rate over 1 are included.

<table>
<thead>
<tr>
<th>Topic term</th>
<th>Co-occurrence rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>monty</td>
<td>3</td>
</tr>
<tr>
<td>language</td>
<td>2</td>
</tr>
<tr>
<td>pet</td>
<td>2</td>
</tr>
<tr>
<td>circus</td>
<td>2</td>
</tr>
<tr>
<td>flying</td>
<td>2</td>
</tr>
<tr>
<td>cgi</td>
<td>2</td>
</tr>
<tr>
<td>snakes</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 18. Rating results for the documents in Figure 17

Step 3. Two sets of the terms $WH_0$ and $WL_0$ are chosen based on the number of co-occurrences. Let “monty” belong to $WH_0$ and all other terms to $WL_0$.

Step 4. Four terms: $wt_1$-$wt_4$ are picked up from $WL_0$. These terms are “language,” ”pet,” “circus,” and “flying.”

Step 5. Four supplemental queries are formulated:

\[ QT_1 = \{\text{python, language} \} \]
\[ QT_2 = \{\text{python, pet} \} \]
\[ QT_3 = \{\text{python, circus}\} \]

\[ QT_4 = \{\text{python, flying}\} \]

**Step 6. Clustering**

- As the first step of clustering, the term with the highest co-occurrence is picked up (in our case it is the word “monty”). So documents \( d_2, d_3, \) and \( d_6 \) (from Figure 17) should be assigned to the set of documents \( D_{01} \).

- Documents that do not belong to the set \( D_{01} \) are rated again and the results are as follows. Figure 19 shows only words with a co-occurrence of 2 or higher.

<table>
<thead>
<tr>
<th>Topic term</th>
<th>Co-occurrence rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>language</td>
<td>2</td>
</tr>
<tr>
<td>cgi</td>
<td>2</td>
</tr>
<tr>
<td>snakes</td>
<td>2</td>
</tr>
</tbody>
</table>

**Figure 19. Rating results**

So, \( w_{max}^2 \) is “language” and \( D_{02} = \{ d_7, d_8 \} \).

- Now rating for the rest of the documents takes place and results are shown in the Figure 20.

<table>
<thead>
<tr>
<th>Topic term</th>
<th>Co-occurrence rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>cgi</td>
<td>2</td>
</tr>
<tr>
<td>snakes</td>
<td>2</td>
</tr>
</tbody>
</table>

**Figure 20. Rating results**
According to the rating results $w_{max}$ is “cgi” and $D_{03} = \{d_1, d_2\}$.

- Three clusters are formulated from the $D_{01}$, $D_{02}$, and $D_{03}$:

$$W_1 = \{\text{monty, flying, circus}\}$$

$$W_2 = \{\text{language}\}$$

$$W_3 = \{\text{cgi}\}$$

**Step 7.** Three supplemental queries are formulated, using $W_1$, $W_2$, and $W_3$:

$$Q_1 = \{\text{python, monty, flying, circus}\}$$

$$Q_2 = \{\text{python, language}\}$$

$$Q_3 = \{\text{python, cgi}\}$$

**Step 8.** This step is the same as Step 1. The only difference is that document sets $DT_1$-$DT_4$ and $D_1$-$D_3$ are obtained by sending $QT_1$-$QT_4$ and $Q_1$-$Q_3$ independently to a general search engine.

**Step 9.** Repeats Step 2 and Step 6 on the larger set of documents obtained in Step 8. This is a concluding step of the query expansion algorithm, where three major topics are identified along with three independent sets of rated terms, which are passed to the final operation of the proposed system - ranking of the topic-specific search engines.

Figure 21 shows some of the experimental results of the Query Expansion Algorithm. All the terms are obtained by the system from the Internet without maintaining any kind of database for expanding the original query. Depending on the
dynamics of the WWW, these results are able to change with time, which allows us to use the system without any changes.

<table>
<thead>
<tr>
<th>Original Query</th>
<th>Query expansion results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>First cluster</td>
</tr>
<tr>
<td>python</td>
<td>monty</td>
</tr>
<tr>
<td>cd</td>
<td>music</td>
</tr>
<tr>
<td>organic</td>
<td>food</td>
</tr>
<tr>
<td>food</td>
<td>recipes</td>
</tr>
<tr>
<td>classic cars</td>
<td>parts</td>
</tr>
<tr>
<td>train</td>
<td>model</td>
</tr>
<tr>
<td>college sport</td>
<td>basketball</td>
</tr>
</tbody>
</table>

Figure 21. Results of the query expansion algorithm

5.2 Search engine ranking

The rank of topic-specific search engines allows the system to decide to which search engine the query will actually be routed.

5.2.1 The algorithm

The query expansion algorithm and the search engine selection index provide input data for the search engine ranking algorithm. A specific metric, the goodness, is introduced to
specify which of the search engines from the system are more likely to contain relevant information for the user.

The goodness of a search engine $e$ for a given set $W=\{w_1, w_2, \ldots\}$ of query expansion terms is calculated as follows:

$$goodness(e,W) = \sum_{w \in W} f_i \times c_i$$

where $c_i$ is the number of co-occurrences of $w_i$ counted in the query expansion process and $f_i$ is the frequency of term $w_i$ in the search engine selection index for $e$ ($f_i=0$ if there is no $w_i$ in the index). It is important to remember that not only topic terms but also non-topic terms are used in the calculation of goodness.

For convenience, a temporal supplemental table is created with ranking results for each cluster provided by the query expansion mechanism. This table contains identifier for the cluster along with a set of terms and corresponding frequencies of those terms. Because all the information about search engines is stored in the database it is very convenient to perform calculations of goodness using an SQL statement:

```sql
select clustnum, address, sum(tempdat.rate*combination.rate) as goodness, title, searchstring from combination, tempdat where combination.dword=tempdat.word group by clustnum, address order by clustnum, goodness desc;
```

where combination is a view that holds information about the specialized search engines and tempdat is a table which contain temporal information on the clusters.
After the goodness has been calculated, the proposed system generates a ranked list of search engines and selects the first three for each cluster as query routing results.

5.2.2 Examples

The calculation of goodness is a straightforward procedure. Figure 22 shows some experimental results for this operation.

<table>
<thead>
<tr>
<th>Original Query</th>
<th>Expanded Query</th>
<th>Topic-Specific search engine</th>
<th>Title</th>
<th>Goodness</th>
</tr>
</thead>
<tbody>
<tr>
<td>python</td>
<td>python monty</td>
<td><a href="http://www.filmbug.com/">http://www.filmbug.com/</a></td>
<td>Filmbug movie and movie star search</td>
<td>18.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://www.findtravelto.com/">http://www.findtravelto.com/</a></td>
<td>World travel search engine and travel directory - local tourism guide</td>
<td>10.4</td>
</tr>
<tr>
<td>python</td>
<td></td>
<td><a href="http://www.dustcloud.com/">http://www.dustcloud.com/</a></td>
<td>Dustcloud Media Search engine</td>
<td>8.4</td>
</tr>
<tr>
<td></td>
<td>python</td>
<td><a href="http://www.cs.indiana.edu/">http://www.cs.indiana.edu/</a></td>
<td>Indiana University computer science</td>
<td>18.6</td>
</tr>
<tr>
<td></td>
<td>programming</td>
<td><a href="http://shareware.cnet.com/">http://shareware.cnet.com/</a></td>
<td>CNET.com - Shareware</td>
<td>10.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://shopper.cnet.com/">http://shopper.cnet.com/</a></td>
<td>Latest prices – CNET.com</td>
<td>8.3</td>
</tr>
<tr>
<td>python</td>
<td>python language</td>
<td><a href="http://www.dustcloud.com/">http://www.dustcloud.com/</a></td>
<td>Dustcloud media search engine</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://www.apple.com/">http://www.apple.com/</a></td>
<td>Apple – Site Map</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://www.cs.indiana.edu/">http://www.cs.indiana.edu/</a></td>
<td>Indiana University Computer Science</td>
<td>0.2</td>
</tr>
<tr>
<td>linux</td>
<td>linux software</td>
<td><a href="http://shareware.cnet.com/">http://shareware.cnet.com/</a></td>
<td>CNET.com - Shareware</td>
<td>178.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://www.vortex.com/">http://www.vortex.com/</a></td>
<td>Vortex Technology</td>
<td>141.8</td>
</tr>
<tr>
<td>Query</td>
<td>Similarity Score</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>linux training</td>
<td>IBM Corporation-developerWorks Search 68.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>linux fast</td>
<td>Vortex Technology 36.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>linux fast</td>
<td>CIAC security Website 19.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>linux fast</td>
<td>Excite Travel 12.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>linux fast</td>
<td>CNET.com - Shareware 7.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>linux fast</td>
<td>Vortex Technology 3.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>linux fast</td>
<td>CIAC security Website 2.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>java bean</td>
<td>IBM Corporation-developerWorks Search 60.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>java bean</td>
<td>Vortex Technology 45.8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>java bean</td>
<td>Indiana University Computer Science 36.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>java bean</td>
<td>CNET.com - Shareware 27.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>java bean</td>
<td>Vortex Technology 26.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>java bean</td>
<td>Excite Travel 2.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>java bean</td>
<td>CNET.com - Shareware 2.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>java bean</td>
<td>Vortex Technology 1.0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 22.** Results of Search Engine Ranking Algorithm

Experiments show that the results should be ignored when “goodness” of the search engine is lower than a certain threshold value (less than ~3.0). As was mentioned earlier, the third cluster sometimes does not produce good results, because it is either rudimentary or there is not enough information to define it.
6. Experimental Results

This chapter shows the system interface and some experimental results.

6.1 System interface

The proposed system provides a simple keyword search web interface (shown in Figure 23). In this interface we have an input field for the user query and a button “Submit” to initiate the process of searching. The number of search keywords may vary and is not limited, but experiments show the system performs best using one or two word queries.

![Web system interface](image)

**Figure 23.** Web system interface

After a query has been submitted to the system, the result page appears after a short delay (which may vary depending on the Internet connection and stability of the network, but usually does not exceed 1-2 minutes). A result page from the query “python” is shown in Figure 24.
Figure 24. Routing results for query "python"

Three independent topics are provided to the user. Three topic-specific engines correspond to each suggested topic. Each search engine is listed with (i) goodness, (ii) search link, (iii) go to link, and (iv) title.
The Search link returns to users the search results of the topic-specific search engine with the expanded query. The Go to link takes the user to the interface of the particular engine, so the user can submit a query by him/herself.

Figures 25 and 26 shows the results of exploring “CNET.com – Shareware” by using Search and Go to links from Figure 24 (the original query: “python” and the expanded query: “python programming”).

![CNET.com: Shareware from Go to of Figure 24](image)

**Figure 25.** CNET.com: Shareware from Go to of Figure 24
Figure 26. Search results of “python programming” on CNET: Shareware from Search of Figure 24

6.2 Experiments

The first experiment shown in Figures 27 illustrates the query routing results for the original query “palm os”. Three terms related to this query topics were identified: “software,” “emulator,” and “handheld”. All of the topic-specific search engines that expanded queries are being routed to belong to the Computer section, which justifies the
accuracy of the query routing algorithm in this case. The last topic “handheld” can be ignored, due to the small \textit{goodness} factor.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{Figure27.png}
\caption{Routing results for the query “palm os”}
\end{figure}
In the second experiment the original query was “auburn”. The results of the query routing results are shown in Figure 28. Three topics were identified by the system: “university,” “football,” and “lodging”. These topic terms reflect the availability of the information on the Web related to the original broad query. This example well illustrates the performance of the query expansion algorithm of the system, because Auburn is well known for its University and College Football team.

The routing results for this query would be much better if the system included specialized search engines on education and sport. This problem has only occurred because the implemented system is only a prototype, which covers only selected topics, and can easily be solved by adding some more topic-specific search engines.
Query Routing Results

Query: auburn

university

[174.88039] Search / Go to RailServe - The Internet Railroad Directory
[172.74402] Search / Go to Case Western Reserve University
[147.82353] Search / Go to Indiana University Computer Science

football

[17.943218] Search / Go to Case Western Reserve University
[16.777105] Search / Go to RailServe - The Internet Railroad Directory
[13.436166] Search / Go to Excite Travel

lodging

[24.490198] Search / Go to World Travel search engine and travel directory - local tourism guide
[14.920214] Search / Go to Excite Travel
[11.788498] Search / Go to US State Department - Services - Consular Affairs

Figure 28. Routing results for the query “auburn”

Very interesting results are shown in Figure 29. In the last experiment, for the original query “safari”, query routing results: “africa,” “adventure,” and “game” are related. In
addition, topic-specific search engines for the first two clusters are the same. This illustrates the behavior of the system when the initial query is already too specific. Providing a mechanism to detect and merge similar clusters can prevent this kind of behavior.

**Figure 29.** Routing results for the query “safari”
7. Conclusions and future work

The industry of web searching continues to grow along with the rapidly growing WWW. The proposed Query Routing Algorithm makes the search easier by providing some sort of guidance for inexperienced users and for users of the Internet in general. This project has achieved two goals: (1) retrieving relevant terms directly from the Internet and (2) keeping the database of the topic-specific engines small but meaningful. Expanding the user query allows the user to retrieve more relevant information, because adding more words to the original query narrows down the search results, making selection of appropriate results much easier and more efficient.

Experimental results show that the proposed system can pick up two clusters relevant to the original query 80% of the time and select the relevant topic-specific search engine for each expanded query 90% of the time. In almost every section of this report, we have tried to describe all the possible features and tasks that can be added to this system to make it more precise. The results shown are more than satisfactory and prove that the system can be used in real life situations.

One of the disadvantages of the proposed system is a deterioration factor, which is related to the specialized search engines. Even though they averagely well maintained, some of the factors, like HTTP addresses, are hard coded into the system and so to make the system functional we have to keep track of any changes.

For future work, some features like adding a new topic specific search engine can be automated. Also, features such as the “phrase generator” which was described in [18] would be a very valuable addition to the system.
8. References


