

AN ENHANCED WEB SEARCH SCHEME

By

KERRY R. FLEMING

Bachelor of Science

Tougaloo College

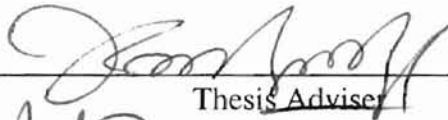
Tougaloo, Mississippi

1997

Submitted to the Faculty of the  
Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
MASTER OF SCIENCE  
December, 2000

AN ENHANCED WEB SEARCH SCHEME

Thesis Approved:

  
Thesis Adviser





  
Dean of the Graduate College

## ACKNOWLEDGMENT

I would first like to thank God, because none of this would be possible without him. I sincerely thank my advisor, Dr. K.M George, for his time, help, support, leadership, and guidance. Without his encouragement and help, the completion of this work would have been impossible. I would also like to thank both of my committee members, Dr. A. Burrell and Dr. G.E. Hedrick, for all their help.

I appreciate the professors who gave me support and guidance throughout the program. I also wish to express my sincere gratitude to the professors of Tougaloo College who supported and provided me with the necessary background education.

I would like to give special thanks to my parents, Noah and Dorothy Coffee, for their support and leadership. I also appreciate the support from my three brothers during difficult times, and lastly, I would like to thank my friends, especially Rolanda Gustavis, for their help, support, confidence, love, and encouragement during these years of study.

## TABLE OF CONTENTS

Chapter	Page
1. INTRODUCTION .....	1
1.1 Overview .....	2
1.2 Thesis .....	3
1.3 Organization .....	4
2. LITERATURE REVIEW .....	5
2.1 Search Schemes .....	6
2.2 Other Text Retrieval Systems .....	11
2.3 Intermediaries .....	13
3. DESIGN MODEL .....	17
4. WEB-BASE INTERMEDIARY IMPLEMENTATION .....	19
4.1 Data Generator .....	20
4.2 Data Monitoring .....	22
4.3 Data Editing .....	25
4.4 Performance and Security Issues .....	37
5. CONCLUSION .....	40
5.1 Summary .....	40
5.2 Future Work .....	40
BIBLIOGRAPHY .....	41
APPENDIX I .....	43

## LIST OF TABLES

Table	Page
I. Point distribution .....	35
II. Filtering scheme cycle .....	38

## LIST OF FIGURES

Figure	Page
2.1. Simple user interface .....	9
2.2. Advance user interface .....	10
2.3. Intermediaries locations .....	14
3.1. Scheme model .....	17
4.1. Information cycle .....	19
4.2. Scheme Interface .....	20
4.3. A segment of the Data Generator .....	21
4.4. Execution output of the Data Generator .....	22
4.5. FSA diagram for the above regular expression .....	23
4.6. Data Monitoring interface with page numbers .....	25
4.7. Search criterion .....	26
4.8. Interface for the Initial Query Reformulation scheme.....	28
4.9. Submitting queries using CGI scripts .....	28
4.10. Interface for the Second Query Analyzer .....	29
4.11. Selecting blocks in the scheme .....	30
4.12. Integrated Query option .....	31
4.13. Related Word section .....	32
4.14. Persistent Word Search option .....	33
4.15. Boolean section .....	33

Figure	Page
4.16. Example of the ordering diagram.....	34
4.17. Optimal score equation .....	34
4.18. Actual score equation .....	36
4.19. Setting page ranking .....	36
4.20. Final list snapshot .....	37

## CHAPTER 1

### INTRODUCTION

As a result of the rapid growth of the web, finding specific information on a certain topic by navigating from one link to another seems impossible. In order to speed up the process, many users resort to search engines hoping to find what they want. According to [14,17,18], search engines are web tools that use sophisticated search schemes to search the web or a company's database. With these schemes, search engines can gather information from the user and produce vital information for the user. However, there have been many complaints with search engines about the vast amount of web pages they produce and the relevance of each page. For novice users, filtering through an enormous list of web pages trying to find a relevant page is tedious and time consuming. However, many developers of search engines realize these problems, and try to give the user improved schemes such as Simple Search, Advance Search, Meta Search, or a series of pull down menus. If understood and used correctly, all schemes are helpful, but for most users, there is not enough information to help guide them through the technical writing and the puzzling pull down menus [3,15].

In this thesis, implementation of an enhanced search scheme is presented to solve the problem of manually filtering through a list of relevant and irrelevant results to find relevant pages. With the use of basic text retrieval features and intermediaries, the enhanced search scheme will help users reduce specified resulting list from search engines and find relevant information.



## 1.1 Overview

As the number of users, businesses, and schools that are introduced to the Web increases, the complexity of finding information also increases. Search engines provide the necessary tools for finding information on the web; however, in their attempt to assist users, search engines possess a certain level of complexity. According to Schwartz [14], the complexity of search engines lies in their variety, content, resource strategies, and the tools they deploy to assist users. For most users, the concern deals with the search results displayed by a given search engine. Search results are the set of pages containing a set of URLs (Uniform Resource Locator) that matched the users' query strategically. URLs are addresses for resources on the Internet. However, many novice users could care less about sophisticated search tools or strategies used by search engine [14].

The problem with search engines is the expansive amount of search results that contain a mixture of relevant and irrelevant URLs. Users use search engines to avoid the frustration of manually surfing through the web, but it seems that many search engines create the same frustration with their list of search results.

Users do not want to spend countless hours filtering through an enormous list of results in order to find relevant information on a particular subject. Users want to use a web search tool that has simple features, high performance, and a user-friendly environment [3, 15]. Not all search engines on the web can provide these features, but the enhanced search scheme presented in this paper can.

## 1.2 Thesis

In this thesis, we present the design and implementation of a scheme that enhances the functionality of search engines. The scheme described in this thesis is an enhanced search scheme using a web-based intermediary.

This scheme provides the necessary functions which users' desire when searching for information on the web. With the use of a web-based intermediary, this scheme increases user interaction and search engines' productivity. Ordinary search engines establish limited interactions for users. For example, a user submits a search query to a search engine, and the search engine processes the query and displays the results. After the submission of the search query, users' interactions are limited. At this point users can only interact with links returned by the search engine. Since the interactions are limited, users can not inquire on how the links displayed are relevant to the users' subject. Nor can they inquire on adding more supporting words to strengthen their search. In addition to their limited user interactions, there are other problems that hinder search engine performance such as the vast amount of links they produce and their cumbersome set of relevant and irrelevant links. With these problems, users may have to spend several hours filtering through enormous data trying to find relevant information about a specific subject; however, for most users, this process is frustrating and time consuming [3, 15]. Users want simple to operate web search tools that give them what they want. The scheme implemented in this thesis addresses these concerns by providing a user-friendly interface and a productive user interaction.

This scheme picks up where other search engines end by providing the necessary filtering and searching schemes that extract relevant links from ordinary search engine

results. This scheme reduces the time users would spend manually filtering through an enormous amounts of links, and the time it would take to find relevant links that matches the users' subject.

### **1.3 Organization**

The remainder of this thesis is divided into the following chapters:

- Chapter 2: A literature review of the basic search schemes, text retrieval systems, and intermediaries.
- Chapter 3: An illustration of the enhanced search scheme framework.
- Chapter 4: A discussion of the implementation and functionality of the web-based intermediary.
- Chapter 5: A summary of the thesis and suggestions for future work.
- Appendix I: A glossary of terms and definitions.

## CHAPTER 2

### LITERATURE REVIEW

In order to correct or to improve some of the problems with search engines, one must understand the development stages of web search tools.

The development of web search tools became apparent when the world was introduced to the Internet in the early 1980's [15]. Scholars, businesses, and consumers used every available new feature the Internet had in order to communicate and exchange files. Features such as telnet, ftp, newsgroup, gophers, and WAIS, became the suggested tools to use for finding information. As substantial users began to use and experiment with these features, more helpful features were spawned to aid in finding information. For example, files available via anonymous FTP could be found using a system called Archie. Online directories, such as HYTELNET, pointed to libraries and other collections that could be obtained with Telnet. Gophers introduced the ability of keyword search on text files using menu lines through their systems called Veronica and Jughead. Even though there were a variety of features to use, they all had their restrictions. For instance Gophers--at the time--were very popular, but could only retrieve information from one site or other gopher sites [14].

As the Internet advanced, so did the tools for information retrieval. By 1991, Brewster Kahle, at Thinking Machines, Inc., developed WAIS. WAIS (Wide Area Information Server) was designed using thirty years of research in the field of information science that included statistical text for retrieval. In addition to the design, was the Z39.50 protocol for interoperability between multitype automated library cataloging [18].

At this point, public WAIS sites seem to be a step above the other tools. These public sites provided a collection of directories that could be searched by anyone using WAIS. Further more, WAIS presented a list of files ranked primarily on the basis of search term occurrence [14]. However, the system was restricted when trying to gather information from other servers that didn't comply to their specific format [18]. By 1991, CERN released the first WWW (World Wide Web) line mode browsers. At this time, servers were limited and knowledge of markup languages was still in an experimental stage, CERN web site began to exploit resource discovery. An interesting feature of the CERN Web site included an alphabetized subject listing of links to pages that formed the World Wide Web Virtual library [14]. By the next few years, numerous HTTP (Hypertext Transfer Protocol) resources emerged making it hard for one Web site to keep track of new resources. With this increase in resources, a new stage emerged with the development of a new information retrieval tool called search engines. Search engines are programs that search for keywords in documents found on the web or in a database that have pre-selected their documents from various locations [18].

By 1996, search engines were featured in many journals, newspapers, and network television; however, the schemes used in search engines' implementation hindered many users with issues about usability [3].

## **2.1 Search Schemes**

The services search engines provide are greatly needed, but how do they go about helping user find information on the web?

Before discussing the different schemes, the term "search engine" must be clarified. The term is often used interchangeably to describe search engines using databases and search engines using directories. They are not the same, and the difference between the two is based on how their listings are compiled [17].

Database search engines gather information from their databases. These databases are filled with listing--web pages--sent in by authors wanting exposure, or by web crawlers that roam the Internet storing links and information about each page [14,18]. Two examples of database search engines are AltaVista, which has the largest database on the web [19], and HotBot [18]. When a search is processed, the search engine checks all the pages acquired in its database allowing users a variety of pages to choose from--depending on the length of the database. Also, since the crawlers continuously keep traversing through the web, newer pages are added to the database, and many old pages are updated.

Directories which are sometime known as categories, are constructed by humans. They receive a short description of each page that is submitted by the author, and the directory manager determines which directory to store each page. When a search is done on the directories, the search engine looks for matches only in the description submitted [18].

Whether a user utilizes databases or directory structures of search engines, the search engine scheme dictates how well a search engine will perform a suitable search. The next section covers some of the search schemes used by major search engines, and how some older models might have helped influence the schemes used by many search engines today.

According to Hahn (1999), on-line systems have basic features that are grouped in the following categories: search capabilities, browse capabilities, and other miscellaneous help function [3]. Many of today's search engines incorporate some of these basic features in addition to other techniques.

Search capabilities work in one of two ways. Either they help specify the relationship between terms in a search statement, or they help to facilitate the interpretation of a particular word [3]. Most search engines incorporate into their search scheme many of the features that defines search capabilities such as: boolean operations (AND, OR, NOT), proximity operation (NEAR), natural language query, data ranging, search term weighing, stop words, case sensitivity, and more.

The next category is browsing capabilities. They help users to determine which pages in the search result lists are of interest [3]. The features that encompass these capabilities are ranking and relevance feedback, zoning which is a process of displaying key portion of each page retrieved, highlighting which displays the words that match the search term, and a feature that ask "How many pages should be displayed? [3]". Of all of the browsing capability features, users rely on the relevance and ranking feedback to be precise and accurate. As mentioned before, search engines can produce a vast amount of search results. Embedded in this list, each page is given a certain measure of quality to let the user know how relevant the page is to the user. This measurement is called a relevant score. The scores may reflect the number of times a search term appears in the title, the beginning of a document, and/or if all the search terms are near each other [18]. Some search engines give the users some flexibility on controlling the relevance of a page by changing the weight (or importance) of each search word. Along with the relevance

score, there is the ranking feedback. The ranking feature structures each page in alphabetical order using the title [18], or in descending order using the relevance score. In addition, both methods are designed to help users find information quickly and easily.

The last category deals with miscellaneous capabilities. These capabilities center around various features like iterative searches, stored search queries, vocabulary browsing, delivery of full source documents, and a choice of simple or advance user interfaces [3]. The most popular features used in the miscellaneous category are the choice of simple or advanced user interfaces.



Figure 2.1. Simple user interface



The simple user interface shown in figure 2.1, is designed for novice users. It makes it easy for users to just type in their query and submit it without the worries of how it is interpreted. However, if a user wants to put restrictions on how their search query should search, then the advanced user interface is more appropriate.

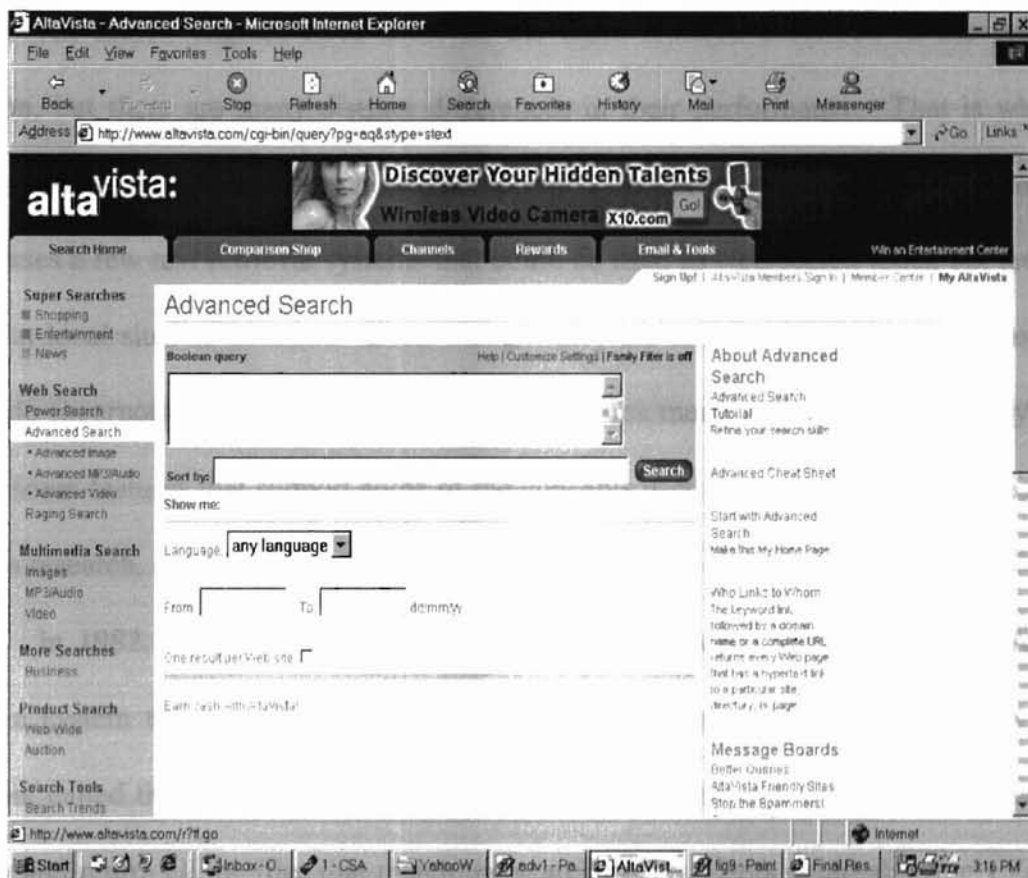


Figure 2.2. Advanced user interface

The advanced user interface (figure 2.2) is designed to help the user communicate with the search engine to find exact information he/she needs. Most of the advanced user interfaces are full of sophisticated operations that help narrow the searching area. Many of them use boolean operations and other detailed questions for the user to answer. However, most users, even novice users, want to narrow their searching area, but the complexity of the advanced user interface make it hard for users to use such an interface.

There are other web search tools also, but many users have to ask themselves--"how easy are they to use, and how well do they perform?"

## **2.2 Other Text Retrieval System**

All search engines have some of the basic features described in the previous section, but there are many distinct differences in their performance. That is why the same search on different search engines produces different results [17]. This section discusses a few text retrieval systems that could be used as web search tools, but are only used at local sites or domains. Even though they are restricted to a particular domain, they do incorporate some of the basic search features mentioned above. These systems have some features that support some of the concepts used in the search scheme designed in this research.

In 1982, the European Space Agency Information Retrieval Service (ESA-IRS) used a system that improved and supported casual user's performance. The inventor, Martin, called the system ZOOM. ZOOM was designed to take the users' initial search query and return all the files that match the search query. Users could also apply ZOOM to the initial set of files. In addition, ZOOM could compute a frequency analysis for a given field such as titles, authors, corporate sources, and abstracts. After the frequency analysis, a ranked list of results are displayed along with new terms with the option of formulating a new search query, and the ability to expand the initial search [3].

Also in 1982, Ingwersen [3] described a system using the formal framework of the traditional information retrieval process. This framework was developed using steps that define the mental stage of a user. The first step included the informational need. The

next step formulated the need. The steps that followed included the development of the search profile-topic, choice of tools, discussion based on terms, descriptions, abstracts, titles, and the evaluation of the documents. In addition to this project, in 1984, Ingwersen [3] produced another study that claimed user (searchers) must have adequate amount of "IR knowledge" to use an information retrieval system. He also claims that the system's knowledge consists of the settings, and the conceptual relations generated by authors.

By 1997, Shneiderman, Byrd, and Croft [3] proposed a user-interface framework for text searches. The framework consists of four phases: formulation phase, action, review of results, and a refinement phase. This framework is designed to give the user informative feedback concerning the initial search and how it was prepared. The entire framework was based on the idea that upon the completion of a search, it should be obvious to the user what happened and why.

In addition to his previous project, Shneiderman, in 1997, worked with Doan, Plaisant, and Bruns to develop a new approach to the network query user interface. This approach consisted of two phases: query preview and query refinement. This approach added a diverse search using dynamic queries and query previews. These concepts help guide users in eliminating undesired data sets. They also reduced the data volume into a manageable size, and help refine queries locally before submitting the query over a network [3].

Another technique that is adopted in this research is from two scientists working on user interfaces. Veerasamy and Navatle developed a user interface for a digital library that provided visual feedback and an on-line thesaurus [3].

By 1998, Hussam [3] developed a feature that helps users locate desired information using semantic highlighting. This feature allowed users to highlight a few sentences or a paragraph that enable the software to search all the highlighted text for a match and provide users with a graphical relevance of the results.

As of today, search engines are the most desired web search tools when trying to find information on the web. However, many of their interfaces are neither simple nor clear. They are designed with complex operations and obscure key features that result in confusion, frustration, and wasted time following irrelevant links [17]. According to [3], recent studies confirm that when users are given more information on their search and more control over their search, users' performance and satisfaction increase. With the understanding of this concept spawned the design and implementation of the intermediary used in this enhanced web search scheme.

### **2.3 Intermediaries**

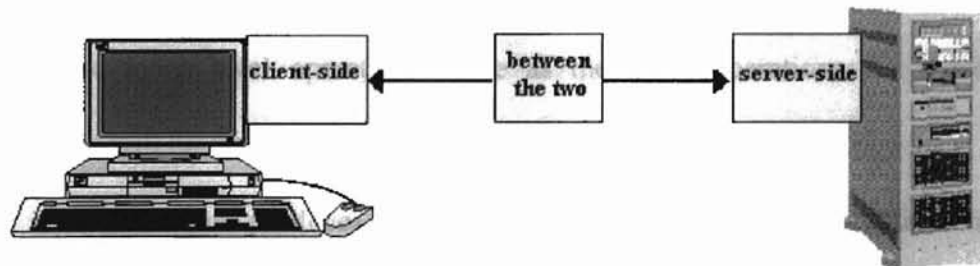
In the enhanced search scheme designed in this paper, the intermediary is the bridge between the user and search engines. This particular intermediary provides the methods that allow the users to have more informational knowledge and control over their searches.

Intermediaries are defined as "computational entities that operate on information as it flows along a stream [1]". A stream is defined as the path from one end point to another. Intermediaries are convenient due to their ability to convert common information streams into intelligent streams. These streams enhance the quality of information passed from one point to another in addition to other services [1].

The concept of intermediaries is common as other web tools, but at times, difficult to detect. Human Intermediaries are one example of intermediaries, and travel agents are considered human intermediaries. The agents' jobs are to translate customer request into data and then store that data into the airline reservation computer. The travel agents become the bridges between the customers on the telephone and the airline reservation system. In addition, the travel agents are able to enhance the process by adding additional information. Information such as suggesting ways to lower costs, explaining the results of computer queries, and so on [1].

However, today many human intermediaries are being replaced with computerized intermediaries. For instance, human gas station attendants functioned as intermediaries who controlled the gas pumps for customers in exchange for cash. Now, due to the understanding of intermediaries and the overwhelming advances in computer technology, computerized intermediaries within the gas pump can handle financial transactions [1].

Depending on implementation and performance, intermediaries can perform many different ways. They can run on the client side under the control of an individual user, or on the server-side as an information provider as displayed in figure 2.3.



**Figure 2.3. Intermediaries locations**

Locations of intermediaries are intuitive because they all perform the same five basic functions: customizing, filtering, annotating, transcoding, aggregating, and caching [ 11].

Each function performs a specific task when given certain information. For instance, the customization function gets its information from the user or the user's environment to perform its duties. An example is Amazon.com. Based on the user's previous purchases, links of personalized recommendation of books are added to web pages for others to buy. Transcoding is the process of transforming one output into another specified output. Transcoding gathers information about the data's input and suggested output. A possible scenario would be converting HTML (HyperText Markup Language) documents to a wireless mark up language for wireless phones. The aggregation function combines information from several different sources. For example, a single page of search results is produce by merging results from several search engines. Caching, another function, gathers data that were last stored or last changed from a local repository. all these functions are considered as high-level intermediary functions, however, all of these functions can be implemented by combing three lower-level operations: data generation, data editing, and data monitoring [3,11].

These lower-level operations serve as the basic operations for a web based structure that will be discussed later in this section.

Data generation produces information for other intermediary functions to use. It may also retrieve raw information from a Web server or local cache. Some examples of data generation are Common Gateway Interface (CGI) scripts and Java Servlets.

After the data generation process, the data editing function takes the raw data and edit it to meet the user needs. One need might require a transformation of raw data into a form that is compatible with a user's viewing device such as a palm pilot. Another need might be to personalize the data. An example would be constructing data in a form that the user could traverse through easily.

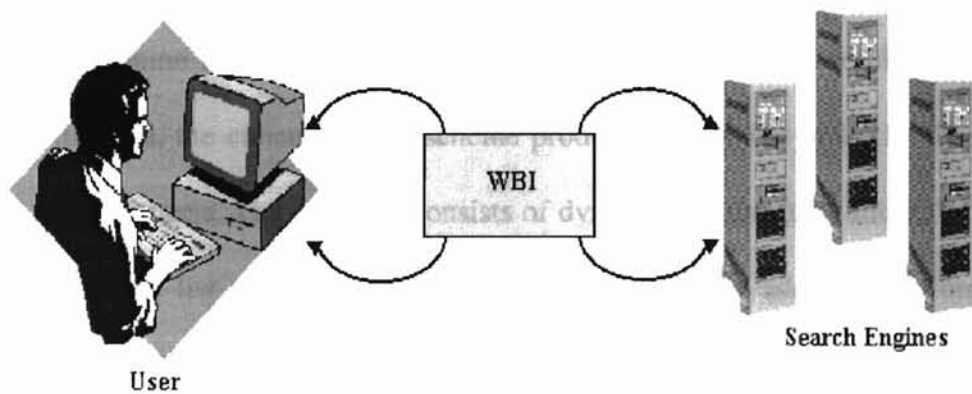
The last operation is data monitoring. Data monitoring tracks information and requests in the intermediary framework. Monitoring allows the intermediary infrastructure to cache web resources for faster retrieval, logging usage patterns, billing, and building user model that support Web browsing activities [11].

Data generation, data editing, data monitoring, and the high-level functions are the tools that make up the intermediary framework. In order to enhance the functionality of intermediaries on the Web, WBI (pronounced WeB-ee) was developed [3,11]. WBI, meaning Web intermediaries, serve as an infrastructure for designing web-based intermediaries [1]. WBI are used as the blueprints in building any web-base intermediary because they combine all the operations of high-level and low-level functions. As web intermediaries are designed to handle more information streams, they become a great advantage when sharing information among themselves and other resources on the web. For example, a mail intermediary that caches received mail. The mail intermediary can make mail available through HTML web pages to HTTP intermediaries. In order to implement this process, the mail intermediary and web intermediary need to share the mail content. The WBI is the infrastructure used to distribute information between the mail intermediary and the other intermediaries on the web [2]. In this research, the WBI architecture enhances the interaction between the user and the search engine on the web.

## CHAPTER 3

### DESIGN MODEL

The enhanced web search scheme designed in this paper is constructed with the user as its focal point. The figure below diagrams the transaction between the user, the web-intermediary, and the search engine.



**Figure 3.1. Scheme model**

The entire model is designed to be user-friendly. The combined steps from start to end are strategically detailed. The first step gives the user a chance to submit a search query. After entering their query, the user has a choice in selecting between three search engines. In the section on future work, alterations of increasing this number is mentioned, but for now, three was an arbitrary number used for the design phase. The search engine chosen will receive the search query for processing. This process will generate a list of results. At this point, the capabilities of search engines end, leaving the user helpless in finding a relevant page. However, the scheme implemented in this paper



takes advantage of the shortcomings of typical search engines, and adds a level of enhancement. With this enhancement, a user can chose from filtering through one page of results at a time or all pages at once. With the selection of either option, the scheme formulates search criterion (form) for the user to fill out. This search criterion is designed with several searching schemes that focus on extracting all the links that relate to the users' topic. With this search criterion, users avoid having to manually step through each link in the result list, and in addition, this criterion can gather enough information from the user to determine how relevant the links are in the initial set of results. After the filtering is finished, the enhanced web scheme produces a new list of URLs that match the users' search criteria. This new list consists of dynamic features that allow the user to browse through the list easier with added features that display additional information. This information shows a visual representation of how each link matches the users' criteria. In addition, a ranking feature is added in order to group the links in descending order based on the links' relevant score. At this point, users can review the list, or continue interacting with the scheme to produce a better result list. The goal in this scheme is to give the user the ability to interact with ordinary search engines. This interaction increases users' searching power and time.

The next section covers the implementation of the web-based intermediary that handles transactions between users and search engines.

## CHAPTER 4

### WEB-BASED INTERMEDIARY IMPLEMENTATION

As discussed earlier, intermediaries use three components to construct its web-base infrastructure. In this section, the implementation and functionality of these components are explained for clarity. Figure 4.1 diagrams the web-base intermediary. According to the diagram, one can see how information travels through the infrastructure and the components that handles it.

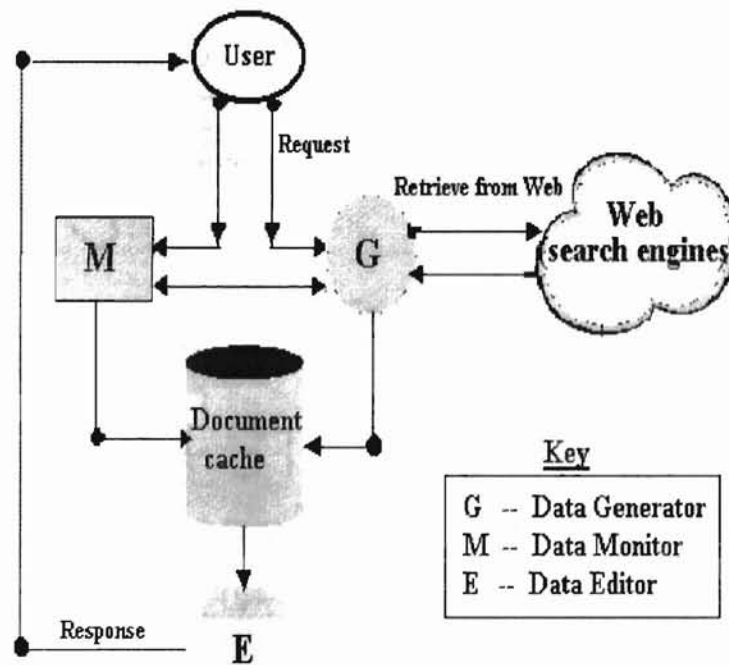


Figure 4.1. Information cycle

## 4.2 Data Generation

The functionality of this component is to generate a composite set of all pages produced by a specific search engine. The process requires the user to enter a search query in the section provided by the enhanced search scheme interface. The figure below illustrates the design of the interface.

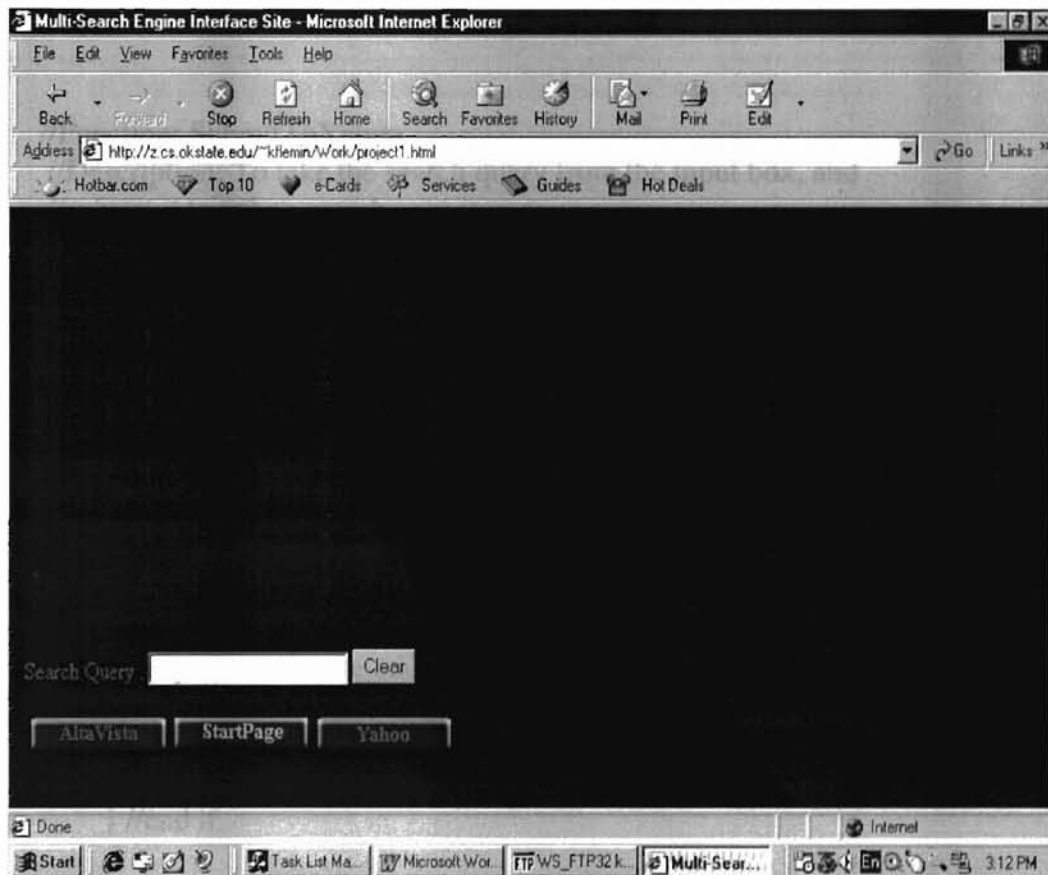


Figure 4.2. Scheme Interface

After the user enters the query, there are three search engines for selection. The data generator, implemented using CGI scripts, submits the query to the chosen search engine. The search engine produces a number of links matching the user's query. Based on the

specified search engine, the links produced by the match are divided into subsets. These subsets are HTML pages that contain a certain number of links. The search engine specified sets a default number of links to be placed on each page. However, after all links are produced, the data generator takes the first subset of links and displays them to the user. Figure 4.3 and figure 4.4 show a segment of the data generator and the effect of its execution.

```
//Function: SubmitToYahoo
//Description: To take the search query from the input box, and
//submit it to yahoo search engine.
function DisplayNewWin()
{
    var squery = "";
    squery = top.SearchWin.SearchForm.sqtxt.value;

    if(squery == "")
        alert("Please enter a search query");
    else
    {
        //This part pops up the reduce html in the third frame.
        parent.SearchWin.SearchForm.action =
            "http://z.cs.okstate.edu/~kflemin/Work/YahooWork/new-cgi/CgiPost";
        parent.SearchWin.SearchForm.submit();
        setTimeout("DisplayOutput()",3000);

    } //end if.
} //end of function.
```

**Figure 4.3. A segment of the Data Generator**

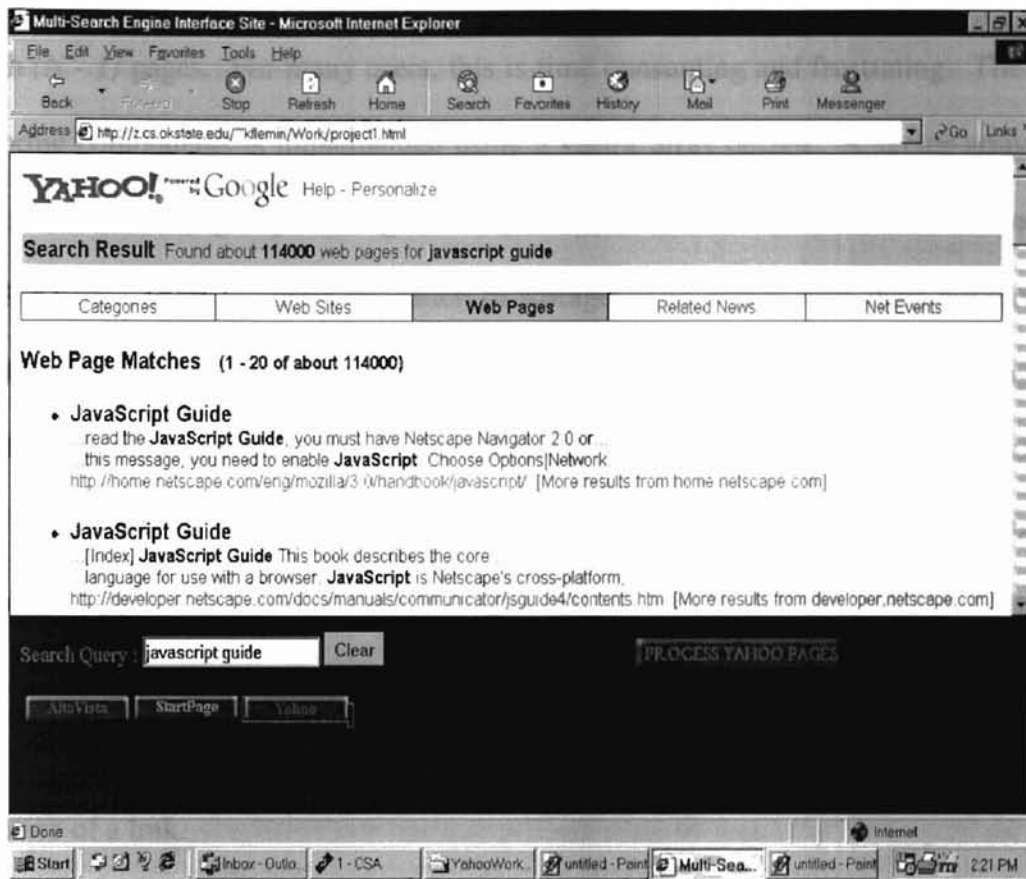
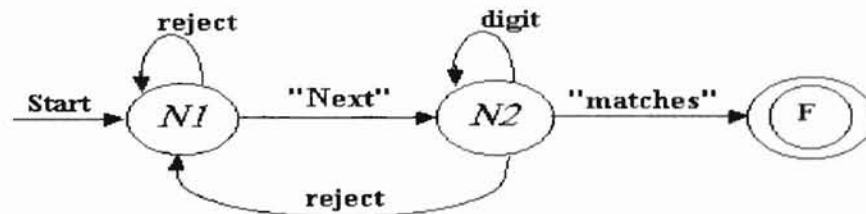


Figure 4.4. Execution output of the Data Generator

## 4.2 Data Monitoring

After the data generator performs its task, the data monitoring component waits for execution. The component is executed through the process page button selected by the user. The intuitive reasoning for this component is to provide the user with a quick reference to any page produced by a search engine. As cited earlier, search engines can produce  $N$  number of pages, where  $N \geq 0$ , from a given query. According to Yahoo's search engine, their pages are constructed in a link-list design. In order to get to any other page beside the first page, a user must traverse page by page. For instance, consider the

worst case scenario, to go from page 1 to the  $N^{th}$  page (last page), a user must traverse through  $(N - 1)$  pages. For many users, this is time consuming and frustrating. The data monitoring components is implemented using a vector array design. A vector array is a data structure used to group and organizes data in a list form. The difference between it and other arrays is the dynamic allocation of storage. The vector array does not have to pre-define the amount of storage need at the beginning of execution. It adds storage as needed, a concept used in Java and JavaScript to create dynamic array data structures [4,5,6,7,8,10,12,16]. The data-monitoring component gathers pages using a regular expression. The component searches through a list of links to find the link that connects to the next pages. The regular expression for this function is  $(Next (digit)^* matches)$ , and digit represent numbers 0-9. The finite automaton in figure 4.5, shows the acceptance of a link.



**Figure 4.5. FA diagram for page searching**

After a link is accepted, the page's link is cached into the vector array. The vector array serves as the buffer for all pages produced by the specified search engine. After all pages are buffered, the data-monitoring component produces a data information page for the user to interact with. This page consists of the numbered pages produced by the specified

search engine, and a link to each individual page. Figure 4.6 displays the execution output of the data monitoring scheme. The vector array holds the link to the pages--not the actual pages--which lead to an implementation decision between server-side caching versus local caching. The decision was to cache all pages on the server-side due to the problems with local caching such as security and capacity.

A few applications running on the web have the advantage of storing information on users' local system using cookies. Cookies are blocks of data that are stored on a client system--if granted permission [6,7,13,16]. However, this process relies on the permission from the user. The user can grant foreign applications permission to store unidentified information on their system if desired, but the fear of hackers and security breaches causes many users to deny any application from storing information on their system. With the design of this scheme, the caching process must happen in order to give the user the power and flexibility to find information. The other problem is memory capacity. An addition to the design of this scheme is its ability to be dynamic. As stated earlier, different search engines and search queries can cause the number of pages produced to increase or decrease. Since this process is not static, the memory capacity must be able to support the growth. If the capacity can not support the caching scheme, then the user and the other components in this scheme are restricted in performing other advanced tasks. Therefore, to avoid security and capacity issues, server-side caching proved to be the safest.

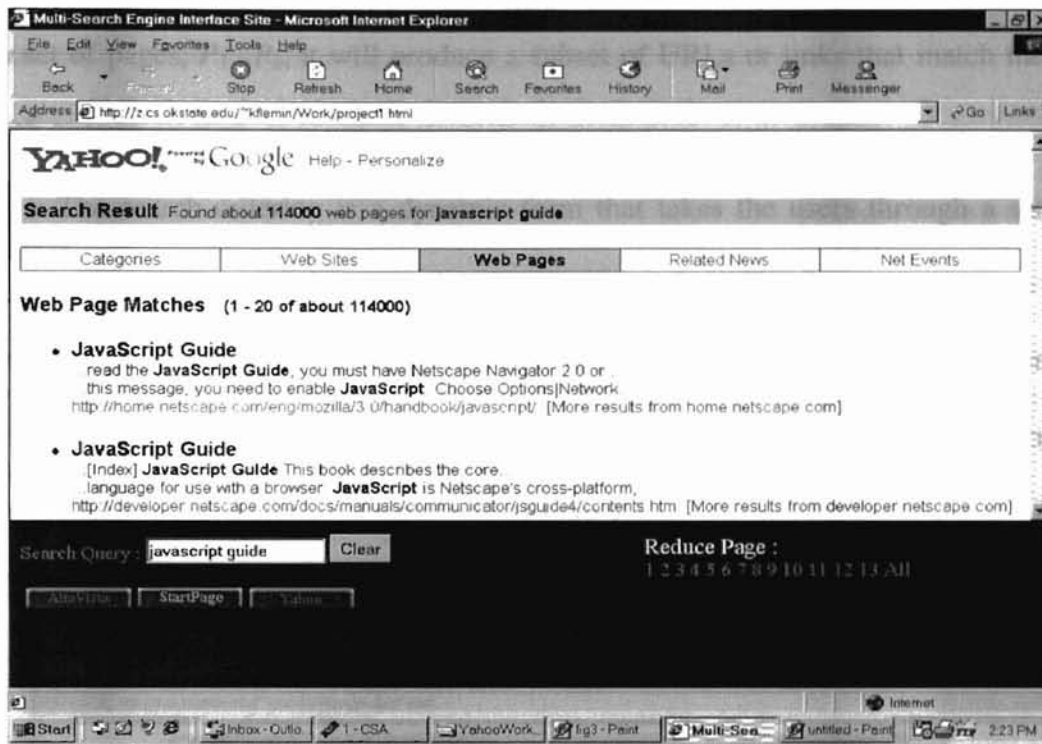


Figure 4.6. Data Monitoring interface with page numbers

### 4.3 Data Editing

In this web-based intermediary infrastructure, the data editing component is the powerhouse of all components. It contains the necessary schemes that determine a relevant page based on the user's search criteria. In addition, this component handles the filtering process, which help eliminate undesired pages.

The data editing components contain several schemes. The actual data-editing scheme is defined below:

$$\Lambda = \{ S \mid f(U) \rightarrow S, \text{ where } S \subseteq U, U \text{ are URLs from } \{ P_n \} \vee \{ P_1, \dots, P_k \}, P \text{ represent pages, and } k \geq n \geq 1 \}$$



This scheme states that  $S$  is the final set of URLs produced by the data editing component represented by  $f$ . When the function  $f$  is given a set of URLs  $U$  gathered from a page  $P_n$  or a set of pages,  $P_1...P_k$ , it will produce a subset of URLs or links that match the users search criteria.

The search criterion is a dynamic form that takes the users through a series of steps that help the user define exactly what to search for. The dynamic structure of the search criteria is constructed using several filtering schemes. These schemes produce intelligent responses based on the user's input and interaction with the search criteria. A picture of the search criteria is displayed in figure 4.7.

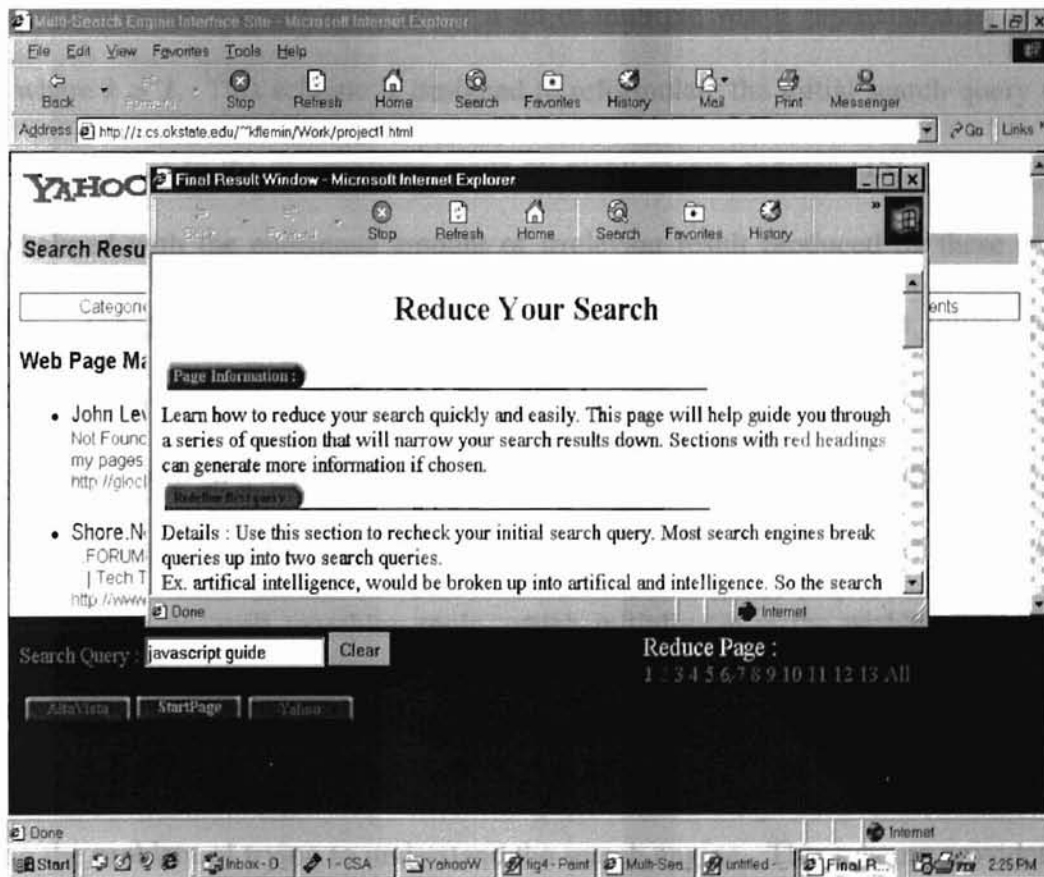


Figure 4.7. Search criterion

Each section in the criteria has a short description about the functionality of each task, and a place for user's input. Below is a descriptive list and summary of the functionality of the filtering schemes used in the back-end of the search form:

- **Initial Query Reformulation scheme**

$$S \rightarrow^R (W_1 \vee \{W_1, \dots, W_k\})$$

$R(S) = \{W \mid$  1.  $W$  is a set containing 1 word, or  
 2.  $W$  is a set containing  $n$  words, where  $n \geq 1$ ,  $n$  is the total number of words produced by  $P(S)$  }

$P(S)$ , produces  $(x+1)$  words, if there exist  $q$  elements in  $S$ , where  $q$  represents spaces and  $S$  is the initial search query string, and  $x =$  number of spaces.

The statements above state that given the initial search query string  $S$ , there is a function  $R$  that will produce a single word  $W_1$  or a set of multiple words represented by  $\{W_1, \dots, W_k\}$ , where  $k > 1$ . This scheme is designed to reformulate the initial search query used by the user. Due to the assumptions made by many search engines [15], a user can be overwhelmed with the enormous amount of irrelevant result produced by these search engines. The problem begins with the misinterpretation of the user's initial query. According to [18], one way to get the optimal performance out of a search engine is to understand the selected search engine. However, such information is preparatory and unavailable to users. Other methods of locating information about a particular subject lead to the help of web searching tools, which initially cause the problem. Now the process becomes a cycle which is time consuming and unfeasible. With such an intolerant process, the development of this scheme focus on breaking that cycle and solving the problem of trying to understand the search engine. This scheme provides the users with information on the performance of search engines, especially about the initial query supplied by the user. This scheme details how search engines handle a query

submitted by the user. This scheme also provides suggestions on reformulating the initial query in order to extract irrelevant links. Figure 4.8 illustrates the interface for this scheme.

**Redefine first query:**

Details : Use this section to recheck your initial search query. Most search engines break queries up into two search queries.  
Ex. artificial intelligence, would be broken up into artificial and intelligence. So the search would be every thing on artificial or every thing on intelligence. To avoid this situation use this section.

Check to see if the first search word can be redefined?  Yes  No

Do you wish to break *javascript guide* into 2 separate queries?  Yes  No  
If yes, then each web page will be checked for: **javascript, guide**,  
If no, then each web page will be checked for: **javascript guide**

If you choose to separate *javascript guide*, Do you wish for each word to be present in the web page searched?  Yes  No

Figure 4.8. The interface for the Initial Query Reformulation scheme

According to [4,6,7,13], when submitting a query over the web, any search query that contains a space is computed into multiple queries. Figure 4.9, diagrams the transaction to show the transformation of the query.

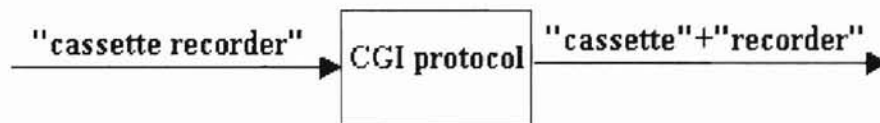


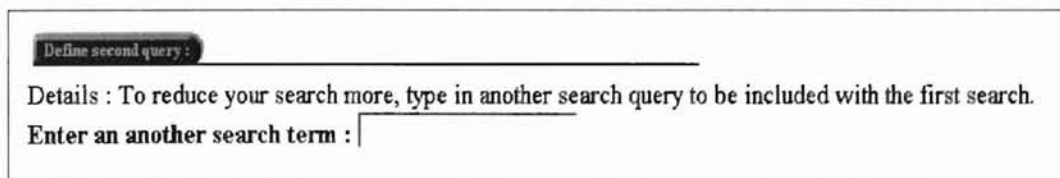
Figure 4.9. Submitting queries using CGI

According to figure 4.9, when the search engine receives the user query, the query includes a plus sign. This plus sign lets the search engine know that the search should

include all pages related to "cassette" or "recorder". Therefore, to eliminate the miscommunication between the user and the search engine, this scheme suggests two possible solutions: single search query and multiple search queries. These solutions provide a better interpretation of the initial query. The single query option lets the user chose to keep the initial query as one search query, and the multiple search query option allows the user to divide the initial query into multiple search queries. In addition, this scheme suggest to the user that if the multiple search option is chosen, then the user can select to have all words present or any words can be present in a searched document. The creation of this scheme is based on users' performance and satisfaction issues mentioned in chapter 2.

- **Second Query Analyzer**

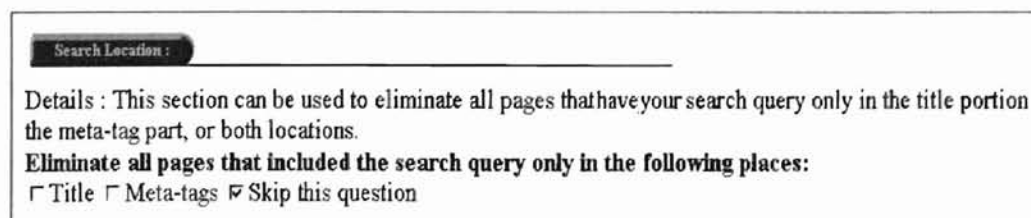
The second query analyzer scheme has some of the same concept and implementation as the previous scheme, but this one allows for a second query to be submitted. This scheme strengthens the ability of allowing users to narrow their search and eliminate irrelevant results. Figure 4.10 shows how the user can interact with this scheme.



**Figure 4.10. The interface for the Second Query Analyzer**

- **Block Sensitive Search**

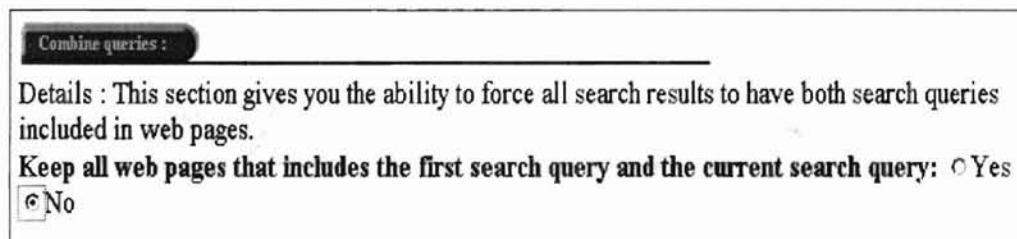
This scheme focuses on search words found in a particular location. By design, each web page can be divided into three blocks: Title block, Meta-tag block, and the body block. According to the assumptions and interpretation by search engines, most search engines search three blocks, but give preference to results that are returned with the search word occurring only in the title, or the meta-tag, rather than in the body. In addition, the ranking scheme ranks these results very high in the result list [19]. According to [19], a project was conducted to test the performance of different search engines. The test showed that many search engines assume that if a word does occur in the title block then the whole document is related to the user's subject. However, not all titles or meta-tag represent the document it labels, therefore, a careless assumption adds to the number of irrelevant links. In order to avoid this assumption, this scheme gives the user an option as to which block can be eliminated from the search. Users can choose either the meta-tag, or the title block, or both. The body block is not an option because all words should be contained in the body. An advantage of this scheme is to exclude any links that has words occurring only in the title or meta-tag block. Figure 4.11 displays how users can select either block for elimination.



**Figure 4.11. Selecting blocks in the scheme**

- **Integrated Query Search**

This scheme allows the integration between the initial query and the second query provided by the user. This scheme prompts the user to choose either "yes" or "no" for the integration. If yes is chosen, then the search acts as a boolean *AND* operation. Meaning the user has forced both queries to be present in a document. If no is chosen then the search acts as a boolean *OR* operation, and either query or both queries can be present in a document. Figure 4.12 illustrate the integrated query search.



**Figure 4.12. Integrated Query option**

- **Related Word Composer**

The implementation of this scheme is to produce more supporting search words for the user to chose. This scheme interacts with three online dictionaries: computer, medical, and Webster. With this scheme, the user can enter a word and chose an online dictionary. If the online dictionary supports that word, then a list of supporting words are displayed. From the list, the user can pick which supporting word to add to the search criterion. A great advantage of this scheme is its ability to strengthen the semantics of the user's query. This scheme produces supporting words, and the more words the user picks

the better the semantics is defined. Take for instance searching for acronym words such as ATM. Most search engines do not consider the semantic meaning behind words, and if they do, there are many assumptions made. Since ATM can represent Automated Teller Machine or Asynchronous Transfer Mode, the semantics can be lost if users do not specify them. Using this scheme allows a user to strengthen the semantic of a query, and to eliminate pages that do not match the semantics. Figure 4.13 displays the design layout of the scheme.

**Add related words :**

Details : Use this section to find more related words using various databases. Choose from a list of words.

Select related word(s) for  ?

Subjects : Computer/Internet , Medicine , General , Clear-Area

**Related Word(s) Section :**

<input type="checkbox"/> language	<input type="checkbox"/> Netscape	<input type="checkbox"/> platform
<input checked="" type="checkbox"/> World-Wide Web	<input type="checkbox"/> scripting language	<input checked="" type="checkbox"/> Java
<input type="checkbox"/> server	<input type="checkbox"/> scripting	<input type="checkbox"/> server-parsed HTML
<input checked="" type="checkbox"/> browsers	<input type="checkbox"/> C	<input type="checkbox"/> syntax
<input type="checkbox"/> forms	<input type="checkbox"/> database	<input type="checkbox"/> front-ends
<input type="checkbox"/> Microsoft	<input type="checkbox"/> Netscape Navigator	<input type="checkbox"/> Sun
<input type="checkbox"/> VBScript	<input type="checkbox"/> Home	<input type="checkbox"/> Yahoo
<input type="checkbox"/> Usenet	<input type="checkbox"/> comp.lang.javascript	<input type="checkbox"/>
<input type="checkbox"/> Java Open Language Toolkit	<input type="checkbox"/> Java Remote Message Protocol	<input type="checkbox"/> Java Runtime Environment
<input checked="" type="checkbox"/> JavaScript	<input type="checkbox"/> Java servlet	<input type="checkbox"/> JavaServer Pages
<input type="checkbox"/> Java servlet		

**Figure 4.13. Related Word section**

- **Persistent Word Search**

This scheme keeps count of the number of occurrences of the user's initial query in a given document. After the number is computed, it is compared to the number input by the user. With this scheme, the user can set how many times the initial query should be

matched in a document. If a document does not meet this condition, then it is eliminated from the list of results. Figure 4.14 shows the layout of this scheme.

Occurrence number: \_\_\_\_\_  
Details : Set the number of times a word should be displayed in a web page.  
Include all pages whose word count is greater than :

Figure 4.14. Persistent Word Search option

- **Boolean Interpreter**

This concept is used in many of the advanced search engine schemes. This scheme adds to the clarification of two search queries. The boolean operators used in this scheme are *AND* and *OR*. The *AND* operator forces both search queries to be contained in each document searched. The *OR* operator shows flexibility. It allows either query or both queries to be present in a document. Figure 4.15 illustrates the interpreter.

Boolean section: \_\_\_\_\_  
Details : Use the boolean operators to reduce your search.  
Use the Boolean Section ?  Yes  No  
Boolean Section :  

javascript	And	guide
	And	
	And	

Figure 4.15. Boolean section



- **Pattern Sensitive Search**

This scheme deals with the ordering of multiple words. The user's initial query and second query are used to form a list of multiple words. After the words have been generated, the user can select the ordering of the words. The ordering is a permutation ordering. The scheme will generate  $N$  words, and the user can order the list  $N!$  ways [9] to capture the semantics of the query. The figure below shows how the user can set the ordering using pulldown menus.

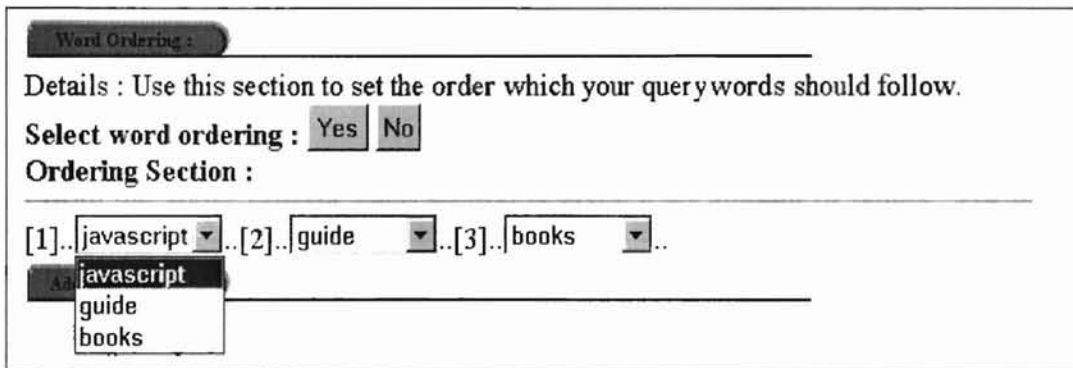


Figure 4.16. Example of the ordering diagram

- **Status Page Recognizer**

This scheme determines a page ranking, sometimes referred to as relevant score. The ranking is divided into three status values: high, medium, and low. If a page receives a high ranking, then it meets or exceeds the optimal criteria deduced from the users search criterion. A medium ranking ranges from half of the optimal high score to the high score. A low ranking is any score below the minimum medium range. If not the same criterion, each criterion can produce a different optimal score. Figure 4.17 shows the equation for the optimal score.

$$\text{Optimal score} = (\text{section points}) + (\# \text{ of single or multiple words produced from the user's initial and second query})$$

**Figure 4.17. The optimal score equation**

From a given criterion, the computation of the optimal score can be based on the following:

- ◆ a user chooses to order the words
- ◆ a user chooses to use the boolean section
- ◆ a user chooses to use related words
- ◆ a search word is found in the title, meta-tag, or body
- ◆ the total occurrence count for each word the user chose to be present in a document

Then from the above criteria, each section is given a set amount of points. For instance, using the boolean section is 40 points. The related word section is 50 points, which will be added to the total score. The list below shows the point distribution of each section.

**TABLE I  
POINT DISTRIBUTION**

<b>Section</b>	<b>Points</b>
Related word section	50
Ordering section	20
Boolean section	40
Title section	10
Meta-tag section	20
Body section	30

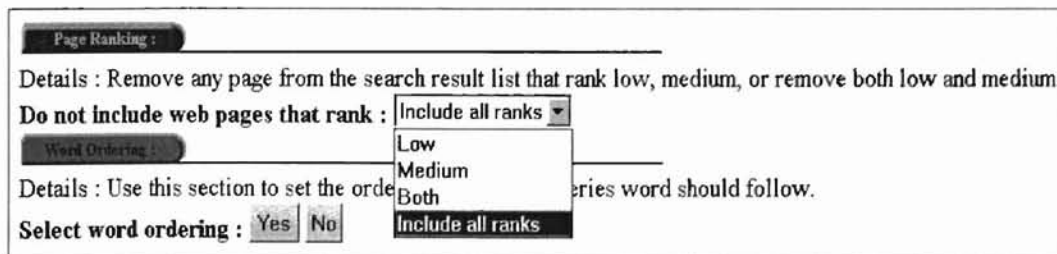
After the section points are calculated, the number of single and multiple words must be computed. However, computing the initial search query and the second search query score requires additional strategies. Depending on the user choice on selection single queries or multiple queries, each word receives a score of one point. Meaning, each word

must be present in a document at least once. Therefore the total number of single and multiple words are added to the section points to make up the optimal score. However, each document that is searched is given an actual score.

$$\text{Actual score} = (\text{section points}) + (\# \text{ of occurrences for single and multiple words found})$$

**Figure 4.18. Actual score equation**

The actual score is calculated in the same form as the optimal score, but focuses on words found in the document. The section points are scored if any of the words specified by the user is found in that section. For instance the title section, the title section points are not added to the actual score until there is a word occurrence in the title section. If a word does occur in the title section, then the actual scores adds the title section points to the number of times that word actually occurred in the title section. After the actual score is computed, the page is given a rank of high, medium, or low. An advantage of this scheme is its ability to allow the user to choose which ranked pages to eliminate from the result list. Users can chose to eliminate all pages that rank low, medium, or eliminate both low and medium pages leaving all pages that are related to the user's subject. Figure 4.19 illustrates this ability.



**Figure 4.19. Setting page ranking**

After applying the search criteria to the set of URLs from a set of pages, the data-editing scheme caches all the URLs that match the user's criterion. The URLs are displayed to the user using Dynamic HTML and JavaScript. This final page shows the user how each section in the search criterion performed its task and additional information about the page description and ranking. Figure 4.20 shows a snapshot of a final list.

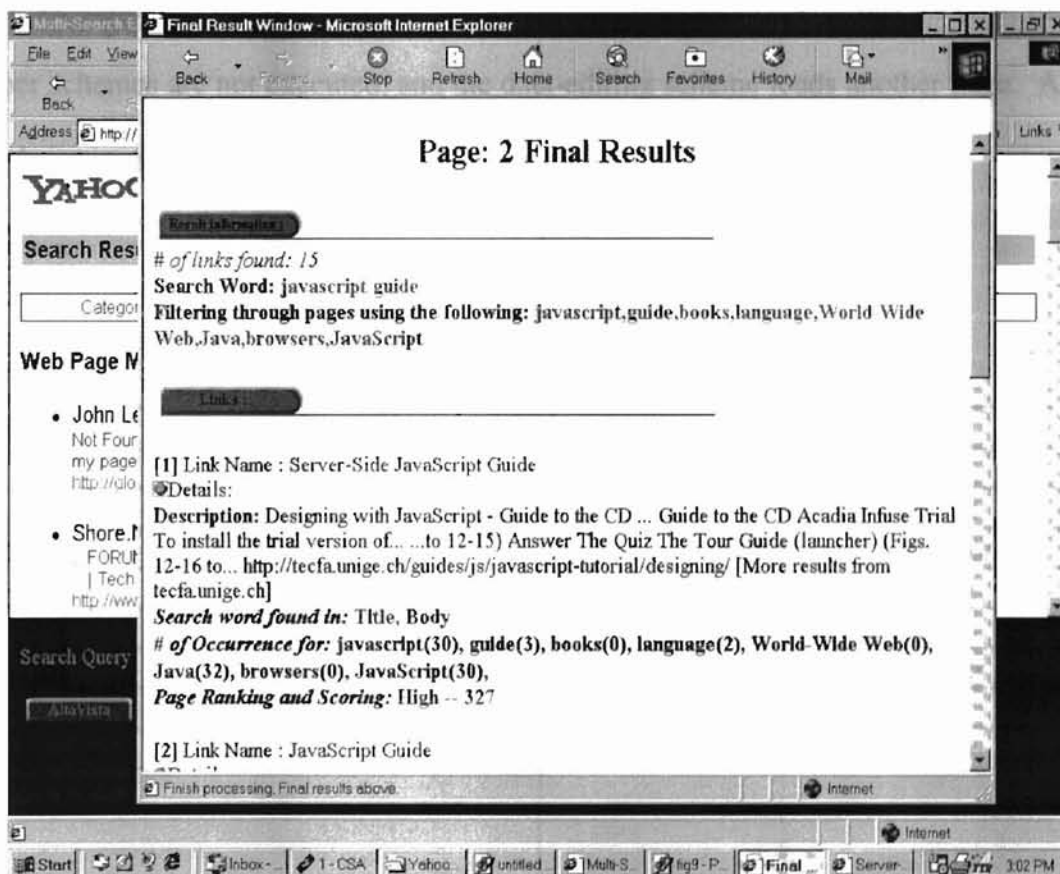


Figure 4.20. Final list snapshot

#### 4.4 Performance and Security Issues

The performance is based on a few specifications such as browsers' capabilities and modem design. In order to meet optimal performance an Internet 5.5 browser is required,

a 56Kbps modem or higher is required, and the JavaScript enable option must be set as a browser option. The interior structuring of the scheme execution helps to increase performance. Local caching would increase the performance; however, security issues made it difficult to consider. The ordering of the filtering schemes help improve the search time. Based on the user search criteria, performing a certain scheme first can dictate whether another scheme should execute. For instance, the persistent search scheme starts the filtering process. If a page fails to comply with this scheme, then the other schemes are not executed, and the data-editing scheme loads another page. A chart of the execution cycle of the filtering scheme is below.

**TABLE II  
FILTERING SCHEME CYCLE**

Order Number	Scheme
1	Persistent search
2	Pattern Sensitive search
3	Initial Query reformulation
4	Second Query Analyzer
5	Boolean Analyzer
6	Related Word Composer
7	Block Sensitive search
8	Status Page Recognizer

Another security issue that effects the performance is the forbidden ability to manipulate web document across server domains. The Enhanced Search Scheme interacts with different search engines to gather links from their result pages. The ability to use those links across different server domains directly is a security problem. According to web standards, only users on the same domain can manipulate their web

documents. In order to manipulate a search engine's page, this scheme had to regenerate the page on its host machine. After regenerating the page, the page was accessible for the data generator to manipulate.

## CHAPTER 5

### CONCLUSION

#### 5.1 Summary

In this thesis, the design and implementation of an enhanced web search scheme is presented. This scheme provides time-saving techniques using a web-base intermediary that help clarify users' search queries and extract relevant links from ordinary search engines results. This scheme also provides a user-friendly interface that is implemented on top of search engines. In test cases, this scheme has shown reduction in unrelated links, and thus, increase users' search performance on the web.

#### 5.2 Future Work

In this thesis, the Enhanced Search Scheme focuses on time and usability. Future work in increasing time performance could be in using Java Servlets instead of CGI scripts. Java Servlets connect to the server initially which allows for communication between the web page and the server to remain open until the page terminated. However with CGI scripts, there connection is short term. Meaning, for every transaction between the web page and the server, a new process will have to start and stop which increases connection time and slows execution performance. Other future work could be increasing usability by adding more search engines to the scheme. This scheme uses three, but adding more would give the user a variety of search engines to poll. Also, adding more filtering options would increase usability. Filtering options can focus on the state of a page. State being the status of a page such as new or old. Such filtering options could be

capturing the time of a page (how long it's been on the web), number of hits for a page, or the last time a page was referenced.



## BIBLIOGRAPHY

- [1]. Barrett, Rob and Maglio, Paul P. "Intermediaries: An approach to manipulating information streams." IBM Systems Journal, Vol. 38 (4), 1999, pp. 629-41.
- [2]. Barrett, Rob and Maglio, Paul P. "Intermediaries: New places for producing and manipulating Web content." Computer Network and ISDN Systems, 1999, pp. 509-518.
- [3]. Fang, Xiaowen and Salvendy, Gavriel. "Keyword comparison: a user-centered feature for improving web search tools." Int. J. Human-Computer Studies, November 1999, pp. 915-31.
- [4]. Flanagan, David. JavaScript: The Definitive Guide, Third Edition. Sebastopol, CA: O'Reilly and Associates, Inc., 1998.
- [5]. Frentzon, Jeff and Sobotka, Henry. JavaScript Annotated Archives. Berkeley, CA: Osborne/McGraw-Hill, 1998.
- [6]. Goodman, Danny. Dynamic HTML: The Definitive Reference. Sebastopol, CA: O'Reilly & Associates, Inc., 1998.
- [7]. Goodman, Danny. JavaScript Bible, 3rd Edition. Foster City, CA: IDG Books Worldwide, Inc., 1998.
- [8]. Hall, Marty. Core Web programming. Upper Saddle River, NJ: Prentice Hall PTR, 1998.
- [9]. Johnsonbaugh, Richard. Discrete Mathematics. Englewood Cliffs, NJ: Macmillan Publishing Co., 1993.

- [10]. Ladd, Eric and O'Donnell, Jim. Using HTML 4.0, Java 1.1, and JavaScript 1.2. Indianapolis, IN: Que Corporation. 1998.
- [11]. Maglio, Paul and Barrett, Rob, "Intermediaries Personalize Information Streams." Communications of the ACM. Vol. 43 (8), August 2000.
- [12]. Mudry, Robert. The DHTML Companion. Upper Saddle River, NJ: Prentice Hall PTR., 1998.
- [13]. Niles, Robert and Dwight, Jeffrey. CGI by Example. 1998.
- [14]. Schwartz, Candy. "Web Search Engines." Journal of the American Society for Information Science. Vol. 49 (11), September, 1998, pp. 973-82.
- [15]. Sneiderman, Ben, Byrd, Don and Croft, Bruce. "Clarifying Search: A User-Interface Framework for Text Searches." D-Lib Magazine, January 1997.
- [16]. White, Chuck. Internet Explorer 5 Developer's Guide. Foster City, CA: IDG Books WorldWide, Inc., 1999.
- [17]. "How Search Engines Work",  
<http://searchenginewatch.com/webmasters/work.html>.
- [18]. "Search Engines", <http://www.webreference.com/content/search/bkground.htm>.
- [19]. "Tips on Popular Search Engines",  
<http://www.homline.edu/library/bush/handouts/slahandout.html>.

## APPENDIX I

### GLOSSARY

**CGI** - Acronym for Common Gateway Interface. The specification that defines communications between information servers and resources on the server's host computer, such as databases and other programs.

**Cookie** - a block of data that a web server stores on a client system.

**FTP** - Acronym for File Transfer Protocol. The protocol used for copying files to and from remote computer systems on a network using TCP/IP, such as the Internet.

**Gopher** - an Internet utility for finding textual information and presenting it to the user in the form of hierarchical menus, from which the user select submenus, files, or documents that can be downloaded and displayed.

**HTML** - Acronym for Hypertext Markup Language. A tool used to create web documents for the World Wide Web.

**HTTP** - Acronym for Hypertext Transfer Protocol. It is the client/server protocol used to access information on the web.

**Human Intermediaries** - human beings providing the necessary communications between two or more parties.

**HYTELNET** - a menu-driven index of Internet resources that are accessible via telnet, including library catalogs, databases, bulletin boards, and network information servers.

**Interoperability** - referring to components of computer systems that are able to function in different environments. With software, interoperability occurs when programs are able to share data and resources.

**Newsgroup** - a forum on the Internet for threaded discussions on a specified range of subjects. A newsgroup consists of articles and follow-up posts.

**Search Engine** - a program that searches for keywords in a document or a database. On the Internet, it is a program that searches for keywords in files and documents found on the web, in newsgroup, gophers, and in FTP archives.

**Search Query** - a term or condition that search engines use to find information.

**Search Results** - a set of pages containing a set of URLs to web sites that match a user's topic.

**Stream** - to transfer data continuously--beginning to end--in a steady flow.

**Telnet** - a protocol that enables an Internet user to log on to and enter commands on a remote computer linked to the Internet.

**URL** - Acronym for Uniform Resource Locator. An address for a resource on the Internet. URL are used by Web browsers to locate Internet resources.

**WAIS** - Acronym for Wide Area Information Server. A UNIX-based document search and retrieval system on the Internet that can be used to search over 400 WAIS libraries for indexed files that match a series of keywords. Users need a WAIS client to use a WAIS server.

**WWW** - Acronym for World Wide Web. The total set of interlinked hypertext documents residing on HTTP servers all around the world. Documents on the World Wide Web are called pages or web pages.

VITA

Kerry R. Fleming

Candidate for the Degree of

Master of Science

Thesis: AN ENHANCED WEB SEARCH SCHEME

Major Field: Computer Science

Biographical:

Personal Data: Born in Jackson, Mississippi, the son of Noah and Dorothy Coffee.

Education: Graduated from Kosciusko High School, Kosciusko, Mississippi in May 1993; received Bachelor of Science degree in Mathematics with emphasis in Computer Science from Tougaloo College, Tougaloo, Mississippi in May 1997. Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in December, 2000.

Experience: Lecturer and Teaching Assistant at Oklahoma State University, Stillwater, Oklahoma, 1997 to present.