INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Long Query as an Effective Method for Improving the Quality of Information Retrieval on the Web

by

Isak Taksa

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

2002

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

10/24/01
Date

Professor Jacob Shapiro
Chair of the Examining Committee

10/24/01
Date

Professor Theodore Brown
Executive Officer

Professor Linda Friedman

Professor Pai-chun Ma

Professor D. Frank Hsu

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

THE CITY UNIVERSITY OF NEW YORK

Abstract

Long Query as an Effective Method for Improving the Quality of Information Retrieval on the Web

by

Isak Taksa

Adviser: Professor Jacob Shapiro

The quality of information retrieval on the Web is a subject of great interest to the Information Retrieval community as well as to a very large and growing community of Web users. This thesis addresses how long queries can improve the quality of information retrieval on the Web. Long queries allow the user to write the query in natural language, making it easier for the user to completely describe the information need. Using long queries in the Web environment has not been addressed in its entirety in the scientific literature nor has it been adequately incorporated into any commercial search engine.

We propose and analyze several novel algorithms dealing with all aspects of long query information retrieval on the Web such as selecting of search terms and phrases, constructing multiple query formulations, merging and ranking search results. We also present a user feedback option to direct the query formulation process and a new user interface especially designed to handle long queries.

We developed a meta-search engine, which incorporated the proposed algorithms, the feedback option, and the user interface. We performed a series of experiments to evaluate the new meta-search engine and compare its search results with the results of popular search engines on the Web. These experiments clearly demonstrated that using long queries in the Web environment is practical and can substantially improve the quality of information retrieval.

Acknowledgements

I would like to express my deepest gratitude to my adviser Professor Jacob Shapiro for his time, encouragement and expert advice throughout the exciting and often challenging process of working on my dissertation. I am indebted to the members of the examining committee: to Professor Linda Friedman for her interest and indispensable observations; to Professor Pai-chun Ma for his input and comments during the preliminary phase of this research; to Professor D. Frank Hsu for his critical review and useful advice. I appreciate the help and valuable comments of Professor Theodore Brown, Executive Officer of the Ph.D. Program in Computer Science. Special thanks go to Mr. Joseph Driscoll for guiding me through the administrative maze of graduation process.

TABLE OF CONTENTS

ABSTRACT iv

LIST OF TABLES x

LIST OF FIGURES xii

Chapter

1 INTRODUCTION 1

1.1 Motivation 1
1.2 Satisfying User's Information Needs 4
1.3 Objectives of this Research 6
1.4 Organization 8

2 PREVIOUS AND RELATED WORK 11

2.1 Process of Information Retrieval 11
2.2 Creating and Using Long Queries 11
2.3 Using Phrases to Focus Long Queries 12
2.4 Choosing the Number of Search Terms in a Query 13
2.5 Query Formulations and Multiple Results Presentations 14

3 DESIGNING THE LONG QUERY SEARCH ENGINE 15

3.1 Guiding Principles 15
3.2 Building Blocks 16

4 USER INTERFACE AND FEEDBACK 19

4.1 Accepting the Long Query 19
4.2 Query Refinement Process 20
4.3 Selecting Algorithms and Operational Parameters 22
4.4 Submitting Subqueries to Search Engines and Merging Results . . . 23
4.5 Reviewing and Saving Search Results 26
4.6 Monitoring the Search Process 27

5 DESCRIPTION OF THE PROPOSED ALGORITHMS 29

5.1 Order Query Terms (OQT) Algorithm 31
5.2 Query Phrase Extractor (QPE) Algorithm 35
5.3 Open End Query (OEQ) Algorithm 36
5.4 Close End Query (CEQ) Algorithm 38
5.5 Single-Engine Fusion (SEF) Algorithm 40
5.6 Multiple Presentation Counter (MPC) Algorithm 41
5.7 Similarity Analysis Calculator (SAC) Algorithm 42

6 EVALUATING THE PROPOSED ALGORITHMS 47

6.1 Judging Relevancy of Search Results 47
6.2 Position of the Original Document 48
6.3 Precision @ 10 & Precision @ 20 49
6.4 Recall and Relative Recall 49
6.5 Precision at 11 Standard Recall Levels 51

7 DESCRIPTION OF EXPERIMENTS 55

7.1 Structuring the Experiments 55
7.1.1 Classifying the Experiments 56
7.1.2 Developmental Experiments 57
7.1.3 Comparative Experiments 60
7.1.4 User Feedback Experiments 61
7.2 Creating the Experiment Environment 62
7.2.1 Putting Together a Query Collection 62
7.2.2 Selecting Search Engines 62
7.2.3 Collecting and Analyzing Data 63
7.3 Organizing and Conducting the Experiments 63
7.4 Experimental Results 64
7.4.1 Open End Engine Experiments 64
7.4.2 Close End Engine Experiments 67
7.4.3 Similarity Analysis Calculator Experiments 71
7.4.4 Comparative Experiments 74
7.4.5 User Feedback Experiments 78
7.5 Discussion of Experimental Results 81
7.5.1 Fine Tuning the Algorithms 82
7.5.2 Comparing the New Engine with the Commercial Engines . . . 84
7.5.3 Applying User Feedback 85

8 CONCLUSIONS 86

8.1 Contributions 86
8.2 Results 87
8.3 Future Work 89

9 ABBREVIATIONS 90

10 APPENDICES 91

A Google and AltaVista Initial Windows 92
B Sample Long Query 93
C Long Query Search Engine Log 94
D Stop Words and Symbols 95
E Words and Documents Frequencies 96
F Operational Parameters for the Close End Algorithm 100
G Experiment Guide 101
H Collection of Long Queries (samples) 104
I Categorization of Long Queries 105
J Data Collection Workbook 107
K List of Experiment Configurations 109
L Position of the Original Document (Top 20 Search Results) . . 110
M Precision *a* 10 & Precision *a* 20 111
N SAC Algorithm Interface 112
O SAC Algorithm (Input/Output Data) 113

11 BIBLIOGRAPHY 114

LIST OF TABLES

1-1	Comparative Characteristics of Concrete Information Need (CIN) and Problem Oriented Information Need (POIN)	5
5-1	Query Terms File Layout	32
5-2	Predecessor Successor Pairs	35
5-3	Subquery Composition Recursive Process	37
6-1	Ternary Relevancy Scale	48
6-2	Calculating Precision Recall Pairs	52
6-3	Calculating Interpolated Precision at 11 Standard Recall Levels	53
7-1	Developmental Experiments (Open End Configurations)	58
7-2	Developmental Experiments (Close End Configurations)	59
7-3	Developmental Experiments (SAC Configuration)	60
7-4	Comparative Experiments (LQSE vs. Commercial Search Engines)	61
7-5	User Feedback Experiments (Long Query Search Engine)	61
7-6	Placing of the Original Document among the Top 20 Results for Open End Configurations (%)	65
7-7	P@10 & P@20 for Open End Configurations	66
7-8	Interpolated Precision at 11 Standard Recall Levels for Open End Configurations	66
7-9	Placing of the Original Document among the Top 20 Results for Close End Configurations (%)	69
7-10	P@10 & P@20 for Close End Configurations	70

7-11 Interpolated Precision at 11 Standard Recall Levels for Close End Configurations 70

7-12 Placing of the Original Document Among the Top 20 Results for SAC Experiments (%) 73

7-13 P@10 & P@20 for SAC Experiments 73

7-14 Interpolated Precision at 11 Standard Recall Levels for SAC Experiments 73

7-15 Placing of the Original Document Among the Top 20 Results for All Configurations (%) 76

7-16 P@10 & P@20 for All Configurations 77

7-17 Interpolated Precision at 11 Standard Recall Levels for All Configurations 77

7-18 Placing of the Original Document Among the Top 20 Results for Open and Close End Configurations in Automatic and User Feedback Mode (%) 79

7-19 P@10 & P@20 for Open and Close End Configurations in Automatic and User Feedback Mode 80

7-20 Interpolated Precision at 11 Standard Recall Levels for Open and Close End Configurations in Automatic and User Feedback Mode . . 80

LIST OF FIGURES

1-1	Information Retrieval Process (Evolution)	3
3-1	Long Query Processing Algorithms	17
4-1	Long Query Search Engine (The Initial Screen to Enter a Long Query and to Select A Search Engine)	19
4-2	Long Query Search Engine (The Query Refine Screen with a Phrase Builder)	21
4-3	Long Query Search Engine (The Search Results Page)	22
4-4	Long Query Search Engine (Selecting the Operational Parameters for the Close End Algorithm)	23
4-5	Long Query Search Engine (The Top of the Log - Links to Various Parts of the Log)	24
4-6	Long Query Search Engine (Individual Subquery Search Results)	25
4-7	Long Query Search Engine (The Final Output)	25
4-8	Long Query Search Engine (The Bottom of the Log – List of URLs)	26
4-9	Long Query Search Process	27
5-1	Long Query Processing Steps	30
5-2	Open End Query Composition	38
5-3	Close End Query Composition	39
5-4	Merging of Intermediate Search Results	40
5-5	Using SAC Algorithm	43

7-1 Interpolated Precision at 11 Standard Recall Levels for Open End Configurations 67

7-2 Interpolated Precision at 11 Standard Recall Levels for Close End Configurations 71

7-3 Interpolated Precision at 11 Standard Recall Levels for SAC Experiments 74

7-4 Interpolated Precision at 11 Standard Recall Levels for All Configurations 78

7-5 Interpolated Precision at 11 Standard Recall Levels for Open and Close End Configurations in Automatic and User Feedback Mode 81

1 Introduction

1.1 Motivation

Years ago, when people looked for information, they went to a library and spoke to experienced librarians. The librarians knew how to evaluate questions even when they were not clearly articulated. The librarians also knew how the information was organized and stored and could therefore direct people to the appropriate catalogue boxes with library cards. Library users would have to study the cards in those catalogue boxes for *titles* of locally available items (manuscripts, magazines, scientific papers, etc.) and would then have to *guess* which titles (or key words on the card) seemed to be relevant and then submit a request for items. The librarian would then retrieve the requested items from storage. If the title was vague or the key word was ambiguous, the retrieved item would likely be irrelevant.

When OPAC (on-line public catalogues) replaced librarians, people began to learn how to "talk" to computers. As information seekers learned to use Gopher and then Mosaic, they were "educated" to express their information needs through a list of keywords or terms. These were the original short queries.

With the widespread use of the Web and its more than 2.1 billion publicly available pages [K00], search engines have become a critical factor in assisting users in finding information on the Web. At the same time, the fact that major providers of popular search engines are commercial enterprises and every pixel on the computer monitor is an income producing property has pressured search engines to limit their query

windows to a small amount of space (see Appendix A). A Web log analysis of AltaVista [SHMM98] and Excite [JS00. SX00] conducted in the last couple of years demonstrates that the average query length is between 2 and 3 terms. But with the amount of information currently available on the Web and its astronomical rate of growth of 7 million pages added daily [K00]. it has become more difficult for a user to find relevant information based on a few terms. Moreover. most users are not sufficiently trained to select the best terms to get the results that they seek.

The problem of improving search quality on the Web has received great attention both in academic circles. as witnessed by the large number of conferences and publications. and in the business world. as witnessed by thousands of general and specialized search engines available on the Web. Many academic studies on improving search quality on the Web concentrate on the analysis of document structure and the interrelationships among documents (links). Fewer studies deal with expressing user information needs and the formulation of queries. Most established search engines and newer search engines attempt to improve search results by incorporating new and improved algorithms: indexing algorithms. search algorithms. and ranking algorithms. There is not enough innovation in the area of user interface to provide users with more and better tools for more precise query formulations. such as user feedback and original query refinement. However. the quality of the information retrieval process largely depends on the user-formulated query. and therefore it is essential that such a query represents the actual user's information need as closely as possible. A way to insure proper conversion of user information need into an adequate query is to allow the user to

input a long query. A long query allows the user to formulate the query in natural language and thus to better and more completely describe the user's information need.

The following figure illustrates how the information retrieval process evolved from a visit to a library (before the Web), to a Web search using contemporary search engines, and our view of how this process could be improved by using long queries.

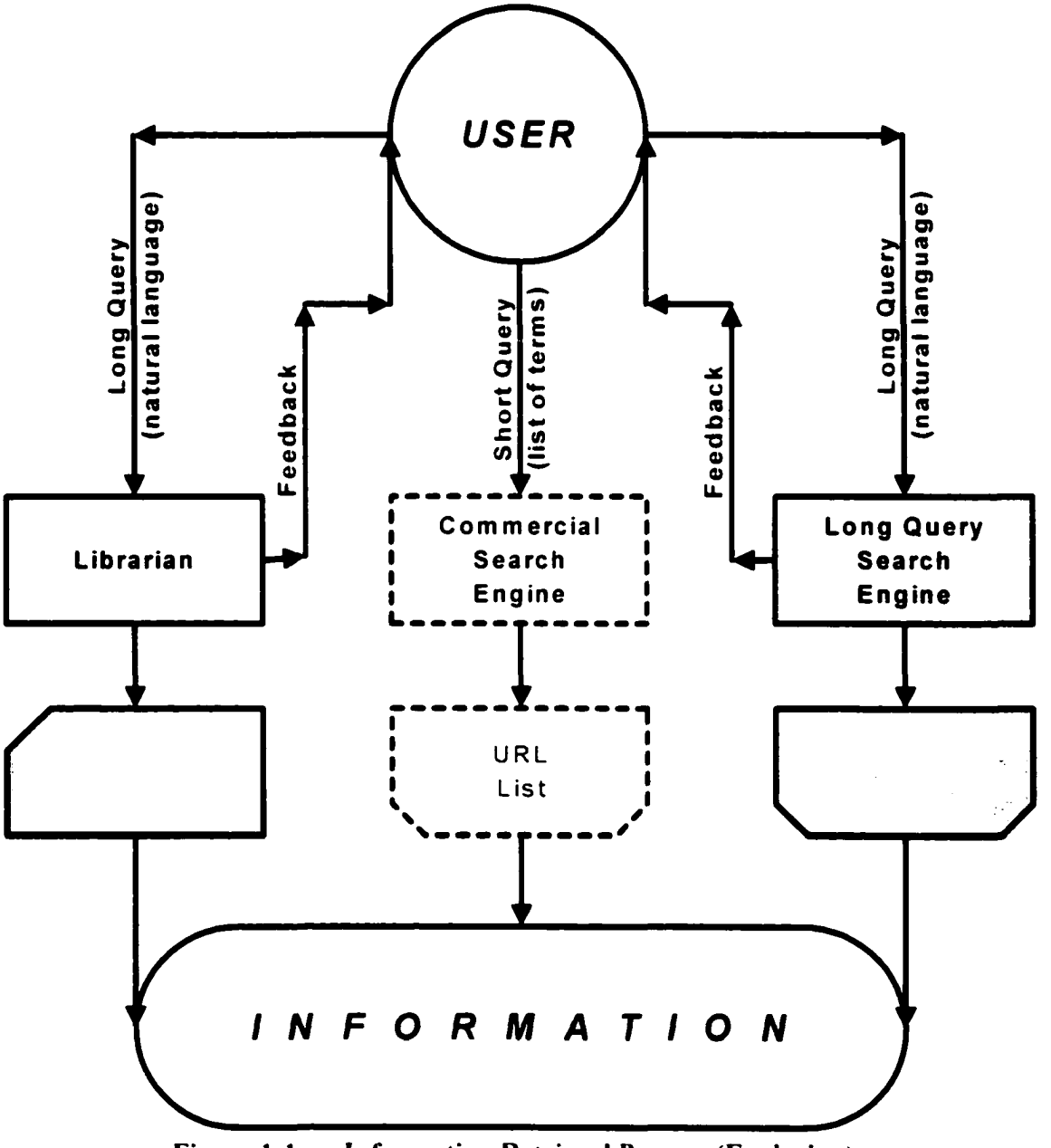


Figure 1-1 Information Retrieval Process (Evolution)

1.2 Satisfying Users' Information Needs

Many search engines provide advanced options which allow a user to enter complex (not longer) queries in a different format and put some restrictions on the output (domain, date, etc). However, even then, the user is not expected (and certainly is not encouraged) to enter more than a few terms, regardless of a user's information need (IN). But there are different types of information needs which lead to different ways of expressing these needs as search queries. For some types of information needs a query might be only a few words long, while other types of information needs will require much longer queries. User information needs can be divided into the following two classes:

i) Concrete Information Need (CIN). For example,

"How many countries are in the UN?";

ii) Problem Oriented Information Need (POIN). For example,

"How can we measure the performance of a search engine?".

The following table [FSV97], adapted for our discussion, describes the differences between the two types of needs.

Concrete Information Need	*Problem Oriented Information Need*
The boundaries of the information need are clearly defined because there is a clear answer.	The boundaries of the information need are not defined because there is no clear answer.
The request is put into exact words, that is, it corresponds exactly to the Concrete Information Need.	Generally, the request does not conform to the Problem Oriented Information Need.
To satisfy a Concrete Information Need, only one good document is needed.	As a rule, the Problem Oriented Information Need cannot be satisfied, even with all good documents existing in the system.
As soon as the good document is found, the Concrete Information Need is satisfied.	As soon as the good documents are delivered, the boundaries of the Problem Oriented Information Need may change and the need may remain unsatisfied despite the retrieval of many of the most appropriate documents available on the Web.

Table 1-1 Comparative Characteristics of Concrete Information Need (CIN) and Problem Oriented Information Need (POIN)

A Concrete Information Need query can usually be expressed by relatively few terms. For example, a query like "UN members" would be sufficient to answer, "How many countries are in the UN?" Such a query could be submitted to any of the existing search engines, for example, Google [G01], and it is quite likely that there will be a document containing the desired information among the "Top 10" documents.

However, a Problem Oriented Information Need query is more likely to start as a short query and then require the user to submit another query after reading some documents. The best technique is to retrieve some sites from this original set of results to find out how the topic is discussed in the documents. The next step would be for the user to formulate a second query based on the retrieved documents and submit it to the search engine. The easiest way for the user to formulate the second query would be to choose one or more of the most relevant documents retrieved in the original search, and then submit that entire document(s) or the relevant sections as a new (long) query.

1.3 Objectives of this Research

The goal of this thesis is to study long queries in the Web environment. We intend to show that long queries (expressed in natural language to enable the user to describe the information need in the manner easiest for the user) will enhance the quality of searches on the Web and improve users' satisfaction with the search results. We also intend to show the importance of the users' ability to refine their long queries by providing a feedback mechanism, which enables users to review the query results and then to alter the queries to obtain more precise results. We also want to demonstrate that with existing

technology it is possible to efficiently process long queries. To achieve these goals we took the following steps:

- Explored the current state of Information Retrieval on the Web

  This step included an examination of the literature and the proceedings of various conferences. We also investigated how existing commercial search engines process long queries (if at all) both in terms of their interface and search results.

- Created a user-friendly meta search engine to process long queries

  This step included developing of a meta-search engine that accepts and processes long queries. It provides a large box so that the user can see the query. It also provides clear instructions on how to enter a long query and how to enter an entire document as a query. If the user chooses to refine the query before running the search, the engine provides the user with a list of the words used in the query in order of their weights. If a common word appeared on top of the list, the user could see that the word was likely misspelled and could correct the error before running the search.

- Proposed, implemented and validated new algorithms

  The heart of this meta-search engine is a variety of new algorithms that provide more precise results than engines which use only short queries. First, we parsed the query and removed stop words and various symbols.

Then, we calculated the weights of the remaining terms and created a file of terms based on these weights. Second, we composed conjunctive subqueries from a variable number of query terms and phrases. Third, we submitted the composed subqueries to search engine(s) of the user's choice and applied one of our ranking algorithms to combine the intermediate results.

- Developed a logging facility

  The purpose of this facility was two-fold. First, we needed it to collect data to test, improve and validate new algorithms. All kinds of data were recorded while the search process was proceeding through various stages. Second, some of the data, which is available to the user, would assist the user in the search process. If the user was unsure about how best to phrase a query, the user could run several searches until he found the right words to use in a search. As the user learned more about the subject and attempted to narrow the search results, it was sometimes helpful to retrieve prior results. These results were saved in a log file.

1.4 Organization

This thesis is organized as follows:

Chapter 2 discusses previous work related to our research. It includes a description of the need for and the functionality of an interactive search

process, a description of why a long query is useful and how to create an environment to support the long query. It also includes a discussion on the length of a long query and phrase usage in a long query. It describes previous work on the multiple formulations of a query and merging of the search results into one ranked output.

Chapter 3 briefly discusses the Long Query Search Engine (LQSE). It provides a short overview of the design principles, user interface and the algorithms we used to build a prototype of a Web accessible meta-search engine.

Chapter 4 provides a detailed description of the user interface. This interface was specifically designed for a long query search, and allows the user to have complete control over all steps of the search process from search term selection to multiple query formulations and to the merging of search results.

Chapter 5 presents an extensive description of the novel algorithms introduced in this dissertation. These algorithms cover various aspects of long query processing, such as: parsing and analysis of the long query, creation and refinement of search queries, submission of multiple search queries to search engines, and the merging and ranking of search results.

Chapter 6 discusses the evaluation of the proposed algorithms. It describes the relevancy scale used by the participants in judging the relevancy of the search results. It also describes the performance measures used to evaluate information retrieval on the Web.

Chapter 7 provides a detailed description of the experiments. It states the objectives, classifies the individual tasks within the complete experiment structure, and defines the various elements of the experiment environment: library of long queries, search engines used for retrieval, and data collection and tabulation methods used to process experimental results. And finally it presents and discusses results of all experiments.

Chapter 8 concludes this thesis with a review of our contributions, a summary of the achieved results and directions for future research.

2 Previous and Related Work

2.1 Process of Information Retrieval

Many researchers describe information retrieval as a process rather than a search event. "Information retrieval systems must not only provide efficient retrieval, but must also support the user in describing a problem that s/he does not understand well. The process is not only one of providing good query language, but also one of supporting an iterative dialog model" [HB96]. Researchers have suggested a number of ways to provide the user with the support necessary to create the most effective query. Some demonstrate that query refinement greatly improves the efficiency of information retrieval [MB00, LS98]. Others suggest iterative algorithms for improving inter-document similarity measurements [A00] and user-driven (rather than search engine driven) document ranking and presentation algorithms [F99]. Others suggest providing the user with the option of recording and storing all the steps of the retrieval process (query formulation, refinement steps, interim and final results, etc.) [F98].

2.2 Creating and Using Long Queries

Many researchers report high quality results when long queries are used during the search process [VH97, GPS99]. A majority of new and encouraging results studying long queries come from the TREC (Text Retrieval Conference) community. Some TREC researchers experimented with longer queries on the IRIS (Information Retrieval Interactive System) project [SYAS97, YMMS98] and lately on the SDR (Spoken

Document Retrieval) track [GAV00]. While some researchers suggest that this topic should be investigated further [J00], there are others who feel that the SDR track should be postponed for at least another year since long query processing may mask degradation of SDR system performance [G00]. Some of the researchers believe that faster methods are more important than "feeding more of the topic description to the search engine" [HCH99].

While researchers agree on the need for a long query, they suggest different implementations. Some researchers believe that a larger query box is needed to induce users to articulate their information needs [SBC97, F98]. Some insist on natural language processing to handle longer queries [F99]. Others blame users for their inability to correctly and clearly express their information needs [PH97]. To assist the user with this process, some researchers indicate a need for a query "formulation agent" [RMMH00]. Others discuss existing tools that already automate query formulation through annotation-based queries [GPS99] or via InQuery that generates queries from TREC's topics [CCH92].

2.3 Using Phrases to Focus Long Queries

While discussing the need for a longer, better-formulated query, many researchers have investigated the importance of a phrase in a query [NO00]. Some indicate that the use of "phrasal search terms increase precision, since phrases tend to be definite concept makers." [LP99] Others discuss phrase recognition and expansion [LP99] and how various weighting schemes could produce better results [H92]. The authors of Phrasier

[JS99] report excellent results by automatically extracting key phrases from source documents and suggesting appropriate query phrases to users. At the same time, other researchers point to modest phrase usage benefits while using long queries [MBSC97]. Major commercial search engines accept and process user supplied phrases. AltaVista and Google both offer semi-automatic phrase searching [S01], with AltaVista checking potential phrases against its database of known phrases [S99]. Infoseek deconstructs long phrases into two-term phrases, stating that it is more efficient for the search process and as beneficial for finding highly relevant results [USPTO00].

2.4 Choosing the Number of Search Terms in a Query

The actual number of terms used in a query has been discussed by many researchers interested in different questions: 1) the influence of the number of terms on the quality of the search, 2) how many terms are needed to identify a specific document or to create robust hyperlinks. Some researchers who have experimented with long queries have concluded that using more then 20-30 terms per query would not improve the quality of the search process [H92]. Several researchers, investigating the concept of a "strong query" (defined as a query designed to identify a specific Web page), suggested in their reports [BB98], that a query consisting of 8 terms is a strong query. A year later, another group of researchers reported this number to be 10 terms per query [HHMN99] and later reduced it to 9 terms per query [HHMN00]. At the same time, a group of researchers working on robust hyperlinks as a solution to broken hyperlinks stated that

with certain assumptions. 5 terms are enough to uniquely identify a Web page [PW00[1], PW00[2]]. However, the terms must be chosen correctly in order to get the right results.

2.5 Query Formulations and Multiple Results Presentations

Some researchers have investigated concurrent usage of various representations of an original query to improve the quality of the search process [H90]. The idea behind this concept is quite simple – create intersecting instances of the original query (subqueries). Use these subqueries to conduct searches (using single or multiple search engines) and then combine all individual results creating one ranked list [PBRC00].

While investigating this concept researchers have found that the probability that a document is relevant is relative to the total number of occurrences (presentations) of this document in combined search results [SK88]. Further studies of this concept showed "that progressive combination of query formulations leads to progressively improving retrieval performance" [BCCC93].

The main issue associated with this concept is what techniques to use to combine multiple results (data fusion) of individual searches. Earlier researchers favored the similarity method (all search results were compared for similarity to the original query). Later researchers favored the ranking method [S98, L97, BKFS95].

3 Designing the Long Query Search Engine

To demonstrate the advantage of using long queries in the Web environment we created a meta-search engine that we will refer to as Long Query Search Engine or LQSE. Envisioned as a research tool it provided us with self-sufficient and computerized access to the public interfaces of several commercial search engines. We used it to conduct actual Web searches and to collect performance data for subsequent analysis.

3.1 Guiding Principles

Our blueprints for the design were theoretical and empirical results in the IR field discussed in Chapter 2 as well as "wish lists" expressed by users and researchers alike. While finding relevant information is the foremost goal of any search engine, we aimed at reaching this goal in a user-friendly way. We claim and prove later on in our experiments that the iterative search process, where the user can interact with the search engine by choosing functions and selecting operational parameters, helps make the search process more successful.

To give the user this capability we built a flexible user interface. This interface provides a large query window to accept a long query and allows the user through a variety of options to shape the formulation and refinement process of the search query, as well as the size of the query, at any time (before and or after the initial search). This is especially important since long queries allow better focusing via an improved selection of search terms and phrases.

Once the query is formulated the user may want to use different algorithms to conduct the search, so this interface also permits the user to select which search algorithm, or combination of algorithms, to use and to enter operational parameters to make these algorithms more efficient.

After the search is finished and the results are reviewed, the user may want to backtrack the search steps to find out what search terms were selected and how they were ordered, how the subqueries were formulated and what search engines were used to conduct actual searches, and what algorithms were used to merge individual search results. All this information is available in the detailed log.

A full description and implementation of the details of the user interface are provided in Chapter 4.

3.2 Building Blocks

The Long Query Search Engine uses a novel approach to long query processing. This approach consists of using one or more algorithm from each of three groups (see Figure 3-1 below).

The original long query is parsed and a number of search terms are selected. The process is accomplished via the Order Query Terms algorithm. Based on these terms and the original query text the Query Phrase Extractor algorithm identifies potential phrases to be used in the subsequent search process.

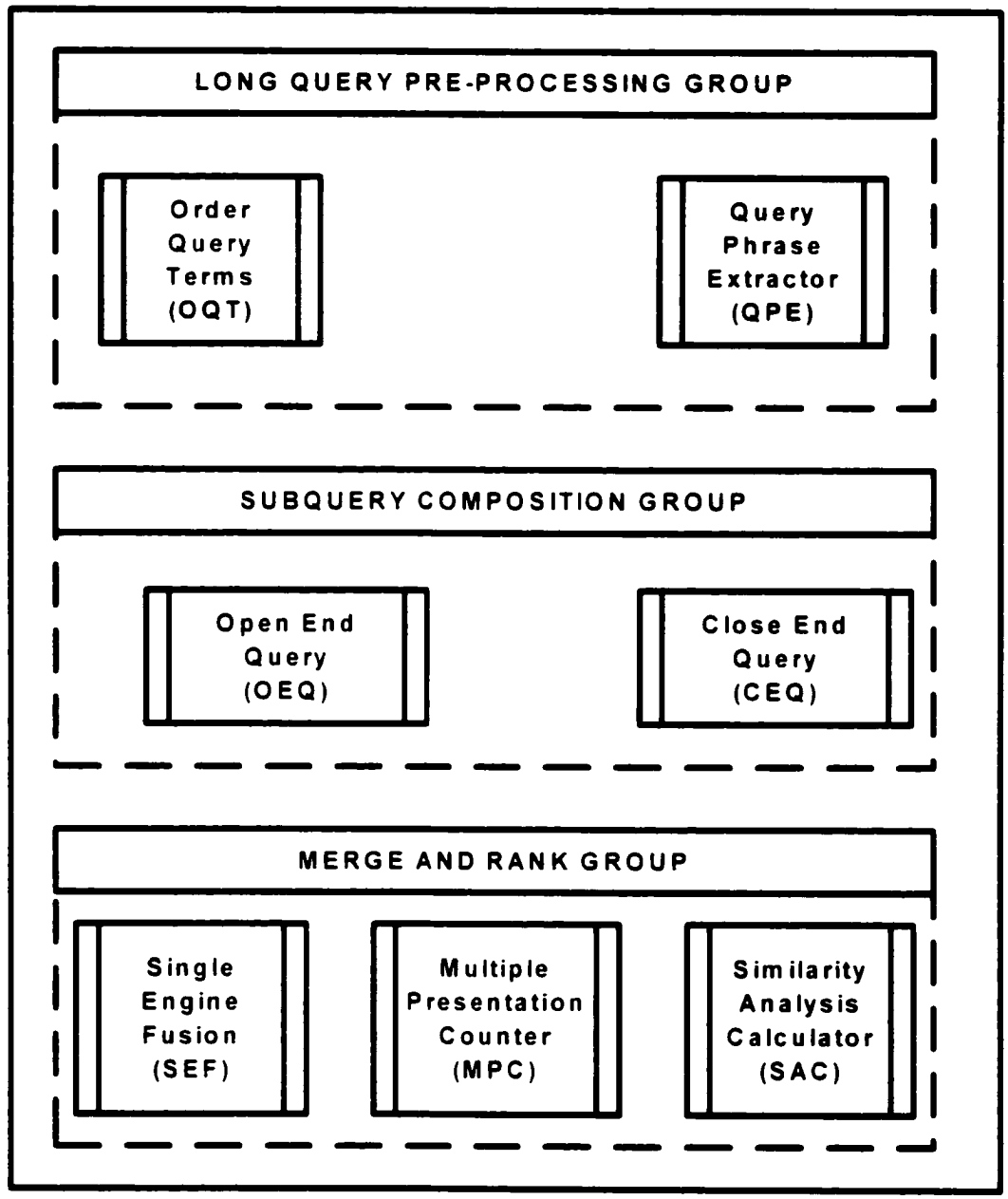


Figure 3-1 Long Query Processing Algorithms

Then either Open End or Close End algorithm from the second group uses the selected search terms and phrases to construct a series of fixed or variable length intersecting subqueries and submits them to commercial search engines.

Once the individual search results are returned. they are merged and ranked into a single output list. This task is accomplished by one of the following algorithms from the third group: Single Engine Fusion or Multiple Presentation Counter. To increase the efficiency of the Multiple Presentation Counter algorithm it is used in conjunction with the Similarity Analysis Calculator algorithm.

A full description and implementation details of the proposed algorithms are provided in Chapter 5.

4 User Interface and Feedback

4.1 Accepting the Long Query

The query can be composed by the user, be it a document inputted from a scanner, an audio clip processed by voice recognition software, a file read from a hard drive, the URL address of a Web site, or a copy of a Clipboard – anything that could be represented in an ASCII character set. Once the initial long query is composed or copied from another source (e.g. see Appendix B), it is entered in the large box (see Figure 4-1 below). There is no real limitation on the length of the document or its source.

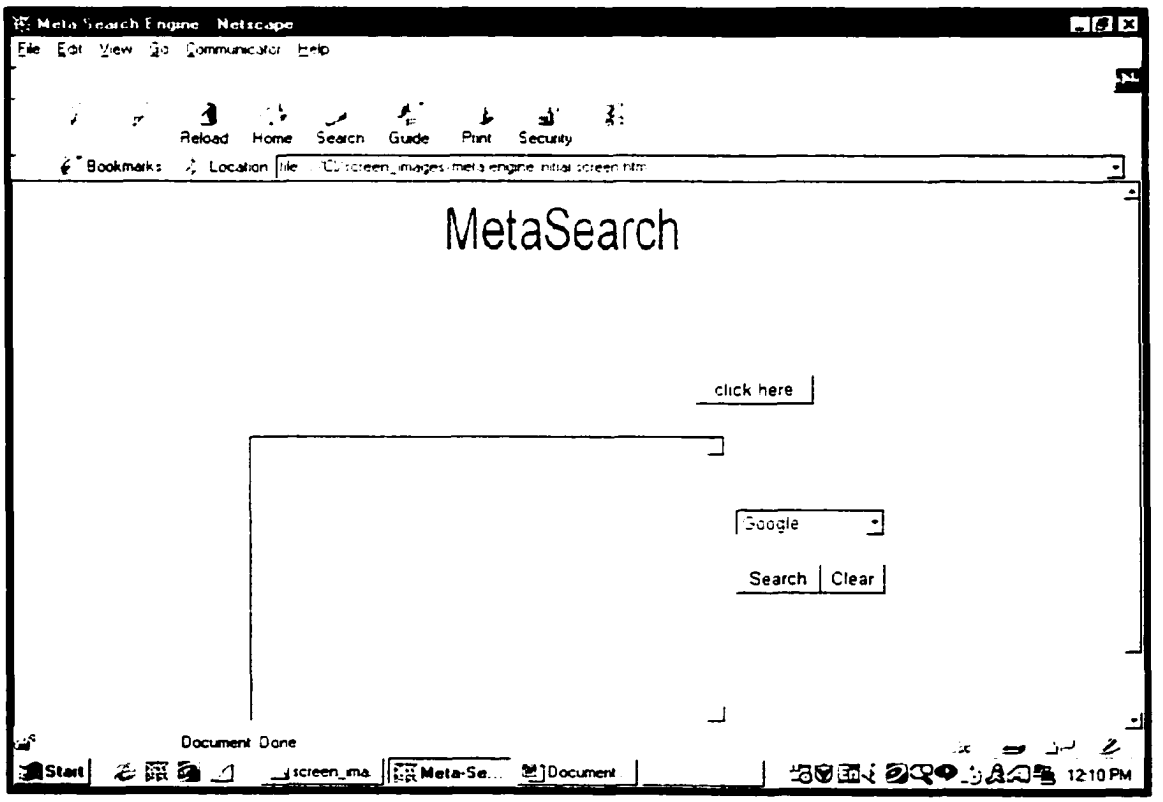


Figure 4-1 Long Query Search Engine (The Initial Screen to Enter a Long Query and to Select A Search Engine)

After the query is accepted, the Long Query Search Engine applies the processes described by the algorithms in the following chapter. The Figure 4-9 shows the steps of the process.

In Step-1 the engine accepts the long query and applies the Order Query Terms (OQT) algorithm to parse the query and organize and select query terms, and the Query Phrase Extractor (QPE) algorithm to extract potential phrases from the query text. Results of these processes are logged for user review (see Appendix C boxes 1 & 2).

4.2 Query Refinement Process

At this point the user decides whether to proceed with currently selected search terms or to review and refine them. A query refine screen (see Figure 4-2 below) provides the user with this capability. In Step-2 (see Figure 4-9), which could be repeated as many times as required, the user is given the option to examine and alter the order of query terms, add and/or delete phrases. This detour in the search process is a very important part of user feedback. It makes this engine a useful and powerful research and retrieval tool because it permits the user to easily rearrange the query terms.

For example, misspelled words and or typos of common words are easily identified because they "jump" to the top of the list of terms. The user then easily corrects the mistake before running the query and avoids a useless search with results skewed by misspelled words.

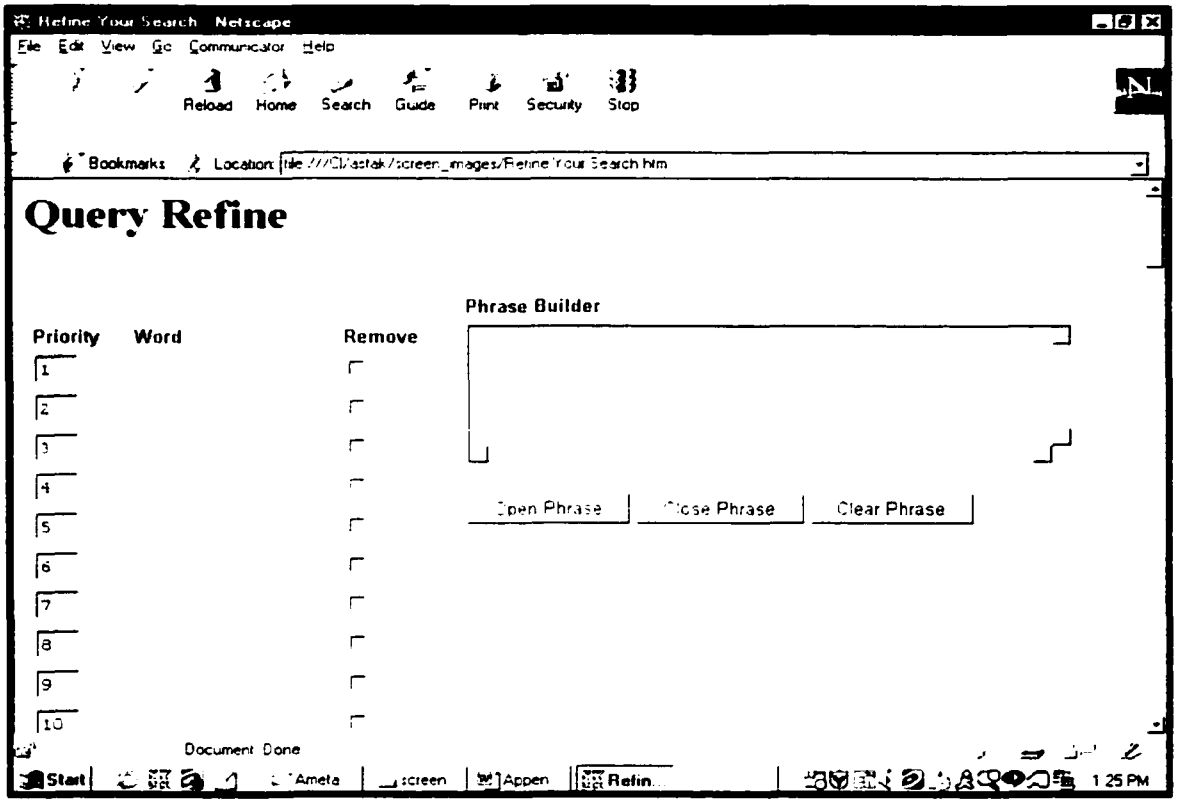


Figure 4-2 Long Query Search Engine (The Query Refine Screen with a Phrase Builder)

Rearrangement of query terms consequently causes variations in the compositions of subqueries and very often improvements of search results returned by search engines. In addition, the feedback option also allows the user to generate phrases that will better focus the search. After viewing the next search result page (see Figure 4-3 below) the user evaluates the quality of the search and whether to continue the query reformulation process or to select a different search engine, or to stop the search process altogether.

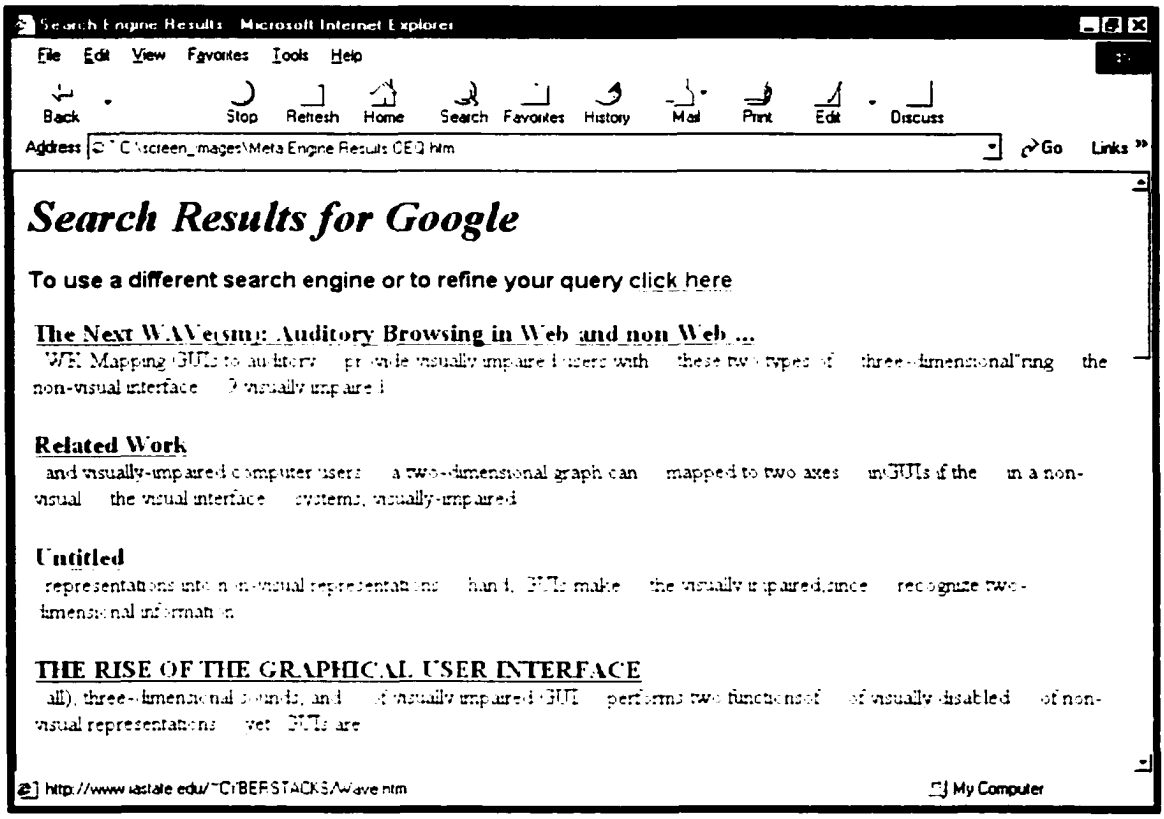


Figure 4-3 Long Query Search Engine (The Search Results Page)

4.3 Selecting Algorithms and Operational Parameters

In Step-3 (see Figure 4-9), the user decides which subquery formulation process to use and which parameters to select. If the user decides to proceed with the default Open End Query (OEQ) algorithm, then only one parameter is required – the number of variable length subqueries. (Once the search process is finished, its intermediate results including the recursive steps to produce the subqueries, the composition of subqueries and the number of search results, are stored in the log (see Appendix C box 3).

On the other hand. if the Close End Query (CEQ) algorithm is selected. then the user is required to set up the process by specifying the total number of search terms to use and the minimum size of the subquery to be created (see Figure 4-4 below).

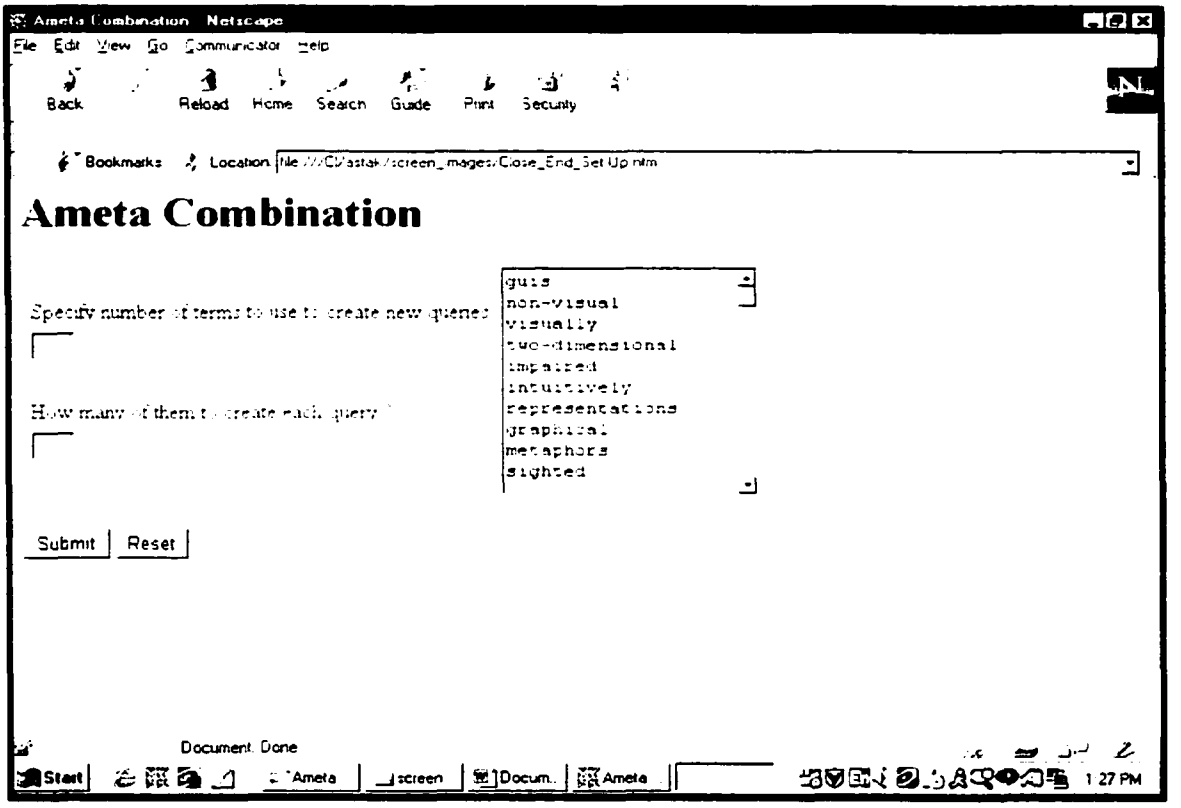


Figure 4-4 Long Query Search Engine (Selecting the Operational Parameters for the Close End Algorithm)

4.4 Submitting Subqueries to Search Engines and Merging Results

In Step-4 (see Figure 4-9). all subqueries created earlier are submitted via public interface to an existing commercial search engine or more than one search engine if the user chooses that option. All results returned from these individual submittals are merged into a single ranked output via one of the merge algorithms described in the next chapter.

While the search process is going on, the user can view intermediate results by "jumping" to various parts of the log (see Figure 4-5 below).

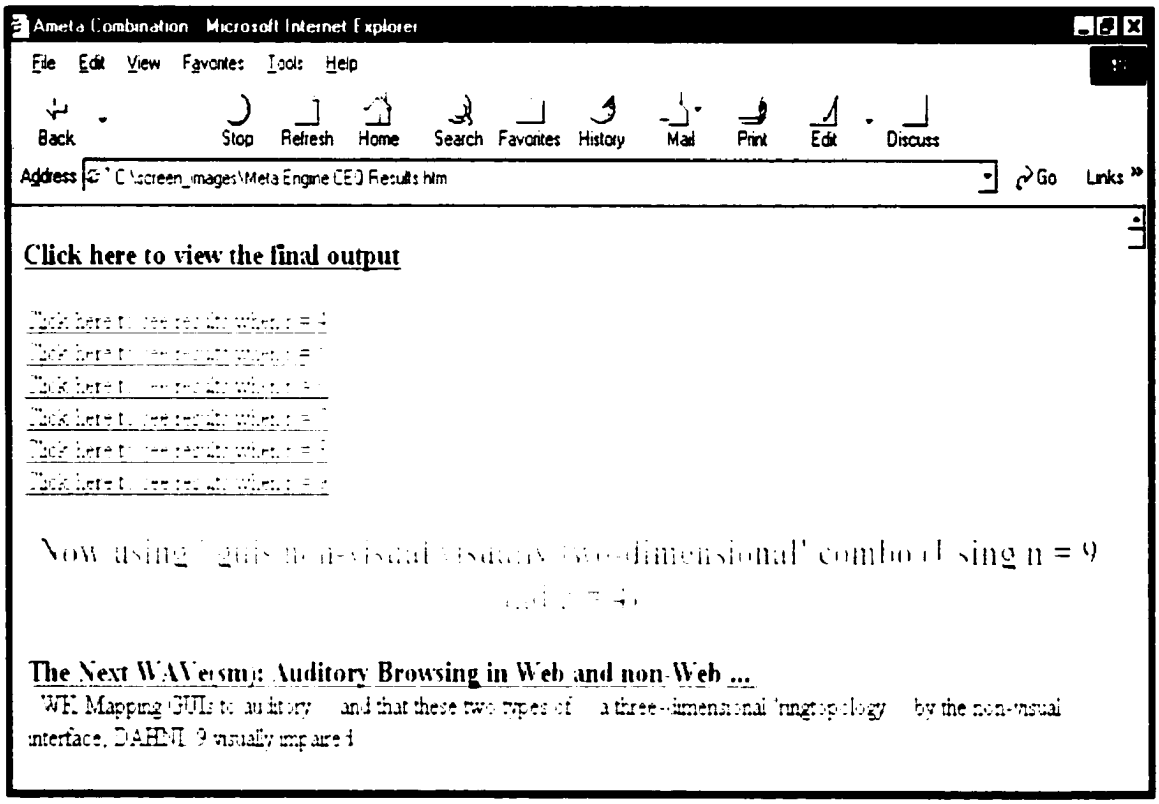


Figure 4-5 Long Query Search Engine (The Top of the Log – Links to Various Parts of the Log)

For example, if the user wants to see how the search engine is performing when the subquery consists of 7 terms, by clicking the link "*Click here to see results when r = 7*" (see Figure 4-5 above) the user can view the results of the actual search process in progress (see Figure 4-6 below). After the completion of the search process the user can view the final output ranked by one of the merge algorithms (see Figure 4-7 below).

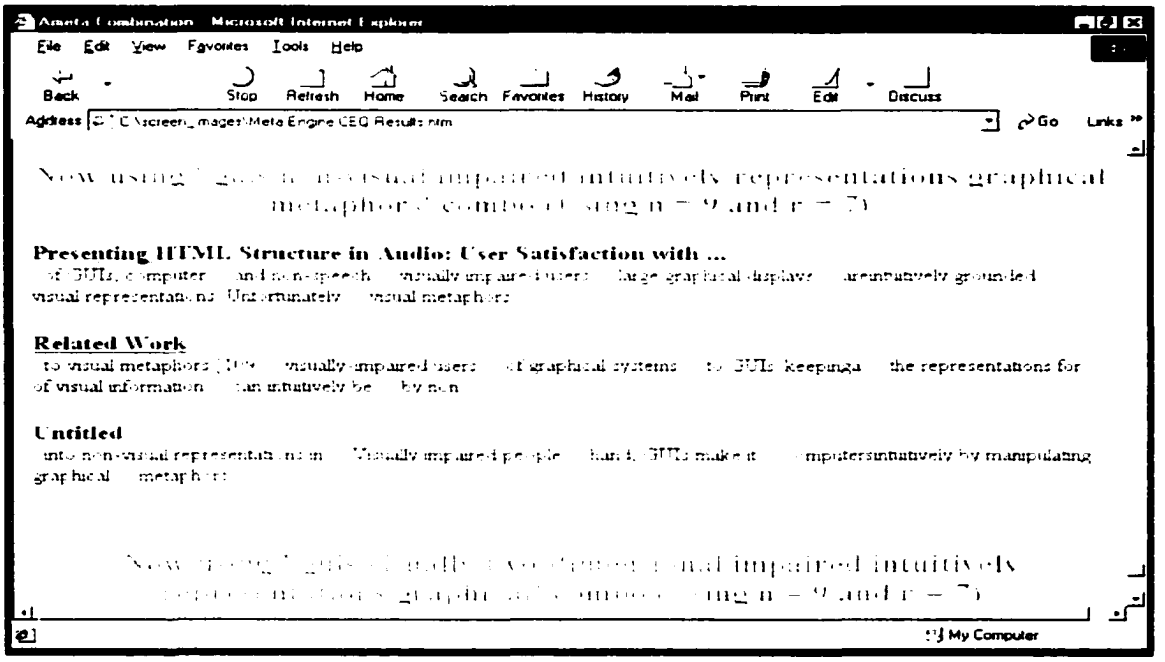


Figure 4-6 Long Query Search Engine (Individual Subquery Search Results))

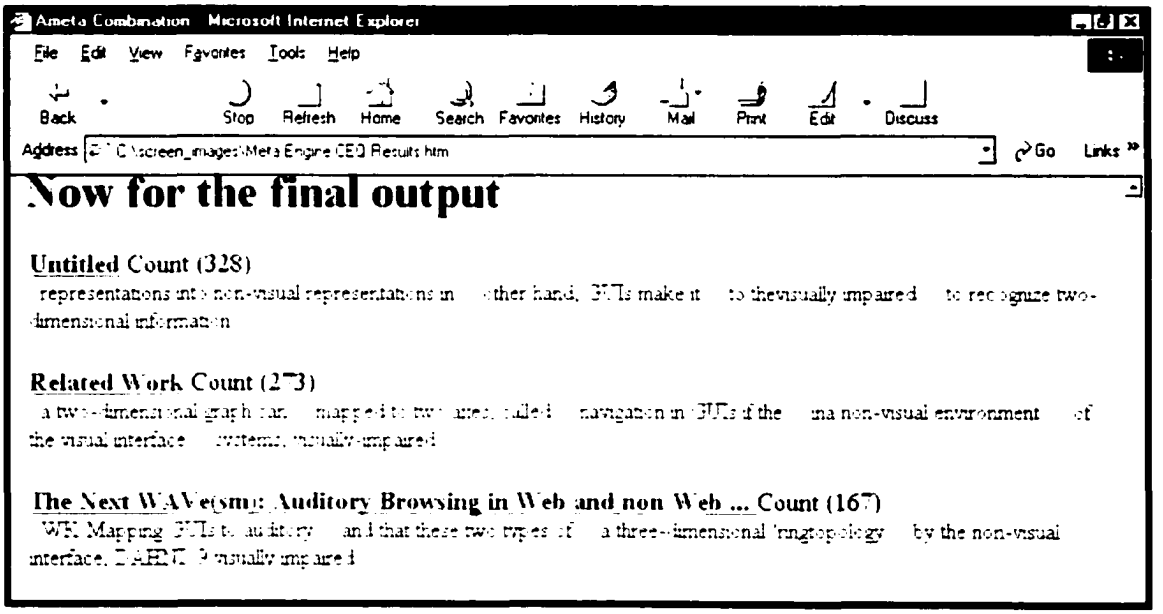


Figure 4-7 Long Query Search Engine (The Final Output)

4.5 Reviewing and Saving Search Results

In Step-5 (see Figure 4-9) the user reviews (in a separate window) Web sites returned by the search and stores interim results. As we pointed out in the Introduction, Problem Oriented Information Need queries, which are at the heart of this research, have a tendency to change, expand and re-focus their boundaries during the retrieval process. Therefore, the user decides whether to continue the search by query reformulation (Step-2) or to terminate the search. And lastly, if the user decides to rank the final output based on the similarity of the found documents to the original long query, the user can use the list of final URLs (see Figure 4-8 below) as an input to the Similarity Analysis Calculator (SAC) described in the next chapter.

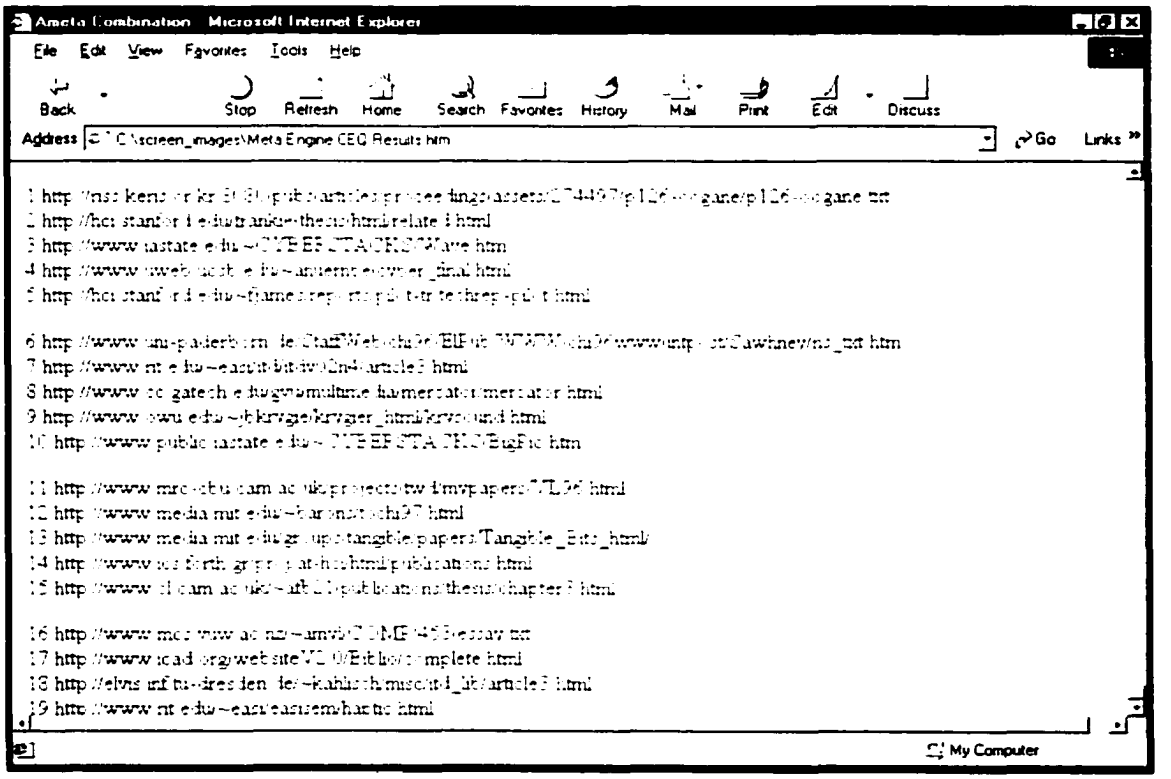


Figure 4-8 Long Query Search Engine (The Bottom of the Log – List of URLs)

4.6 Monitoring the Search Process

Finally, the logging facility which runs in the background, records all steps and states of the search process: terms of the query and their calculated weights, suggested phrases, algorithms selected, composed subqueries and search results.

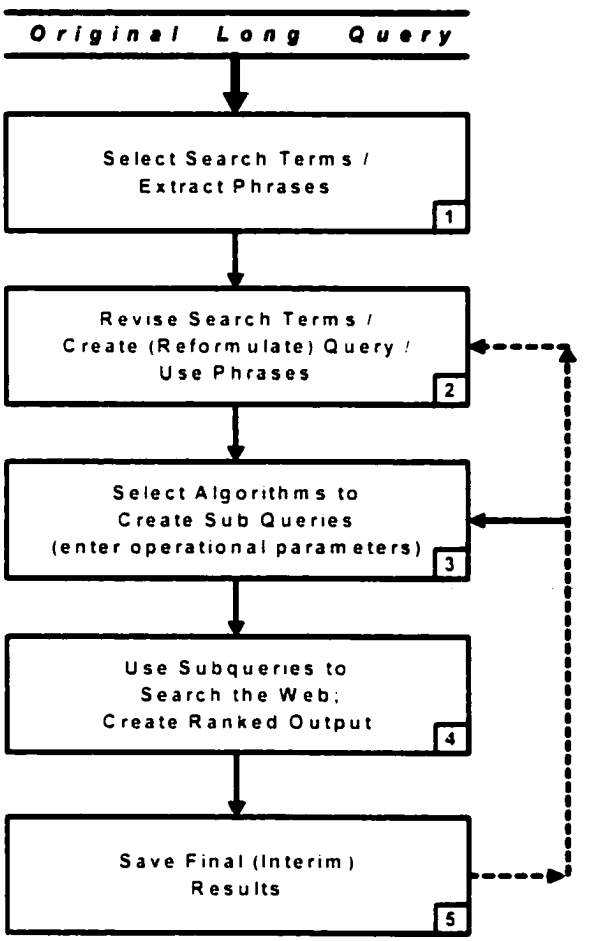


Figure 4-9 Long Query Search Process

This facility provided us a tremendous amount of performance data. Analysis of this data allowed us to fine-tune proposed algorithms, compare their individual performances and to determine optimal and effective combinations of algorithms.

5 Description of the Proposed Algorithms

This chapter provides a detailed description of the new algorithms proposed in this thesis (see Figure 5-1 below). There are three steps in the long query process:

- Long Query Pre-Processing Step (Step #1) consisting of two algorithms: Order Query Terms (OQT), which parses the long query to remove common words (sometimes called "stop" words), and some symbols, and then computes the weights of the remaining words from the original query; the second algorithm Query Phrase Extractor (QPE) uses these words and the original long query text to detect potential two-term phrases. Both algorithms are used as part of the search process.
- Subquery Composition Step (Step #2) consisting of two algorithms: Open End Query (OEQ) which uses a variable number of terms to compose a fixed number of subqueries (Step #2A) and Close End Query (CEQ) which uses a fixed number of terms to compose a variable number of subqueries (Step #2B). Only one algorithm at a time is used as part of the search process.
- Search Engine Merge and Rank Step (Step #3) consisting of three algorithms, which accept the composed subqueries, submit them to search engine(s) of user's choice and then merge multiple search results into a single ranked list. The first algorithm, Single Engine Fusion (SEF), uses one search engine and merges search results from subqueries based on ranking of interim search results obtained from composed subqueries and query terms weights in composed subqueries (Step #3A). The second algorithm, Multiple Presentation Counter (MPC), uses one or more search engines and merges search results based on a number of occurrences of identical results in a combined search output from composed subqueries (Step #3B). Only one algorithm at

a time is used as part of the search process. The third algorithm, Similarity Analysis Calculator (SAC), is used (if preferred) in conjunction with and to expedite the MPC algorithm. It retrieves the Web pages returned in intermediate searches and calculates their similarity to the original long query.

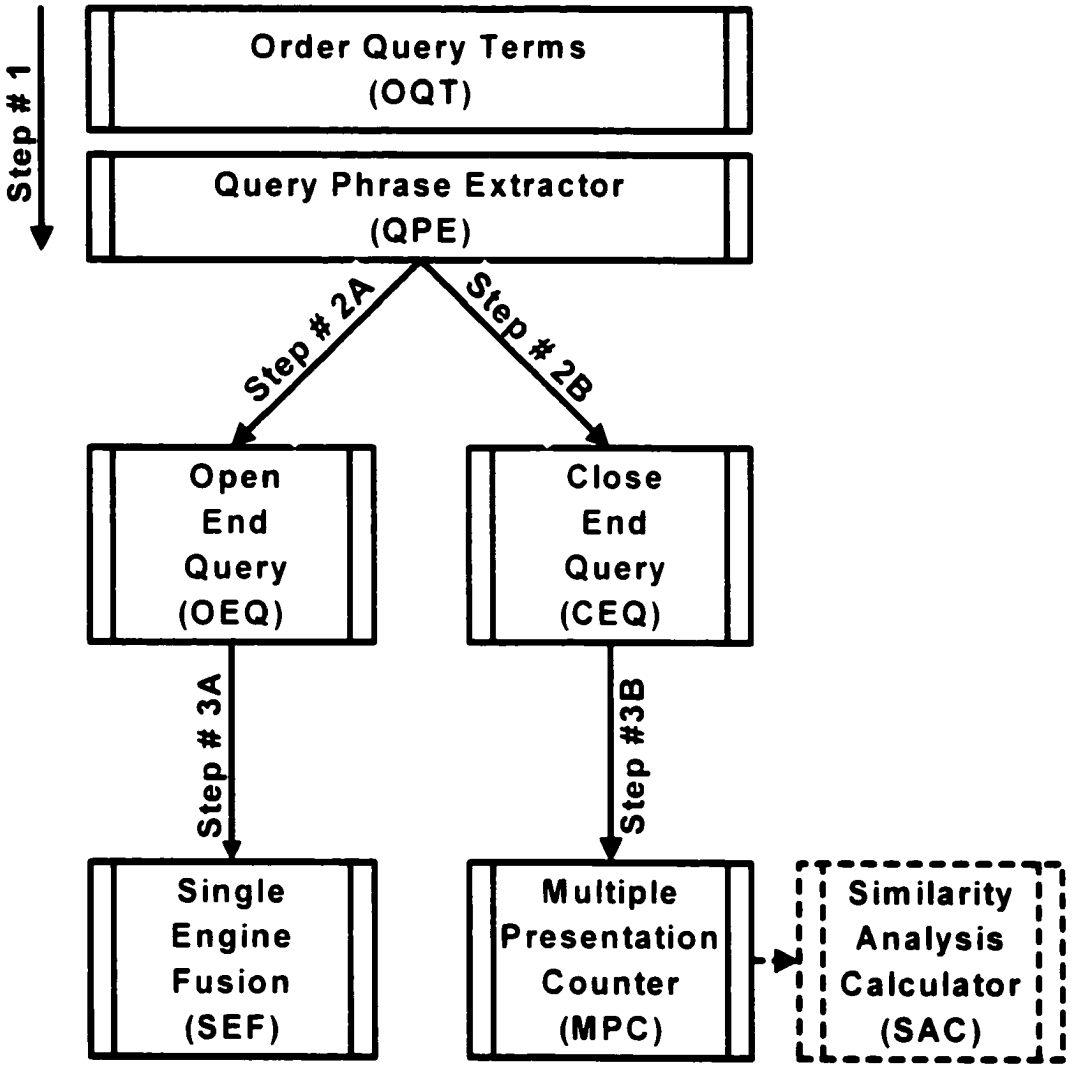


Figure 5-1 Long Query Processing Steps

The description of algorithms includes final values for various constants and tuning parameters. The process of selection and determination of these values we described in Chapter 7, Description of Experiments.

5.1 Order Query Terms (OQT) Algorithm

This algorithm starts by decomposing the original long query into separate words (terms) and removing "stop" words and various symbols (see Appendix D). Then for each remaining term, the term's weight is calculated using ***tf*idf*** measure. There are many variations of ***tf*** and ***idf*** measures and their combinations to obtain the term's weight. In an article [ZM98] the authors describe different choices, perform a series of experiments with these choices and come to a conclusion that the choice of different variations ***tf*idf*** measures and or their combinations have relatively little effect on the quality of search results when averaged over many queries. However, in the article [ZM98] the authors used a vector space model to compute similarity coefficients. We, on the other hand, use the weights to order the terms and on the basis of this order select subsets of terms to be submitted to search engines. So in our case the weights become more important in affecting the quality of the search results. For all of our experiments, we used the following formula to calculate weights of the remaining terms in the query after removing the stop words.

$$W(t_i) = C_1 \sqrt{f(t_i)} * \log_{(C_2)}(N / n_i)$$

where:

- $W(t_i)$ is the weight of the t_i term in the original long query
- t_i is the i^{th} term in the original long query (e.g. term *text*)
- $f(t_i)$ is the frequency of the t_i term in the original long query (e.g. term *text* occurred 4 times in the original long query)
- N is the size of the collection

(e.g. Google's collection is 1.4 Billion documents)

n_i is the frequency of the i^{th} term in the collection

(e.g. term *text* occurs 7 Million times in the Google collection)

C_1, C_2 is the tuning constants (after experiments set to $C_1 = 3$ and $C_2 = 10$)

When calculations are completed, all terms are sorted in descending order of their respective weights, and then stored (with their weights) in a Query Terms File (QTF). Other algorithms throughout the search process use data from this file extensively. Table 5-1 below is an example of a typical QTF data.

Term	*Weight*
pagerank	5.226945
devise	4.022191
approximating	3.583759
long-lost	3.403013
.....	.....
other	0.156778
information	0.086431
about	0.070043
page	0.023611

Table 5-1 Query Terms File Layout

Creating Local Data Base To compute the value of *idf* for a given term we need to know the frequency of this term on the Web (the number of documents on the Web containing this term). Since we do not do our own indexing of the Web pages we needed to obtain this number from search engines used in our experiments. However, to do this in real time would substantially slow the processing because it would require many extra

submissions to a search engine. To speed up this part of the query preprocessing we obtained the frequencies of a large number of the most popular words offline. We assembled a local Data Base of over 100K most popular words and submitted these words to a search engine to obtain their frequencies. There was a decision to be made - which search engine to use to obtain these frequencies. One possibility was to obtain these frequencies for all the search engines used in our experiments. This would have required a lot of extra processing and as we show below it was not necessary. Our main consideration in computing the term's weight is the effect of this weight on the ordering of terms in a query. So if we can show that the relative order of terms on the basis of their frequencies does not depend on a given engine we will be justified in using only one engine.

We decided to compare how randomly selected words would rank against each other on the basis of their frequencies in their respective collections. We selected 50 words from the entire local Data Base (every 2,000th word) and 50 words from all words with a frequency count under 1,000 (every 500th word out of 23,000 such words). Our experiments involved four search engines: AltaVista, Fast, Google, and Northern Light. We obtained the frequencies of selected terms for all of these search engines (all of them specify the number of documents where the term appeared). In addition, AltaVista also provides the total count of how many times the term appeared in the entire collection. This count by itself does not have a utility for us but the way it is obtained allowed us to substantially speed up the process of obtaining the desired frequencies and building a local Data Base.

If more than one term is submitted to any of the engines the engine returns only one number representing the number of documents where these terms appear (in some cases Boolean AND is used: e.g. Goggle. Fast. and in some cases Boolean OR is used: e.g. AltaVista). So submitting more than one word at a time would not allow us to obtain the desired frequencies. However. when several terms are submitted to AltaVista it provides a separate count for every term (the number of times the word appears in all the documents). We used this fact to submit 50 words at a time to AltaVista and obtained the count for every term in our Data Base.

Our objective was to show that the order of terms based on the counts from AltaVista is essentially the same as the order of words based on their respective frequencies in the collections from the other four search engines we chose. To accomplish this we measure the distance between two ranked lists using one of the standard distance measures - Spearman footrule distance [DKNS01]. The Spearman footrule distance is the sum over all elements i in a list of the absolute differences between the rank of i according to the two lists. For two full lists u and v (list size N) their Spearman footrule distance is calculated using the following formula:

$$F(u,v)=\sum_{i=1}^{N}|u_i - v_i|$$

We can divide this number either by N to obtain the average Spearman footrule distance or by $(N^2)/2$ to obtain the normalized Spearman footrule distance.

We performed these calculations to compare the order of various lists (constructed from terms frequencies for all four engines) to the order of list terms based on the term counts obtained from AltaVista. In nearly all cases the above distance

measure showed very closed orderings (most of the time the average is less than 1). See Appendix E for details.

5.2 Query Phrase Extractor (QPE) Algorithm

This algorithm uses the top query terms (key terms) in the QTF created previously. Then the original long query is parsed again to create Predecessor/Successor pairs (two-term phrases) for each of the key term. See Table 5-2 below.

Predecessor & Key Term Pair (frequency)	*Predecessor*	*Key Term*	*Successor*	*Successor & Key Term Pair (frequency)*
#1	*#2*	*#3*	*#4*	*#5*
17	search	*engine*	performs	5
		engine	definition	3
7	search	*engines*	Differ	1
3	quality	*indexed*	Pages	2
		indexed	Page	1
1	simple	*methodology*	approximating	1
1	engine	*methodology*	Based	1
1	defined	*pagerank*	ranking	1
3	broad	*queries*	Better	1
		ranking	applied	1

Table 5-2 Predecessor/Successor Pairs

The algorithm continues by calculating the weight of each created phrase. This calculation is based on the key term weight (calculated previously) and the number of occurrences of this phrase in the original long query (columns #1 and #5 above). The

frequency of the phrase is used to suggest to the user the potential importance of this phrase for the search process and the phrase weight is used in the Similarity Analysis Calculator (SAC) described later in this chapter.

5.3 Open End Query (OEQ) Algorithm

The algorithm consists of creating a fixed number of conjunctive subqueries from a variable number of query terms. It starts with picking the first query term in the QTF (it will be the 1st term for the first subquery, 2nd term for the second subquery, and so on). Then it recursively adds a number of consecutive query terms from the QTF and composes a conjunctive subquery from selected query terms. After each subquery modification (recursion) the new subquery is submitted to a search engine of the user's choice. If the number of results obtained from this search step is greater than 30, then the next query term from the QTF is added to the previously composed subquery and submitted to the search engine again. If the newly added query term causes "no results returned" the process discards this term and uses the next query term to compose a new subquery.

Creating Subquery (Example) The first subquery is composed from the first term from the QTF (step 1). See Table 5-3 below for a description of the steps. After submittal to the search engine (step 2) it returns ***645*** results. According to the algorithm the process uses (recursively) the subsequent three query terms (t_2, t_3, t_4) to compose the next subqueries.

	Terms in QTF	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	*Number of Results*
#	S T E P									
1	Start the Subquery #1 (use t_1)	t_1								
2	Submit to Search Engine									645
3	Modify Subquery #1 (add t_2)	t_1	t_2							
4	Submit to Search Engine									249
5	Modify Subquery #1 (add t_3)	t_1	t_2	t_3						
6	Submit to Search Engine									117
7	Modify Subquery #1 (add t_4)	t_1	t_2	t_3	t_4					
8	Submit to Search Engine									64
9	Modify Subquery #1 (add t_5)	t_1	t_2	t_3	t_4	t_5				
10	Submit to Search Engine									0
11	Modify Subquery#1 (remove t_5)									
12	Modify Subquery #1 (add t_6)	t_1	t_2	t_3	t_4	--	t_6			
13	Submit to Search Engine									35
14	Modify Subquery #1 (add t-)	t_1	t_2	t_3	t_4	--	t_6	t-		
15	Submit to Search Engine									*28*
16	Save the Subquery #1 results									

Table 5-3 Subquery Composition Recursive Process

When the newly created subquery is submitted (step 8) to a search engine, it returns 64 results – more than 30. So the process adds the next term $\mathbf{t_5}$ to the subquery (step 9) and submits it to the search engine (step 10). This subquery submittal returns 0 results and causes the removal of the term $\mathbf{t_5}$ (step 11) and the addition of the term $\mathbf{t_6}$ (step 12). The new subquery returns 35 results (step 13), so the term **t-** is added (step 14). Finally less than 30 results are returned (step 15) – subquery #1 is finished. The process saves the results returned from the last search for future use.

Creating Open End Query (Example) Using the above-described process, four subqueries were created, and submitted to the same search engine. Intersections of subqueries cause the search engine to return multiple presentations of the same document, and we use this fact to rank our final output. Results from searches based on these queries are merged according to algorithms described later in this chapter. The Figure 5-2 below demonstrates the typical makeup of the Open End Query.

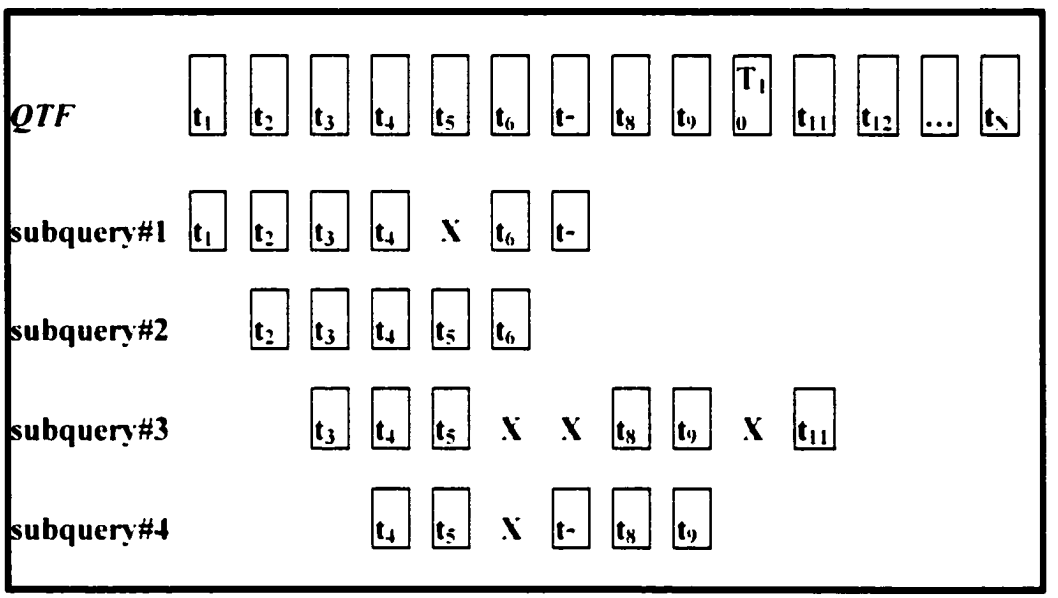


Figure 5-2 Open End Query Composition

5.4 Close End Query (CEQ) Algorithm

This algorithm consists of composing a variable number of conjunctive subqueries from a fixed number of query terms from the Query Terms File. For example, if we choose the top M query terms from the QTF the algorithm would create conjunctive subqueries consisting of all possible combinations of at least (L) terms (where $L \leq M$).

The total number of possible subqueries is $\sum_{i=L}^{M} C(M,i)$ where

$$C(M,i) = \frac{M!}{(M-i)! * i!}.$$

All composed subqueries are then submitted to the search engines of the user's choice and the results from all submissions are merged according to algorithms described later in this chapter. Our experiments determined that the best value for **M** and **L** are **7 ≤ M ≤ 9** and **3 ≤ L ≤ 6**. For example, if the user decides to have M = 8 and L = 6 then the algorithm will create all possible 6-term combinations out of the top 8 terms (28 combinations), and all possible 7-term combinations out of the top 8 terms (8 combinations) and one 8-term combination out of the first 8 terms (see Table 5-3 below). Then all 37 subqueries will be submitted to a search engine(s) and all results are merged into one ranked list.

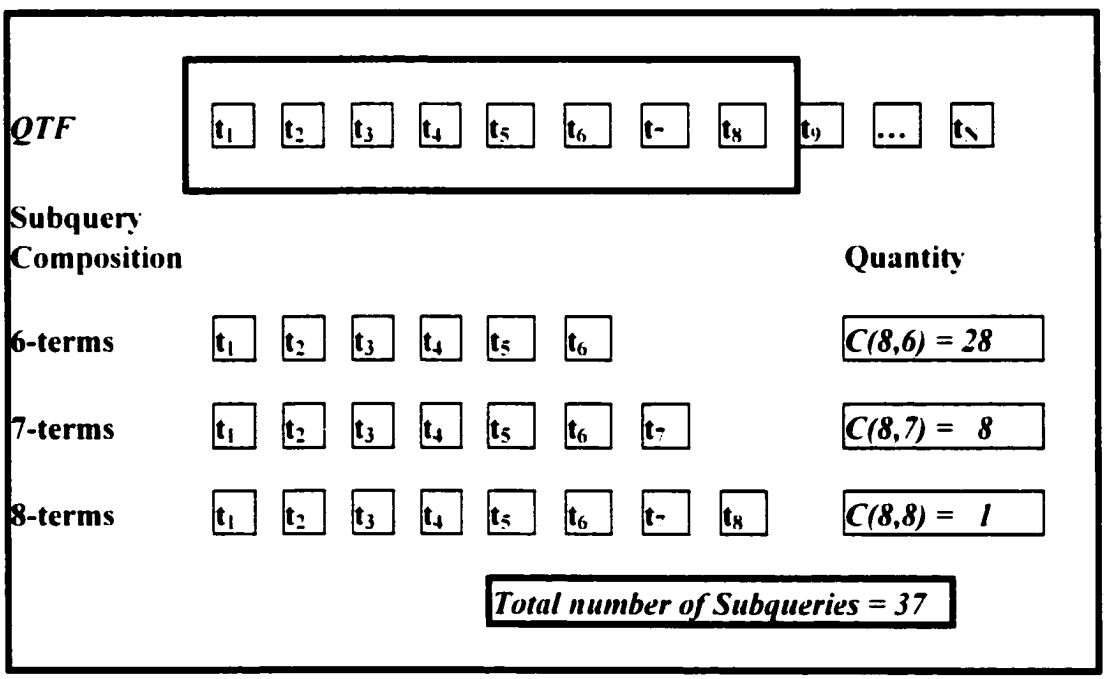


Figure 5-3 Close End Query Composition

5.5 Single-Engine Fusion (SEF) Algorithm

After all subqueries are submitted and search results are returned, this algorithm merges the results from intermediate searches into one ranked list. The merging of intermediate results is illustrated in Figure 5-4 below. For example, after submitting **N** subqueries (SQ_1, SQ_2, . SQ_J,. . SQ_N) the following **N** ranked lists (R_1, R_2, . R_J,. R_N) were returned. Each of the lists contains a different number of distinct ***URLs*** ranked by the search engine. The SEF algorithm uses a predefined constant M (in our experiments the M was set to 30) to limit the number of URLs on each list. More formally, if we denote by M_j the number of URL's used by SEF from list R_j then $M_j = \min(M, |R_j|)$ where $|R_j|$ is the number of URL's on the R_j list.

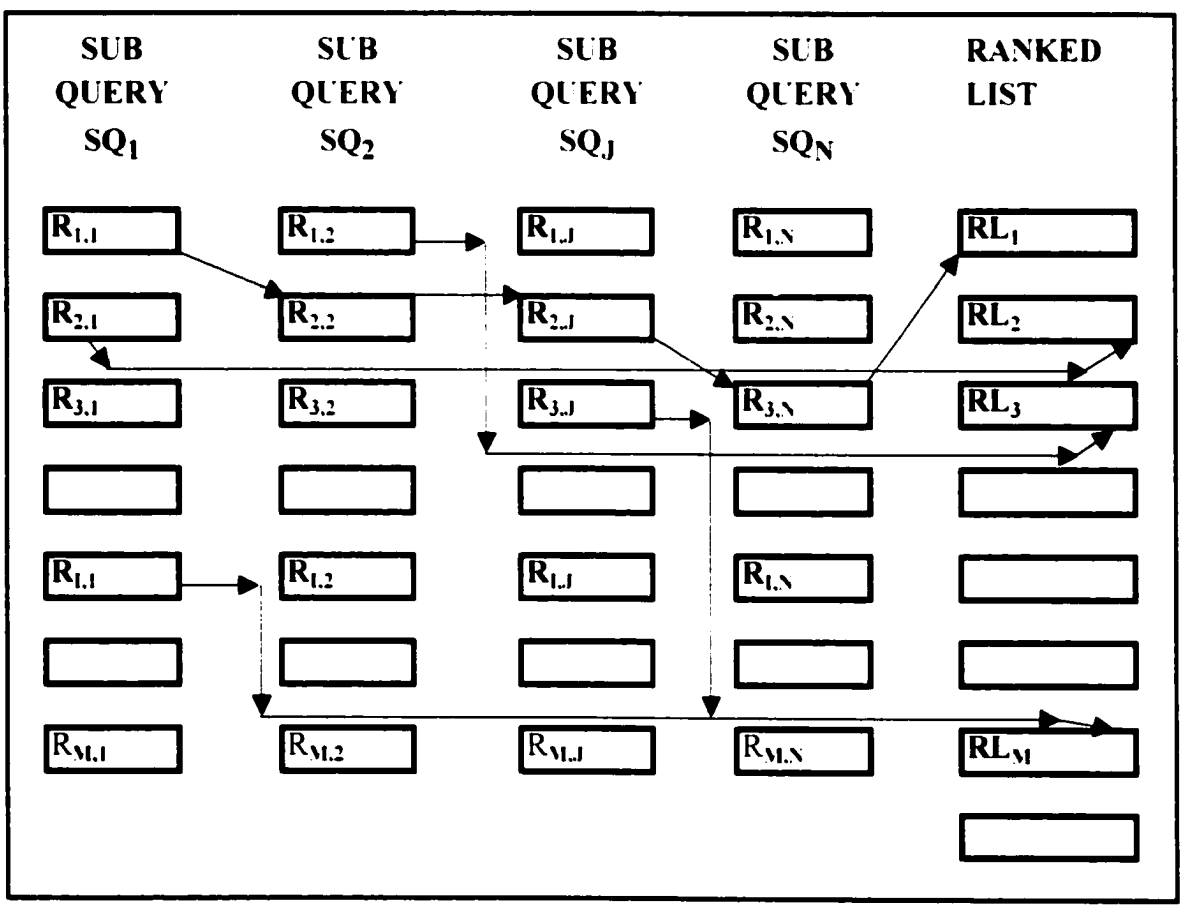


Figure 5-4 Merging of Intermediate Search Results

Then the algorithm proceeds as follows. First, it calculates the weight of each URL $R_{i,j}$ (within the output R_j) using the relative position i ($1 \le i \le M_j$) of that URL from the top of the output R_j and the weight of the subquery SQ_j that produced this output. The following formula is used to calculate the weight:

$$W(R_{i,j}) = W(SQ_j) * (M_j - i + 1)$$

where:

$W(SQ_j)$ (the weight of subquery SQ_j) is calculated as follows:

$$W(SQ_j) = \Sigma (Wt_{i,j})/K_j$$

$Wt_{i,j}$ is the weight of the term i in subquery SQ_j (from QTF)

K_j is the number of terms in subquery SQ_j

The next step of the algorithm is to find URLs which have duplicates in different outputs, remove all but one copy of such URLs and assign a revised weight to each remaining copy. The revised weight of a URL, which has duplicates in several outputs, is the sum of the weights of all the copies. The final step in the algorithm is to combine all the remaining URLs (notice that they are all distinct) into one list and rank the list in descending order of the URLs' weights.

5.6 Multiple Presentation Counter (MPC) Algorithm

This algorithm is designed to merge intermediate search results where the number of lists is relatively large and the subqueries on which these lists are based have little

overlap. In this case the primary consideration for the final ranked order of a URL is how many lists contain this URL. For example, after submitting **N** subqueries (***SQ*$_1$, *SQ*$_2$, . *SQ*$_j$, . *SQ*$_N$**) the following **N** ranked lists output results (***R*$_1$, *R*$_2$, . *R*$_j$, *R*$_N$**) were returned. The number of URL's on each list is chosen as above, i.e. M_j = min (M, length (R_j)). Then the algorithm combines all URLs into one unordered list, which contains at most **(30 * N)** individual URLs. The next step of the algorithm is to sort all URLs in an alphabetical order and count the number of times each URL is present in the interim file. While counting, all duplicate URLs are eliminated and one remaining copy is assigned a weight which is equal to the number of times this URL appeared in the combined list. Finally, the algorithm sorts the remaining URLs in the descending order of their weights. If several URLs have the same weight, then the URL which came from a subquery with the largest total weight of their terms is ranked the highest.

5.7 Similarity Analysis Calculator (SAC) Algorithm

This algorithm accepts a list of URLs, retrieves every Web page referred to by the URL in this list and calculates the similarity coefficient between this page and the original long query. We discuss this algorithm (see Figure 5-5) in more detail below.

The Order Query Terms (OQT) algorithm (process 1) parses the long query and creates TERMS (see Figure 5-5). The Query Phrase Extractor (QPE) algorithm (process 2) uses the original long query and the search terms from the QTF to create PHRASES (see Figure 5-5). The Close End Query (CEQ) algorithm (process 3) creates fixed length subqueries (SQ_1, SQ_2...SQ_N) from TERMS and PHRASES. One or more search engines

(process 4) process these subqueries and return search results for each processed subquery.

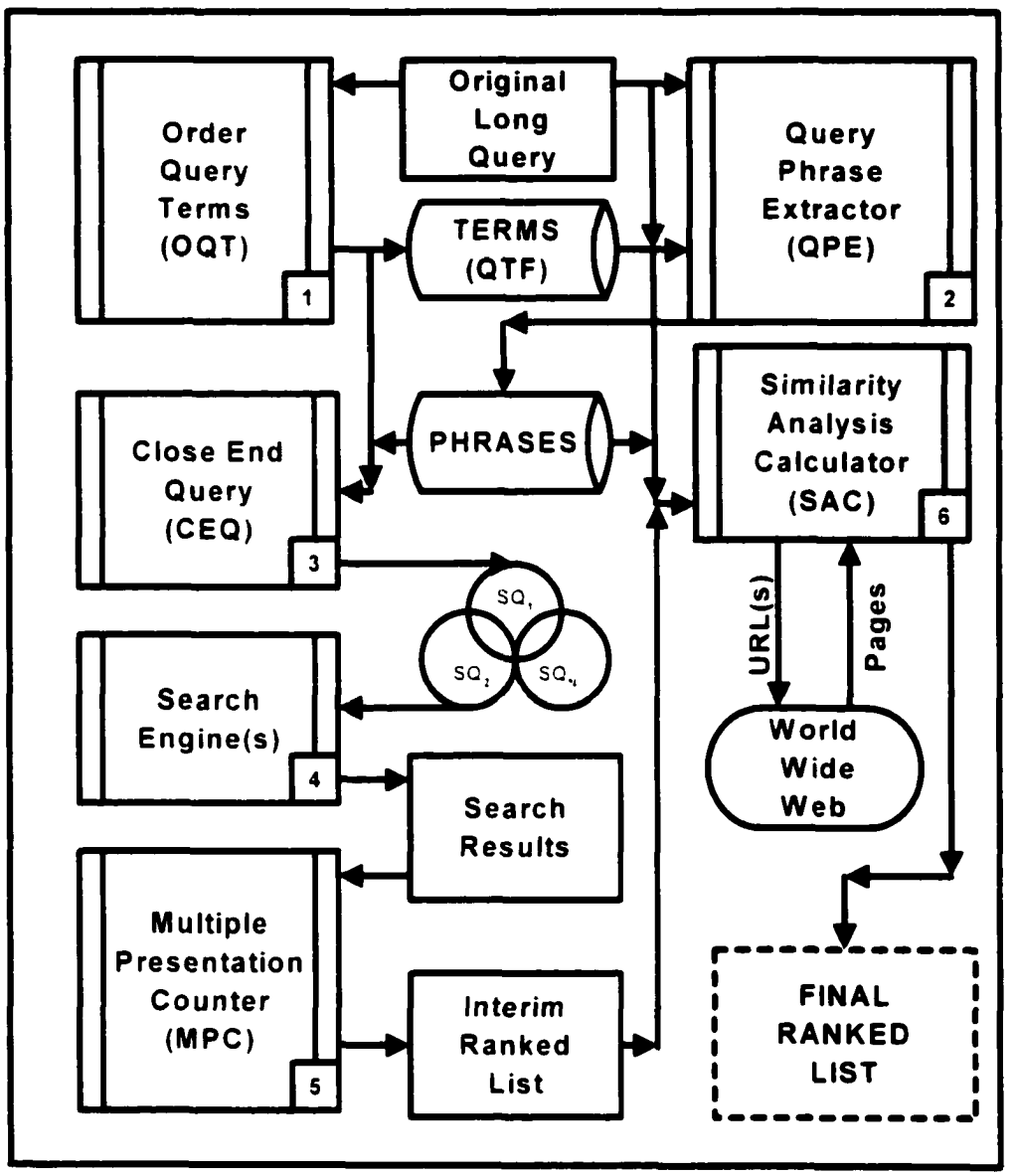


Figure 5-5 Using SAC Algorithm

The Multiple Presentation Counter (MPC) algorithm (process 5) merges the search results from all subqueries into an interim ranked list. The Similarity Analysis Calculator (SAC) algorithm (process 6) uses this interim ranked list of URL(s) to retrieve corresponding Web pages. First, it displays the current values of the operational parameters and then asks the user to input number of keywords that will be used for calculations (see Appendix N Step 1). Then all URL(s) specified in the input (see Appendix O Table 1) are retrieved (see Appendix N Step 2). The SAC algorithm then uses the original long query, TERMS and PHRASES to calculate the similarity coefficient (see Appendix N Step 3) of each retrieved page via the formula described below. After completing all calculation the original interim list is ordered in descending order of the similarity coefficients creating the final ranked list (see Appendix O Table 2).

The following is a formal description of the computations used in the SAC algorithm.

Input:

Original Long Query $\boldsymbol{q}$ – sequence of terms $< t_1, t_2, \ldots t_n >$. Denote by $\boldsymbol{Q}$ the set of all terms appearing in q, i.e. $t \in Q$ if $\exists\, i\ (1 \le i \le n)$ such that $t = t_i$.

Set of terms $\boldsymbol{T}$ where $T \subseteq Q$ and the corresponding set $\boldsymbol{T_w} = \{<t, w_{q,t}>\}$ where $w_{q,t}$ is a real number describing the weight of the term t in query q.

Set of phrases $U = \{ u_1, u_2, \ldots u_k \}$ where each phrase is a sequence of selected terms from Q, and a corresponding set $U_w = \{<u, w_{q,u}>\}$ where $w_{q,u}$ is a real number describing the weight of u in q.

Set of documents $D = \{ d_1, d_2, \ldots, d_N \}$ addressed by URLs

Output: For each $d \in D$, ***SIM (q, d)*** – a real number between 0 and 1

Computation of *SIM (q, d).*

Step 1 For each $t \in Q$ create a set of phrases (a phrase is a tupple of the form $< x, y >$, where x, y are terms) , denoted by V(t), as follows.

Let i be an index where $t = t_i$ Then

If i = 1 then $<t_1,t_2> \in V(t)$ (successor phrase is in V(t))

If i = n then $<t_{n-1},t_n> \in V(t)$ (predecessor phrase is in V(t))

If $1 < i < n$ then $<t_i,t_{i-1}> \in V(t)$ and $<t_i,t_{i-1}> \in V(t)$ both successor and predecessor phrases are in V(t))

Step 2 For every $t \in T$ compute $W_{d,t}$ (weight of term ***t*** in document ***d)***

$$W_{d,t} = 1 + \log_e(1 + f_{d,t}) + C1 * \log_e(1 + C3 * \sum_{v \in V(t)} f_{d,v})$$

where: $f_{d,t}$ - number of occurrences of term ***t*** in document ***d***

$f_{d,v}$ - number of occurrences of generated phrase ***v*** in document ***d***

C_1, C_3 - tuning constants

Step 3 For every u ∈ U compute $W_{d,u}$ (weight of the given phrase ***u*** in document ***d***)

$$Wd,u = 1 + C2 * \log_e(1 + fd,u)$$

where: $f_{d,u}$ - number of occurrences of ***u*** in ***d***

C_2 - tuning constant

Step 4 Final formula. $SIM\ (q, d) =$

$$= K \frac{\sum_{t \in q}(wq,t * wd,t) + \sum_{u \in U}(wq,u * wd,u)}{\sqrt{\sum_{t \in q}(wq,t)^2 + \sum_{u \in U}(wq,u)^2} * \sqrt{\sum_{t \in q}(wd,t)^2 + \sum_{u \in U}(wd,u)^2}}$$

where:

$$K = \frac{1}{\log_{C_4}|d|}$$

|d| - length of document d (number of characters)

C_4 - tuning constant

After conducting a series of experiments we established the following values for the tuning constants:

$C_1 = 0.3$ $C_2 = 0.2$ $C_3 = 2$ $C_4 = 10$

6 Evaluating the Proposed Algorithms

Evaluation of information retrieval (IR) on the Web is much more challenging than evaluation of traditional IR systems. The high intensity of growth and change, the variety and size of indexing vocabularies, and the increased usage of in/outbound links to rank search results make the task of measuring and evaluating retrieval performance a monumental task [BV00]. Lack of foreknowledge of collection content, topic coverage, size and relevancy, and the subjectivity of users while creating queries and evaluating relevancy of search results, makes the use of existing IR evaluation tools difficult and challenging.

In this section we describe several retrieval performance measures we have adopted and used to evaluate and compare various search engines (including the ones we developed for this thesis).

6.1 Judging Relevancy of Search Results

Relevance judgment is a subjective concept, quite often based more on a whim than a thought process. We found out in our early experiments, that asking participants for a binary relevance judgment invites superficial consideration of the document (reading the title or summary only, without actually visiting and analyzing the content of the site) and skews the experimental results. To lessen the inconsistency of such judgment, we introduced a ternary relevance judgment scale to force the participants to go beyond the title or the summary provided by the search engine. The following scale was used to judge the relevancy of the search results:

Relevancy	*Description*
0	**Not relevant** (different topics, no common terms, includes references to missing or restricted access sites)
1	**Somewhat relevant** (mentions same topic as the query, uses common terms, has links to sites deemed relevant by prior searches)
2	**Very relevant** (original document, describes same or very similar topics, closely matches the query, uses query terms extensively, has many links to sites deemed very relevant by prior searches)

Table 6-1 Ternary Relevancy Scale

Participants were asked to evaluate the "Top 20" search results only. This is a time consuming task and grounds for this limit are results of a study [X99] reporting that 58% of users view the "Top 10" results and 70% view the "Top 20" results.

6.2 Position of the Original Document

Since most long queries are excerpts from some existing Web pages, it would be beneficial to see if a search engine would find the original page from which the long query was extracted. If the engine finds the document then it is important to analyze where the search engine places it among its "Top 20" results. We use a table (see Appendix L) to accumulate results for four control intervals (Top, Top 5, Top 10, Top

20). Users almost always look at the first document (Top). They may want to see the first window with search results (Top 5) or investigate the first result page (Top 10), and quite often the second page (Top 20) as well.

6.3 Precision@10 & Precision@20

Precision is a fairly popular measurement of retrieval performance and is calculated as a fraction of retrieved documents.

$$P@N = \frac{|retrieved\_relevant\_results|}{\min(N, |retrieved\_results|)}$$

where:

$P@N$ - precision at "Top N" results

'retrieved_relevant_results' - number of relevant results among documents specified in the denominator;

retrieved_results - number of retrieved results.

We conducted experiments for ***N = 10 & 20.***

We used the table (see Appendix M for a sample) to accumulate Precision@10 and Precision@20 results for all engines.

6.4 Recall and Relative Recall

The other widely used measurement is recall, which is calculated as

$$Recall = \frac{|retrieved\_relevant\_results|}{|collection\_relevant\_results|}$$

where:

|retrieved_relevant_results| - number of retrieved relevant results;

|collection_relevant_results| - number of relevant results in the entire collection.

It is clear that collections used in our experiments (documents indexed by Google, AltaVista, etc.) are enormous, and it is quite difficult, if not impossible to measure the recall of each Search Engine. For our experiments and evaluation purposes we will be dealing with relative recall (R-Recall), which is helpful for the comparison of various search engines when applied to the same query, and is calculated as follows:

$$R\text{-}Recall\ \{SEi(q)\} = \frac{|relevant\{SEi(q)\}|}{\sum_{i=1}^{All} |\bigcup relevant\{SEi(q)\}|}$$

where:

$R\text{-}Recall\ \{SEi(q)\}$ - relative recall for

Search Engine(i) processing query q

$|relevant\{SEi(q)\}|$ - number of relevant results retrieved by

Search Engine(i) processing query q

$\sum_{i=1}^{All} |\bigcup relevant\{SEi(q)\}|$ - size of the collection of distinct relevant results

retrieved by ALL Search Engines processing query q

6.5 Precision at 11 Standard Recall Levels

This measure is based on calculating precision versus recall for each individual query based on 11 standard recall levels which are 0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 100%. Generally, search engines are evaluated by performing numerous searches for various queries. In such a case, all individual precision-recall curves are produced and an average precision for each recall level is calculated according to a formula we adopted from one of the latest manuscripts on the subject [BR99].

$$\bar{P}(r) = \sum_{i=1}^{Nq} \frac{P_i(r)}{N_q}$$

where $\bar{P}(r)$ is the average precision at the recall level r:

N_q is the number of queries used in the evaluation:

$P_i(r)$ is the precision at the recall level r for the i^{th} query.

Recall levels for various queries are almost always different from the 11 standard levels. To bring them all together to the 11 standard levels, we used an interpolation formula from the same source [BR99].

$$\bar{P}(r_j) = max \; r_j \le r \le r_{j+1} \; P(r)$$

This formula guides the process of finding interpolated precision for the j^{th} standard recall level. It is the maximum known precision at any recall level between the j^{th} recall level and the $(j+1)^{th}$ recall level.

Calculating "Interpolated Precision @ 11 Standard Recall Levels" Let us consider the following example. We conducted a number of searches using the same query by submitting it to a number of search engines. After evaluating all search results for relevancy, we decided that our "collection of relevant documents" consists of 15 documents. Now we can take results from an individual search and calculate an Interpolated Precision @ 11 Standard Recall Levels for this particular search. The following is a table of interim results obtained during one of our experiments.

Sequence# (by Search Engine)	*URL Address*	*Relevancy (by participant)*	*Cumulative Numb. of Relevant Documents*	*Calculated Precision*	*Calculated Relative Recall*
A	*B*	*C*	*D*	*E*	*F*
1	URL 1	1	1	*100*	*7*
2	URL 2	0			
3	URL 3	1	2	*67*	*13*
4	URL 4	0			
5	URL 5	1	3	*60*	*20*
6	URL 6	0			
7	URL 7	0			
8	URL 8	1	4	*50*	*27*
9	URL 9	0			
10	URL 10	0			
11	URL 11	1	5	*45*	*33*
12	URL 12	1	6	*50*	*40*
13	URL 13	1	7	*54*	*47*
14	URL 14	0			
15	URL 15	0			
16	URL 16	0			
17	URL 17	1	8	*47*	*53*
18	URL 18	0			
19	URL 19	0			
20	URL 20	1	9	*45*	*60*

Table 6-2 Calculating Precision/Recall Pairs

The first step of the process is to calculate individual Precision/Recall pairs for each relevant document (see Table 6-2 above) found during the search. Based on the binary relevancy in Column C (1 for relevant) we calculate the cumulative number of relevant documents in Column D. For each cumulative number of relevant documents we calculate the precision in Column E as a fraction of retrieved documents (number of the relevant documents in column D divided by the total number of retrieved documents in Column A). So, for example, in row 11 we show that we have found 5 relevant documents. To calculate Precision for this level we divide the 5 (number of relevant documents) by the 11 (number of retrieved documents) and we get 0.4555 or 45%. To calculate relative recall for this level we again divide the 5 (number of relevant documents) by the 15 (number of relevant documents in the collection) and we get 0.333 or 33%.

Now we can conclude the process by interpolating the precision for 11 standard levels. See Table 6-3 below.

Standard Recall Level *G*	*0*	*10*	*20*	*30*	*40*	*50*	*60*	*70*	*80*	*90*	*100*
Precision ***H***	**100**	**67**	**60**	**50**	**54**	**47**	**45**	**0**	**0**	**0**	**0**

Table 6-3 Calculating Interpolated Precision at 11 Standard Recall Levels

Here is the method we used to interpolate the precision for 11 standard recall levels:

For the 0% level: We looked for one or more values between 0% (inclusive) and 10 % (inclusive) in Column F (Table 6-2 above): we found only one 7%: we took the corresponding 100% from Column E (Table 6-2 above) as the interpolated precision for the 0% level.

For the 10% level: We looked in Column F for one or more values between 10% and 20%: we found two at 13% and 20%: we compared two corresponding numbers (67% and 60%) from Column E and selected the maximum of the two – 67% as the interpolated precision for the 10% level.

For the 20% level: We looked in Column F for one or more values between 20% and 30%: we found 27%: we took the corresponding 50% from Column E as the interpolated precision for the 20% level.

For the 30% level: We looked in Column F for one or more values between 30% and 40%: we found two at 33% and 40%: we compared two corresponding numbers from Column E and selected the maximum of the two – 50% as the interpolated precision for the 30% level.

For the 40% level: We looked in Column F for one or more values between 40% and 50%: we found 47%: we took a corresponding 54% from Column E as the interpolated precision for the 40% level.

For the 50% level: We looked in Column F for one or more values between 50% and 60%: we found two at 47% and 45%: we compared two corresponding numbers from Column E and selected the maximum of the two – 47% as the interpolated precision for the 50% level.

For the 60% level: We looked in Column F for one or more values between 60% and 70%: we found 60%: we took a corresponding 45% from Column E as the interpolated precision for the 60% level.

7 Description of Experiments

The objectives of our experiments were as follows:

first to validate the proposed algorithms and to determine the values for program (static) parameters used in algorithm implementation;

second to determine a range of values for operational (dynamic) parameters used in algorithm execution to optimize performance and enhance search results;

third to compare the performance of the proposed algorithms with the performance of major commercial search engines;

fourth to demonstrate the advantages of user feedback in long query processing.

To achieve our objectives, we designed a series of retrieval experiments to collect performance data for the new engine as well as for existing commercial engines.

This chapter explains the structure and contents of our experiments, discusses the environment we created to carry out the experiments, describes the organization and performance of experiments and data collection, and finally discusses and illustrates experimental results.

7.1 Structuring the Experiments

We created three groups of experiments:

- developmental experiments;
- comparative experiments;
- user feedback experiments.

The first group of experiments was used to determine values for some constants used in program development and operational variables used in executing algorithms implemented for the Long Query Search Engine. The second group of experiments was used to compare the performance of the proposed algorithms against commercially available search engines. And finally, the third group of experiments was used to demonstrate the advantages of proposed algorithms when executed along with user feedback versus the original autonomous mode.

7.1.1 Classifying the Experiments

The number of various retrieval experiments was quite large. For ease of reference to individual experiments we introduced a two-character Configuration Id to identify the search engine and operational parameters specific to each experiment. The first character indicates the search engine used for retrieval:

G – Google.

A – AltaVista.

F – Fast (Alltheweb).

N – Northern Light.

O – Long Query Search Engine (LQSE) with Open End Algorithm.

C – Long Query Search Engine (LQSE) with Close End Algorithm.

S – Long Query Search Engine (LQSE) with Similarity Analysis Calculator

The second character indicates the type of experiment conducted with the search engine specified by the first character. The options are as follows:

S, L	**S**hort or **L**ong Query submitted to a commercial search engine (valid when the first character equals "G", "A", "F" or "N")
X	Final version of LQSE with the Open End algorithm (valid when the first character equals "O")
3, 4, 5, 6	Number of variable length subqueries generated by the LQSE – Open End algorithm (valid when the first character equals "O")
Y	Final version of LQSE with the Close End algorithm (valid when the first character equals "C")
A, B, C, E, F, I, J, M	Number and size of fixed length subqueries generated by the LQSE – Close End algorithm (valid when the first character equals "C"); see Appendix F for details.
S	Similarity Analysis Calculator in conjunction with the LQSE – Close End algorithm (valid when the first character equals "C").

7.1.2 Developmental Experiments

The LQSE implemented two distinct algorithms that generate subqueries: Open End and Close End algorithms. For ease of discussion we will refer to these two configurations as an Open End engine and a Close End engine. The following discussion explains the details of each engine and the operational parameters selected for the experiments.

Open End Experiments The Open End algorithm generates a fixed number of variable length queries that were submitted to commercial search engines for retrieval. The results from all submittals were then merged using the Single Engine Fusion (SEF) algorithm. Almost from the beginning we discontinued experimenting with two subqueries due to a lack of tangible results, and concentrated on cases where the number of subqueries was in the range between 3 and 6 (see Table 7-1 below). The best results were obtained when the number of subqueries was either 4 or 5 (neither provided clear advantages). Since construction of each subquery required extra retrieval efforts we decided to use 4 subqueries. This eventually became what we call a final version of the LQSE (Open End engine) identified in the other experiments as **OX**. For description of developmental experiments refer to Section 7.4 below.

Configuration	*Search Engine Used*	*Comment*
O3	**LQSE (Open end)**	**Open End Algorithm (3 subqueries)**
O4	**LQSE (Open end)**	**Open End Algorithm (4 subqueries)**
O5	**LQSE (Open end)**	**Open End Algorithm (5 subqueries)**
O6	**LQSE (Open end)**	**Open End Algorithm (6 subqueries)**

Table 7-1 Developmental Experiments (Open End Configurations)

Close End Experiments The Close End algorithm generates a variable number of fixed length queries that were submitted to commercial search engines. The results from all searches are then merged using the Multiple Presentation Counter (MPC) algorithm. The Close End algorithm accepts two operational parameters M – maximum number of search terms used to generate subqueries and L – minimum number of search terms in a

subquery. Our initial experiments indicated that some of the combinations of M and L (see Appendix F for gray colored columns) did not produce consistent results (final results do not have the minimum 20 documents or have many documents appearing only once during multiple searches). So we concentrated on combinations that offered promising results (see Table 7-2 below). We determined that very good results were obtainable when the value of M and L were as follows: $7 \leq M \leq 9$ and $3 \leq L \leq 5$. Some of the long queries produced excellent results with M=7 and L=5, while others produced good results with M=9 and L=3. Since the "9 3" combination produced the highest number of very good results, it became what we call a final version of LQSE (Close End engine) identified in other experiments as **CY**. For description of developmental experiments refer to Section 7.4 below.

Configuration	*Search Engine Used*	*Comment*
CA	**LQSE (Close end)**	**Close End algorithm (M = 9, L = 3)**
CB	**LQSE (Close end)**	**Close End algorithm (M = 9, L = 4)**
CC	**LQSE (Close end)**	**Close End algorithm (M = 9, L = 5)**
CE	**LQSE (Close end)**	**Close End algorithm (M = 8, L = 3)**
CF	**LQSE (Close end)**	**Close End algorithm (M = 8, L = 4)**
CI	**LQSE (Close end)**	**Close End algorithm (M = 7, L = 3)**
CJ	**LQSE (Close end)**	**Close End algorithm (M = 7, L = 4)**
CM	**LQSE (Close end)**	**Close End algorithm (M = 6, L = 3)**

Table 7-2 Developmental Experiments (Close End Configurations)

Similarity Analysis Calculator Some of the Close End engine configurations (e.g. CA) described above require a large number of subqueries (see Appendix F Part A). To minimize time and resources required by these configurations, we introduced the SAC

algorithm, which works in conjunction with the Close End engine and improves the search results of its low-end configurations (e.g. CM). This combination of Close End engine with the SAC algorithm is classified in our experiments as the CS configuration (see Table 7-3 below). For description of developmental experiments refer to Section 7.4 below.

Configuration	*Search Engine Used*	*Comment*
CA	**LQSE (Close end)**	**Close End algorithm (M = 9, L = 3)**
CS	**LQSE (Close end)**	**With SAC algorithm**
CM	**LQSE (Close end)**	**Close End algorithm (M = 6, L = 3)**

Table 7-3 Developmental Experiments (SAC Configuration)

7.1.3 Comparative Experiments

This set of experiments was designed to compare the performance of LQSE (Open End and Close End) against the commercial search engines. The participants were asked to read the long query assigned to them, create a short query and submit it to all four engines. The participants were also asked to attempt to submit a copy of the original long query (regardless of the query length or character contents) again to all four commercial engines. At the end, the long query was also submitted to the LQSE (Open End and Close End engines). After participants evaluated search results from all 10 searches (see Table 7-4 below) for relevancy, they saved their experiment results in appropriate Excel workbooks for later tabulation. For description of comparative experiments refer to Section 7.4 below.

Configur.	*Search Engine Used*	*Comment*
GS	Google (Short query)	Short Query is formulated by the user
AS	AltaVista (Short query)	Short Query is formulated by the user
FS	Fast (Short query)	Short Query is formulated by the user
NS	NorthernLight (Short query)	Short Query is formulated by the user
GL	Google (Long query)	Long Query is submitted by the user
AL	AltaVista (Long query)	Long Query is submitted by the user
FL	Fast (Long query)	Long Query is submitted by the user
NL	Northern Light (Long query)	Long Query is submitted by the user
OX	LQSE (Open end)	Open End algorithm (Final)
CY	LQSE (Close end)	Close End algorithm (Final)

Table 7-4 Comparative Experiments (LQSE vs. Commercial Search Engines)

7.1.4 User Feedback Experiments

To show that further improvements could be obtained with the user's feedback, we conducted experiments to allow the users to modify the search term selection and ordering process, normally performed by the LQSE autonomously. We compared results collected for the **OX** and **CY** experiments in the preceding section and compared them with results obtained by participants using the feedback option of the LQSE (see Table 7-4 below). For description of user feedback experiments refer to Section 7.4 below.

Configuration	*Search Engine Used*	*Comment*
OX	LQSE (Open end)	Open End algorithm (without Feedback)
OU	LQSE (Open end)	Open End algorithm (with Feedback)
CY	LQSE (Close end)	Close End algorithm (without Feedback)
CU	LQSE (Close end)	Close End algorithm (with Feedback)

Table 7-5 User Feedback Experiments (Long Query Search Engine)

7.2 Creating the Experiment Environment

To conduct our experiments we have created an environment where participants could perform all the required tasks wherever and whenever they desired without supervision or outside assistance. All participants were given a diskette with an assignment sheet, a copy of all the queries, spreadsheet templates to collect test data, and a detailed experiment guide.

7.2.1 Putting Together a Query Collection

We visited numerous Web sites and selected 50 sites to create a pool of long queries for the experiments. Criteria for selection were different in nature – variety of subjects, clarity of the topic and quality of the language, length of description, and many others.

Once the site was chosen, we selected a portion of the text (between 400 and 6,400 characters), removed all identifying information (e.g. name of the company or the product from the manufacturing site, title of the book, author's name or the name of the hero from the literary site, title of the article or keywords from a scientific site, etc.) and stored it as a MS Word file (see Appendix H). Altogether these 50 queries covered 22 disciplines in 5 areas of human knowledge (see Appendix I).

7.2.2 Selecting Search Engines

To perform information retrieval experiments, participants were instructed to use our Long Query Search Engine (Open End and Close End engines) as well as the following four commercial search engines: Google, AltaVista, Fast (Alltheweb) and

Northern Light. These search engines claim to have the industry's largest databases (between 0.4 and 1.6 billion sites), they do their own indexing, maintain their own databases of indexed and un-indexed sites, and perform their own searches. And the most important criteria - they all allow for longer queries (Google –255 characters, AltaVista – 800 characters, Fast – 1024 characters, Northern Light – 4,500 characters). In addition to creating and submitting their own short queries, participants were instructed to attempt to submit the entire long query to all four engines.

7.2.3 Collecting and Analyzing Data

The data were collected on an MS Excel workbook (see Appendix N) developed for these experiments. It consists of two worksheets: one to identify the experiment (Id Page) and the second to collect experiment data (Data Page) from all the individual searches performed by the participants for each query. All tabulation, as well as chart generation were performed by using Excel's built-in functions and/or some additional VBA programs written specifically for these experiments. Some manual administrative tasks to ascertain quality were required (verifying and correcting file names, creating lists of files, etc.).

7.3 Organizing and Conducting the Experiments

To conduct the experiments we assembled a group of 25 undergraduate and graduate students. It has been shown that a group of this size is sufficient to accept the significance of experimental results [M90, WJ99]. Participants were not instructed on how to formulate the queries based on a given abstract or how to use regular (or

advanced) search engine interfaces: however, an exercise session was conducted to familiarize participants with the experiment's procedures and the tasks to be performed.

All the participants received a diskette with an assignment sheet, a MS Word copy of all queries, MS Excel workbook templates to collect experiment data (see Appendix J), and a detailed Experiment Guide (see Appendix G). All participants were asked to complete a variety of tasks - long query text analysis, short query formulation, information retrieval using designated search engines, relevancy judgment and collection of experimental data. Participants had two weeks to complete the assigned tasks at their own convenience without any supervision or outside assistance.

7.4 Experimental Results

The purpose of the first round of experiments, which we call developmental, was to determine the best configurations (static and dynamic parameters) for the proposed algorithms. The developmental experiments are described in sections 7.4.1 – 7.4.3. Once we determined the best configurations we used them in experiments comparing our algorithms with other engines, described in section 7.4.4, and in experiments involving user feedback, described in section 7.4.5.

7.4.1 Open End Engine Experiments

This engine employs Open End Query (OEQ) and Single-Engine Fusion (SEF) algorithms. Based on the user's long query a fixed number of variable length subqueries are created using the OEQ algorithm, then they are submitted to an existing search engine

(e.g. AltaVista), and afterwards all the search results are merged into a single ranked output using the SEF algorithm. As we described in Section 5.3 above, the OEQ algorithm requires one operational parameter – the number of variable length subqueries. Initial experiments with less than three subqueries did not give enough "merging" information to the SEF algorithm, so we concentrated on experiments with three or more subqueries. We did not extend our experiments further than six subqueries, since we saw no improvements in quality. We classified the four configurations participating in these experiments as O4 to O6 where the second character specified the number of variable length subqueries generated by each configuration.

We used 22 original long queries (one from each discipline) out of the total 50 queries (see Appendix I). Table 7-6 below shows that configuration O4 demonstrated the overall best performance among the other engines in this group in finding and placing the original document among the Top 20 documents.

Configuration	*Top*	*Top 5*	*Top 10*	*Top 20*
O3	**41**	**55**	**64**	**68**
O4	**73**	**82**	**91**	**91**
O5	**77**	**77**	**82**	**91**
O6	**77**	**86**	**86**	**86**

Table 7-6 Placing of the Original Document among the Top 20 results for Open End Configurations (%)

Further analysis of these four configurations using the precision measure of P@10 and P@20 shows that O5 performed slightly better then O4, but considering the additional searches required by O5 to build the extra subquery and then to merge the

extra results, we decided to stay with O4. See Table 7-7 for precision results for all four configurations.

Configuration	*P@10*	*P@20*
O3	0.18	0.11
O4	0.29	0.20
O5	0.30	0.20
O6	0.28	0.21

Table 7-7 P@10 & P@20 for Open End Configurations

The third measurement we used to compare the four configurations was interpolated precision at 11 standard recall levels. We calculated, as described in Chapter 6, relative recall, precision and interpolated precision for all queries and summed up the results in Table 7-8 below.

Configuration	*0*	*10*	*20*	*30*	*40*	*50*	*60*	*70*	*80*	*90*	*100*
O3	33	44	52	47	32	24	21	0			
O4	85	72	74	62	56	39	0				
O5	89	70	66	50	53	47	0				
O6	90	79	71	55	49	0					

Table 7-8 Interpolated Precision At 11 Standard Recall Levels for Open End Configurations

Figure 7-1 below illustrates the chart based on Table 7-8 above. Configuration O5 shows some recall levels where its precision is higher than the precision of configuration O4, but overall configuration O4 performed better than configuration O5.

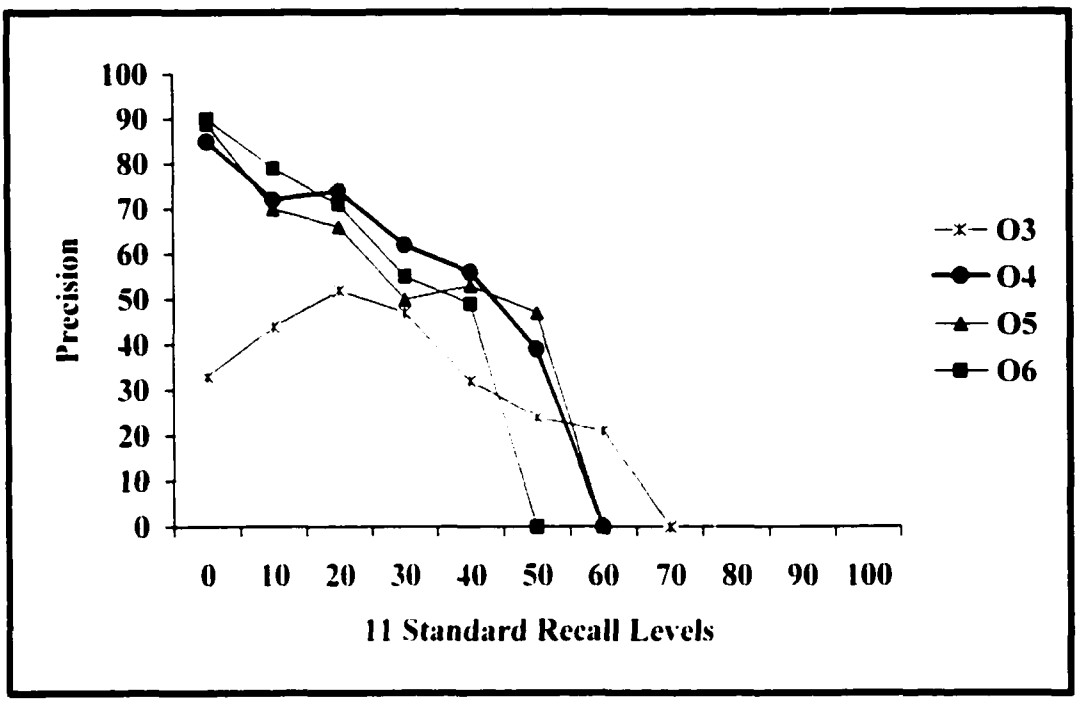


Figure 7-1 Interpolated Precision at 11 Standard Recall Levels for Open End Configurations

We used configuration O4 as the Open End Engine in further experiments and refer to it in the following discussions as the OX engine.

7.4.2 Close End Engine Experiments

This engine employs Close End Query (CEQ) and Multiple Presentation Counter (MPC) algorithms. Based on the user's long query a variable number of fixed length subqueries are formulated using the CEQ algorithm, then they are submitted to one or

more existing search engines (e.g. Google, AltaVista, etc.) and afterwards all search results are merged into a single ranked output using the MPC algorithm.

As we described in Section 5.4 above, the CEQ algorithm is using two operational parameters to formulate fixed length subqueries: M – maximum number of search terms used to formulate the subqueries and L – minimum size of the subquery. Depending on the combinations of these two parameters this engine could be very Web search intensive – the number of subqueries submitted to search engine(s) could jump to hundreds and even thousands. We extensively experimented with this algorithm using the values for $6 \leq M \leq 9$ and $3 \leq L \leq 5$. Using $M > 9$ would cause over 1000 subquery searches for one long query. Correspondingly, using $M < 6$ usually returns one (the original document) or no results. On the other hand, using $L > 5$ usually returns one (the original document) or no results and $L < 3$ returns a large number of mostly irrelevant results.

We listed in Appendix F fifteen various combinations of M and L parameters. Each column (denoted A to O) in the appendix specifies the values of the two parameters and the number of subqueries required to complete one long query search. The figures in the data portion of the appendix specify the number of multiple presentations of each document (this number is used to rank the final list) after all the subqueries are completed. Even though this appendix represents the results of one sample query, it does reflect a typical finding – the following M and L combinations (D, G, H, K, L, N, O) return very few search results (see prevalence of 0 and 1 in the final tally). We conducted all Close End engine experiments with the remaining combinations of M and L. As we mentioned above, we classified all the experiments in this group by a two-character

Configuration Id – first character C for Close End engine and the second character to identify the combinations of M and L (see Appendix F).

We used 22 original long queries (one from each discipline) out of the total 50 queries (see Appendix I). Table 7-9 below shows that configuration CA demonstrates the overall best (almost perfect) performance among other configurations in this group in finding and placing the original document among the Top 20 documents.

Configuration	*Top*	*Top 5*	*Top 10*	*Top 20*
CA	95	96	96	96
CB	82	86	86	86
CC	77	82	86	86
CE	77	82	82	82
CF	77	77	77	77
CI	73	77	77	77
CJ	68	68	73	73
CM	50	55	64	68

Table 7-9 Placing of the Original Document among the Top 20 results for Close End Configurations (%)

Further analysis of these eight configurations using the precision measure of $P\bar{a}10$ and $P\bar{a}20$ shows that configuration CA performed slightly better than configurations CB and CC (which are using M = 9) and certainly better than other configurations using M < 9. See Table 7-10 below for precision results for all the configurations in this group.

Configuration	*P@10*	*P@20*
CA	0.44	0.27
CB	0.44	0.26
CC	0.43	0.24
CE	0.38	0.22
CF	0.38	0.20
CI	0.34	0.19
CJ	0.22	0.18
CM	0.29	0.17

Table 7-10 P@10 & P@20 for Close End Configurations

The third measurement we used to compare the eight configurations was interpolated precision at 11 standard recall levels. We calculated relative recall, precision and interpolated precision for all queries and presented the results in Table 7-11 below.

Configuration	*0*	*10*	*20*	*30*	*40*	*50*	*60*	*70*	*80*	*90*	*100*
CA	95	87	80	64	70	60	47	0			
CB	89	83	69	65	55	43	42	18	0		
CC	78	62	72	61	55	49	42	0			
CE	90	81	74	77	62	50	34	23	0		
CF	84	75	69	65	47	34	0				
CI	76	68	65	54	55	43	20	0			
CJ	72	65	54	57	50	37	33	0			
CM	82	67	55	43	38	21	0				

Table 7-11 Interpolated Precision at 11 Standard Recall Levels for Close End Configurations

Figure 7-2 below illustrates the chart based on Table 7-11 above. Configuration CA – Close End engine (with M = 9 and L = 3) is clearly the best configuration in this group.

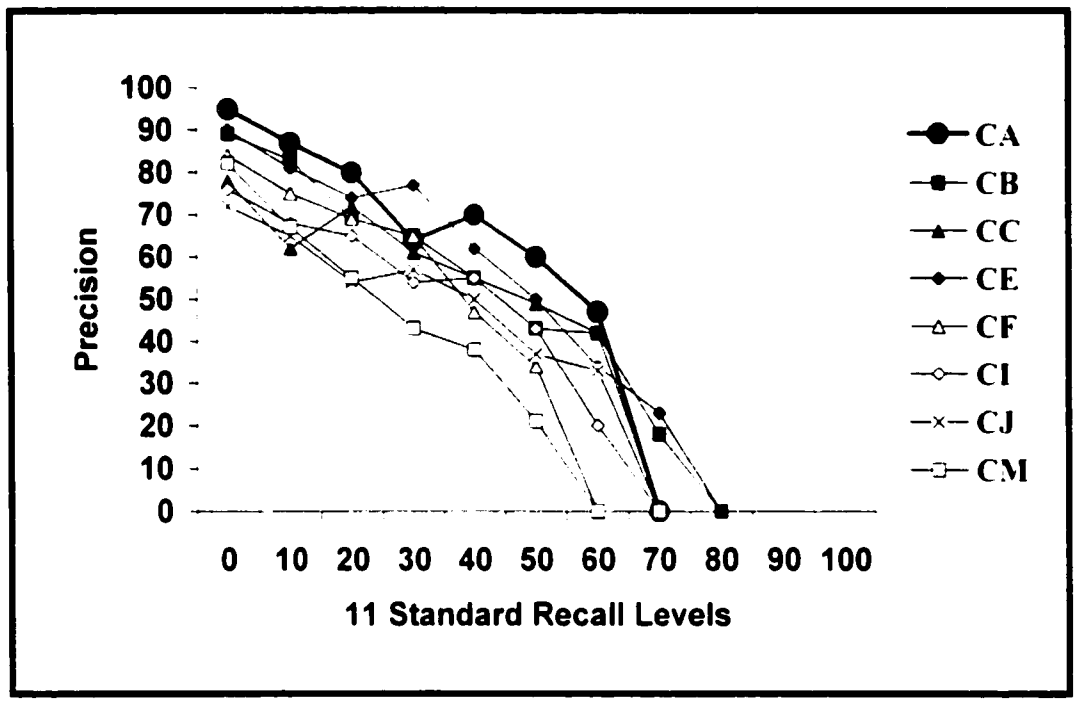


Figure 7-2 Interpolated Precision at 11 Standard Recall Levels for Close End Configurations

We used the CA configuration as the Close End engine in further experiments and refer to it in the following discussions as the CY engine.

7.4.3 Similarity Analysis Calculator Experiments

While analyzing the results of Close End experiments, we spotted some trends that required further investigation. It was clear that the highly relevant documents or their various representations (PDF, HTML, Plain Text, etc) at different Web sites almost

always came to the top of the final ranked list regardless of the value of M (number of search terms used to generate subqueries). Further analysis revealed that the number of presentations (occurrences) of top results (see Appendix F Part B) is closely correlated (95% - 99%) with the number of subqueries generated by various configurations (see Appendix F Part A). In other words, we got the same top results by using fewer subqueries.

Furthermore, the analysis of Appendix F (Part B) showed that many URLs appeared only once and the only reason they made it to Top 20 was because, as the analysis of logs revealed, they were returned by subqueries constructed first, and were given higher priority by the merge process. To test the relevancy of the results that did not make it to the final list, we submitted all the interim results to the Similarity Analysis Calculator (SAC) algorithm (it is a stand-alone program and it is not incorporated into the meta-search engine). Over 30% of these interim results were found to be relevant. To verify how the SAC algorithm could improve the performance of the low search intensive Close End configurations we performed the following experiments.

For this group of experiments we used the prior Close End engine experimental data for two configurations CA (the best performing configuration) and CM (the worst performing configuration). We took the low frequency (number of occurrence = 1) interim results of CM configurations and inputted them into the SAC algorithm. We combined high frequency results from CM with the SAC output to create a final Top 20 results list for the combined configuration, which we classify as CS.

Overall, the new configuration CS performed better than the original configuration CM. There was a slight improvement in the placing of the original document (see Table 7-12 below).

Configuration	*Top*	*Top 5*	*Top 10*	*Top 20*
CA	95	96	96	96
CS	50	55	68	71
CM	50	55	64	68

Table 7-12 Placing of the Original Document Among the Top 20 Results for SAC Experiments (%)

While $P\bar{a}10$ showed only 1% improvement, the $P\bar{a}20$ was 14% higher (see Table 7-13 below).

Configuration	*P@10*	*P@20*
CA	0.44	0.27
CS	0.30	0.31
CM	0.29	0.17

Table 7-13 $P\bar{a}10$ & $P\bar{a}20$ for SAC Experiments

We also calculated relative recall, precision and interpolated precision for all queries for the CS configuration and presented the results in Table 7-14 below.

Configuration	*0*	*10*	*20*	*30*	*40*	*50*	*60*	*70*	*80*	*90*	*100*
CA	95	87	80	64	70	60	47	0			
CS	82	67	62	57	50	48	39	35	0		
CM	82	67	55	43	38	21	0				

Table 7-14 Interpolated Precision at 11 Standard Recall Levels for SAC Experiments

Figure 7-3 below illustrates the chart based on Table 7-14 above. While it is obvious that configuration CS is better than the configuration CM, it is far behind the winning CA configurations. On the other hand, it is important to note here that the CM configuration uses only 9% of the total number of subqueries required for the CA configuration.

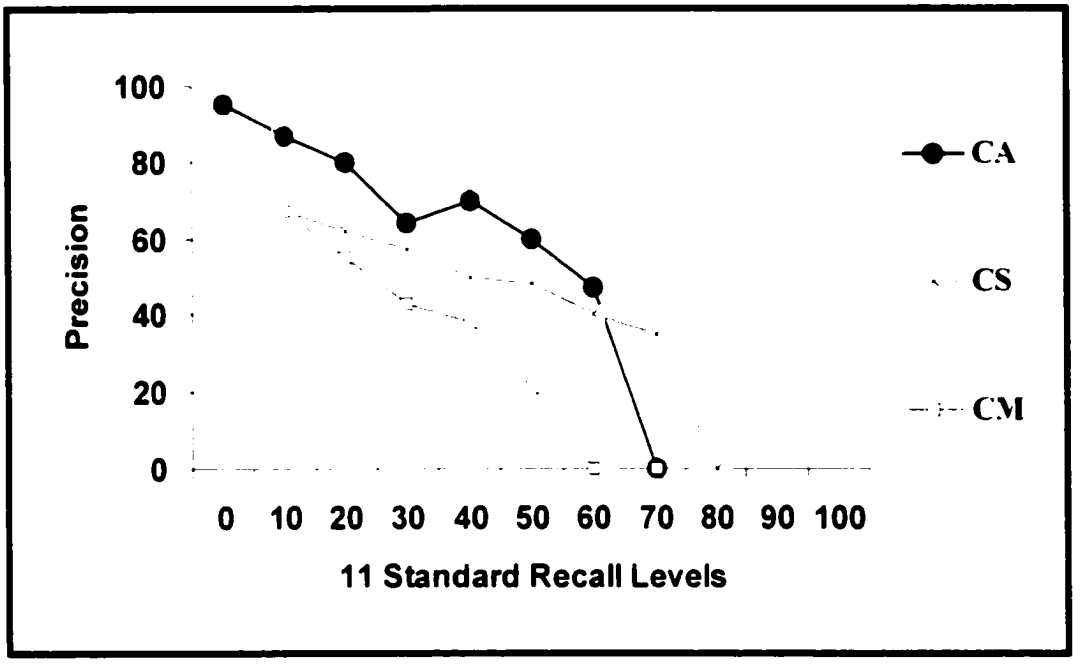


Figure 7-3 Interpolated Precision at 11 Standard Recall Levels for SAC Experiments

7.4.4 Comparative Experiments

For our comparison experiments we used the following search engines: AltaVista, Fast, Google, Northern Light. All four engines were used in two modes: the *short* - when a short query was created by the user and submitted to the engine for information retrieval and the *long* – when the original long query was submitted (as is) to the engine

for information retrieval. We started our experiments last year before Google introduced PDF files to its collection, so we added our earlier results to our current results for comparison.

We classified each configuration by a two-character configuration ID where the first character specifies the engine (A, F, G, N) and the second character specifies the mode (S for short query mode and L for long query mode). The Google engine that we used last year we identified as GP (Google before the PDF files).

Together with our Close End and Open End engines we compared the performance of eleven configurations (see Table 7-15 below for the list of configurations). For these experiments we used all 50 queries (see Appendix I).

From the beginning we detected a trend – our LQSE (configurations CY and OX) together with Google (configurations GL and GP) were clear winners. Fast and Northern Light (all configurations) trailed far behind. Google (GL) and AltaVista (AL), both in long query mode, failed miserably.

In the first measuring category – positioning of the original document, the configuration CY was a clear winner with almost perfect performance. Configuration GS (Google - current release) almost always found the original document, but did not place it on top of the list. See Table 7-15 below for all the results.

Configuration	*Description*	*Top*	*Top 5*	*Top 10*	*Top 20*
CY	Close End Engine (Long Query)	96	100	100	100
OX	Open End Engine (Long Query)	76	90	96	96
GP	Google (Short Query – without PDF)	40	42	42	42
GS	Google (Short Query – with PDF)	44	52	54	54
AS	AltaVista (Short Query)	0	4	4	6
FS	Fast (Short Query)	4	8	14	14
NS	Northern Light (Short Query)	4	12	12	12
GL	Google (Long Query)	0	0	0	0
AL	AltaVista (Long Query)	0	0	0	0
FL	Fast (Long Query)	0	4	6	6
NL	Northern Light (Long Query)	0	6	6	6

Table 7-15 Placing of the Original Document among the Top 20 results for All Configurations (%)

In the second measuring category, precision, configuration CY (LQSE – Close End engine) performed better than the other configurations. It was 12% better at Precision@10 and 6% better at Precision@20 vs. the second place configurations GL (current Google in short query mode).

Configuration OX (LQSE – Open End) was almost as good as configuration GP (last year Google in a short query mode) but not as good as configuration GL (current Google in short query mode). See Table 7-16 below for all configurations precision results.

Configuration	*P@10*	*P@20*
CY	0.46	0.31
OX	0.27	0.18
GP	0.23	0.14
GS	0.34	0.25
AS	0.04	0.05
FS	0.10	0.07
NS	0.17	0.11
GL	0.00	0.00
AL	0.05	0.02
FL	0.11	0.07
NL	0.16	0.11

Table 7-16 P@10 & P@20 for All Configurations

The third measurement category we used to compare the eleven configurations was interpolated precision at 11 standard recall levels. We calculated relative recall, precision and interpolated precision for all queries, and present the results in Table 7-17 below.

Configuration	*0*	*10*	*20*	*30*	*40*	*50*	*60*	*70*	*80*	*90*	*100*
CY	96	83	83	76	59	53	57	0			
OX	82	76	77	64	52	43	39	0			
GP	80	85	71	60	46	26	0				
GS	94	78	73	61	53	66	52	30	0		
AS	22	13	0								
FS	25	15	0								
NS	28	19	8	3	0						
FL	28	25	19	0							
NL	30	18	11	6	0						

Table 7-17 Interpolated Precision at 11 Standard Recall Levels for All Configurations

Figure 7-4 below illustrates the chart based on Table 7-17 above. Clearly the best two engines are the Close End engine and Google. It is important to stress here that while Google results are based on individual short queries formulated by individual users, the Close End engine results were obtained solely in automatic mode, where the user did not have to analyze the original long query and formulate a search query.

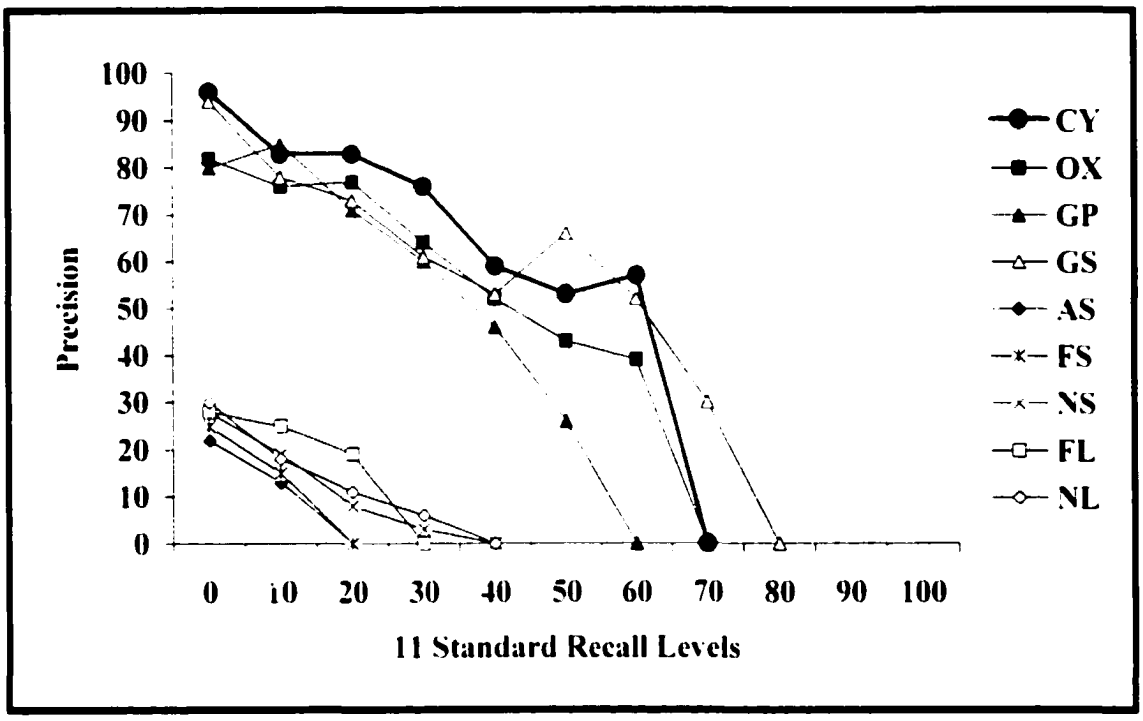


Figure 7-4 Interpolated Precision at 11 Standard Recall Levels for All Configurations

7.4.5 User Feedback Experiments

To demonstrate that the search capabilities of the Open End and the Close End engines could be improved even further by the user's participation, we conducted an additional series of experiments.

The participants were asked to perform the following feedback tasks: first – to read the original long query and evaluate the retrieval results of the engine working in automatic mode; second – to review the list of the search terms ordered by the engine and to decide how to change the content and the order of the list to improve retrieval results; third – to introduce phrases built with search terms from the list (the execution of this task was left to participant discretion). Once the participants completed these tasks they restarted the retrieval process with the revised list of search terms and new phrases (if any). The participants were asked to repeat the process as often as desired.

We classified every configuration by a two-character configuration ID (CY and OX for Close End and Open End configuration in automatic mode and (CU and OU for Close End and Open End configuration in user feedback mode).

Each participant executed two of the fifty original long queries (see Appendix I). Table 7-18 below shows that the participants were able to improve an already high level of the Close End configuration (CU) in placing the original document on top of the result list. While Open End configuration (OU) in feedback mode showed a smaller rate of improvements, it was still higher than the original automatic mode. See Table 7-18 for the results of this experiment.

Configuration	*Top*	*Top 5*	*Top 10*	*Top 20*
CU	**95**	**95**	**95**	**95**
CY	**86**	**91**	**91**	**91**
OU	**73**	**80**	**86**	**86**
OX	**68**	**77**	**86**	**86**

Table 7-18 Placing of the Original Document Among the Top 20 Results for Open and Close End Configurations in Automatic and User Feedback Mode (%)

Further analysis of these four configurations using the precision measure of P@10 and P@20 shows that the Close End configuration (CU) showed a higher degree of improvements (8% for P@10) versus the Open End configuration (OU) that showed a smaller increase (3% for P@10). See Table 7-19 for precision results for the four configurations in automatic and user feedback modes.

Configuration	*P@10*	*P@20*
CU	0.51	0.35
CY	0.43	0.30
OU	0.35	0.25
OX	0.32	0.22

Table 7-19 P@10 & P@20 for Open and Close End Configurations in Automatic and User Feedback Mode

The third measurement we used to compare the performance of the four configurations was interpolated precision at 11 standard recall levels. We calculated relative recall, precision and interpolated precision for all queries and presented the results in Table 7-20 below.

Configuration	*0*	*10*	*20*	*30*	*40*	*50*	*60*	*70*	*80*	*90*	*100*
CU	95	86	77	73	56	55	43	0			
CY	92	83	87	71	62	48	0				
OU	84	76	60	52	40	29	0				
OX	79	71	62	43	27	16	0				

Table 7-20 Interpolated Precision at 11 Standard Recall Levels for Open and Close End Configurations in Automatic and User Feedback Mode

These data and the chart (see Figure 7-5 below) do not show as much improvement as the prior measurement P $\bar{a}$ 10 and P $\bar{a}$ 20.

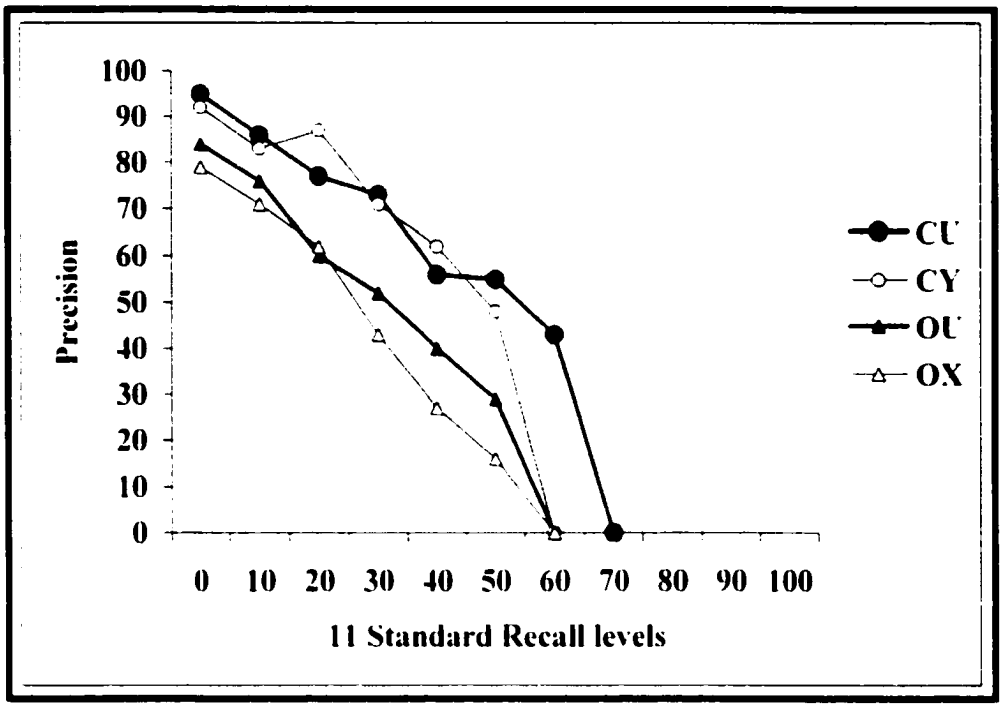


Figure 7-5 Interpolated Precision at 11 Standard Recall Levels for Open and Close End Configurations in Automatic and User Feedback Mode

7.5 Discussion of Experimental Results

The central question of this thesis is how to achieve higher quality information retrieval on the Web through the use of long queries. To answer this question we introduced several novel algorithms, a feedback mechanism, and a new user interface and implemented them as a meta-search engine. We used this engine to conduct an extensive series of experiments to validate the proposed algorithms, to compare the performance of various meta-search engine configurations with some of the existing commercial engines.

such as AltaVista, Fast, Google, Northern Light, and to examine and measure the advantages of user feedback for long query searches.

We then asked the experiment's participants to analyze long queries from our collection compiled from the Web, formulate traditional short queries and to submit them to all the engines in the experiment. The participants were also asked to submit the long queries to the same engines without regard to the query's length or content or the engine's known ability to accept or process such queries. At the end, the same long queries were submitted to our meta-search engine (Open and Close End configurations). The participants evaluated the "Top 20" results returned from each query submitted to each search engine using the ternary relevance scale, which allows them to choose from the following options: not relevant, somewhat relevant, very relevant. We collected all the evaluations and tabulated the final results.

7.5.1 Fine Tuning the Algorithms

Our first task was to establish optimum environments for the Open End and Close End engines. We identified key parameters of our algorithms and constructed different engines' configurations for evaluation in our experiments.

Open End Engine During our initial developmental experiments, we determined that the Open End Query (OEQ) algorithm performed the best with the Single Engine Fusion (SEF) algorithm. Once we determined the make up of the engine, we experimented with configurations using a various number of subqueries built by the engine and submitted to commercial engines for retrieval. From the beginning we discarded configurations with

less than three subqueries because these configurations did not provide enough information for merging algorithms. Subsequently, we concentrated on experiments with three or more subqueries. Further experiments indicated that the most promising configurations are the ones with four or five subqueries. The evaluation results of both configurations were almost identical. Therefore, we chose the configuration with four subqueries because it requires fewer searches on the Web and less computing resources to merge search results.

Close End engine The Close End Query (CEQ) algorithm, which creates a very large number of intersecting subqueries, performed efficiently with the Multiple Presentation Counter (MPC) algorithm. The CEQ algorithm depends on two key parameters (M – maximum number of search terms, and L – minimum number of terms in a subquery). We determined that the combination of $6 \leq M \leq 9$ and $3 \leq L \leq 5$ is the most successful. Using values outside of these ranges causes either a substantial increase of Web searches without any improvement in the quality of search results or a significant decrease in the quality of search results. To improve the search quality of time efficient configurations, we introduced a Similarity Analysis Calculator (SAC) algorithm, which together with the most time efficient configuration ($M = 6$ and $L = 3$) minimizes the number of subqueries needed to produce good quality results up to 90%. The SAC algorithm accepts the interim results (URL list) produced by the MPC algorithm, retrieves the actual documents referred to by the URLs and calculates the similarity coefficient for each document. The final ranking is determined by these similarity coefficients.

After completing the experiments we determined that the Close End engine search results were 19% ($P\bar{a}10$) and 13% ($P\bar{a}20$) better than the results returned by the Open End engine.

7.5.2 Comparing the New Engine with the Commercial Engines

After completing the meta-search engine implementation and the developmental experiments, we conducted experiments to compare the quality of the results of this engine with commercial engines. For these experiments we used Open End and Close End engines and four popular search engines: AltaVista, Fast, Google and Northern Light.

When a long query was submitted in its original form to the commercial engines they either did not accept the query or returned results of poor quality. Our engine, on the other hand, consistently returned many relevant documents among the top ranked results. To see how a long query search with our engine compares with a short query search with commercial engines, we asked our experiment participants to manually formulate short queries based on the original long queries and submit them to commercial search engines. Again, our engine was the clear winner. Our best performing Close End configuration was 12% ($P\bar{a}10$) and 6% ($P\bar{a}20$) better than Google, the best performing commercial engine in our experiments and widely recognized as the best search engine on the Web. These results were consistent throughout the experiments using various evaluation metrics and an extensive collection of long queries.

7.5.3 Applying User Feedback

In the comparative experiments, we did not use the feedback feature of our engine because the commercial search engines are not designed to accept user feedback. We did, however, run other experiments with the feedback feature of our engine.

We asked the experiment participants to evaluate the initial search results. Then they reviewed the list of search terms automatically selected and ordered by the algorithms and were asked to either reorder the list or to introduce new search terms and/or phrases. Once they refocused the query, they restarted the search process. This feedback mechanism was easy to use, and the participants were asked to repeat it for every query as often as desired.

These experiments showed that when the feedback option was used it consistently improved the quality of the search results. The Close End engine showed an improvement of 8% ($P@10$) and the Open End engine showed an improvement of 3% ($P@10$).

8 Conclusions

The quality of information retrieval on the Web has become increasingly important in light of the unprecedented growth and widespread use of the Web. A key factor that determines the quality of retrieved information is a user's query which is a representation of the user's information need. This thesis proposes to improve the quality of information retrieval on the Web through the use of long queries. A long query provides a user with the opportunity to express the information need in natural language. At the same time, longer queries allow for a more detailed description of the information need.

8.1 Contributions

We examined commercial search engines in terms of their ability to accept and process long queries. We found that most search engines do not accept long queries and the few engines that accept them do not produce adequate search results.

We developed a search engine with the following innovations for allowing long query processing:

Novel Search Algorithms - We introduced several algorithms to improve all aspects of long query information retrieval on the Web. Our proposed algorithms take the original long query and generate multiple formulations (subqueries) of this query. The subqueries are submitted to search engine(s) and the obtained search results are merged into one ranked output. These algorithms expand on the idea that if a Web document is obtained numerous times in response to different formulations of the same query, then this document has a higher probability of being relevant to the user.

Flexible User Feedback – We designed a process to give the user more control (if desired) in directing the search query formulation. The user is provided with an opportunity to indicate the importance of search terms and to create phrases from the list of search terms to better focus the long query search process. The feedback process was incorporated into the subquery formulation algorithms and was designed to be accessible by the user at any time during the search process.

User-Friendly Interface – We created a user-friendly interface to allow the user full control over all stages of the long query processing. The user gets a large query window to enter the long query either by typing or pasting it without any restriction on the size or special characters included in the text. Later, the user is given the option of selecting various search engine configurations, algorithms and operational parameters allowing the user to obtain better search results.

8.2 Results

We conducted numerous experiments to assess the proposed algorithms and user interface, and to evaluate their performance as part of the new meta-search engine.

We started with developmental experiments compare the quality of search results for different algorithms and configurations introduced in this thesis. The algorithm and configuration which consistently produced better search results than other configurations was the Close End algorithm, where the maximum number of search terms used in a subquery was 9 and the minimum number of terms in a subquery was 3. Although this algorithm requires more accesses

to the Web than the Open End algorithm, the actual time for obtaining results is quite fast and practically does not depend on the query length. In addition, we developed an algorithm to be used in conjunction with the Close End algorithm which allowed us to use fewer search terms in creating subqueries (substantially reducing the number of accesses to the Web) and still get good quality search results.

We also compared the quality of the search results achieved by our new engine with those of commercial engines. When a long query was submitted in its original form to the commercial engines, they either did not accept the query or returned results of poor quality. Our engine, on the other hand, consistently returned many relevant documents among the top ranked results.

To examine how a long query search with our engine compares with a short query search with commercial engines, we asked our experiment participants to manually formulate short queries based on the original long queries and submit them to commercial search engines. Again our engine was the clear winner. Our best performing Close End configuration was 12% ($P\bar{@}10$) and 6% ($P\bar{@}20$) better than Google, the best performing commercial engine in our experiments and widely recognized as the best search engine on the Web. These results were consistent throughout the experiments using various evaluation metrics and an extensive collection of long queries.

In the experiments comparing our engine to existing engines, we did not use the feedback feature of our engine because the commercial search engines are not designed to accept user feedback. We did, however, run experiments which used the feedback feature of our engine and tested the results against queries submitted without the feedback feature. We demonstrated that search results from long queries could be further improved by the user's involvement in the

query formulation process through the use of the feedback feature. By allowing the user to exercise the feedback option, we were able to show noticeable improvements for the Close End configuration (8% for $P@10$) over the long query without the feedback option version and smaller improvements for the Open End configuration (3% for $P@10$).

8.3 Future Work

Further research on using long queries in the Web environment could proceed in several directions. One direction is introducing dynamic methods in generating subqueries. Presently the Open End algorithm creates four variable length subqueries, each opening with the fixed search term in the ordered list. It will be beneficial to investigate an approach where selecting an opening term in a subquery and selecting the number of subqueries are based on the frequency of the search terms in the collection. A dynamic method could also be introduced in choosing parameters for the Close End algorithm. Presently, the size and the number of fixed length subqueries are chosen statically (as a set of constants) either by the user or as a program default. A promising approach would be to determine the values of the parameters based on the frequency of the search terms in the collection.

Another direction in extending our research is to consider the subject of a long query. Some of our experiments showed that for the same value of operational parameters, the quality of search results fluctuated for different subjects. Correctly classifying the query's subject matter is more likely in the case of a long query and it might be helpful in choosing specific algorithms and/or parameters for further improvement in the quality of search results.

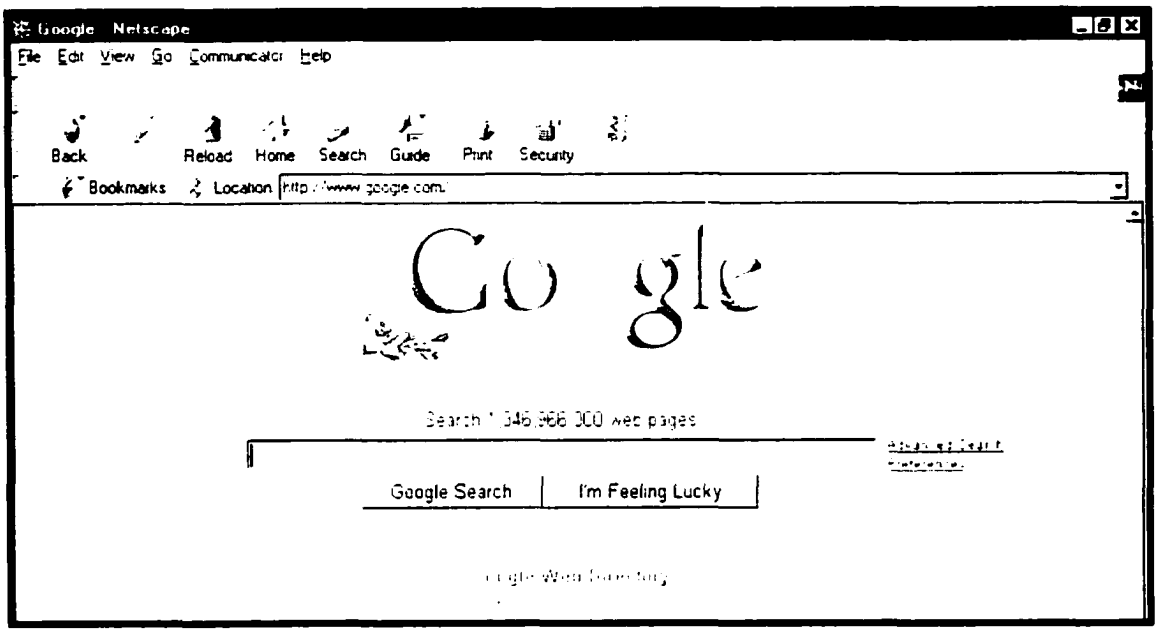
9 Abbreviations

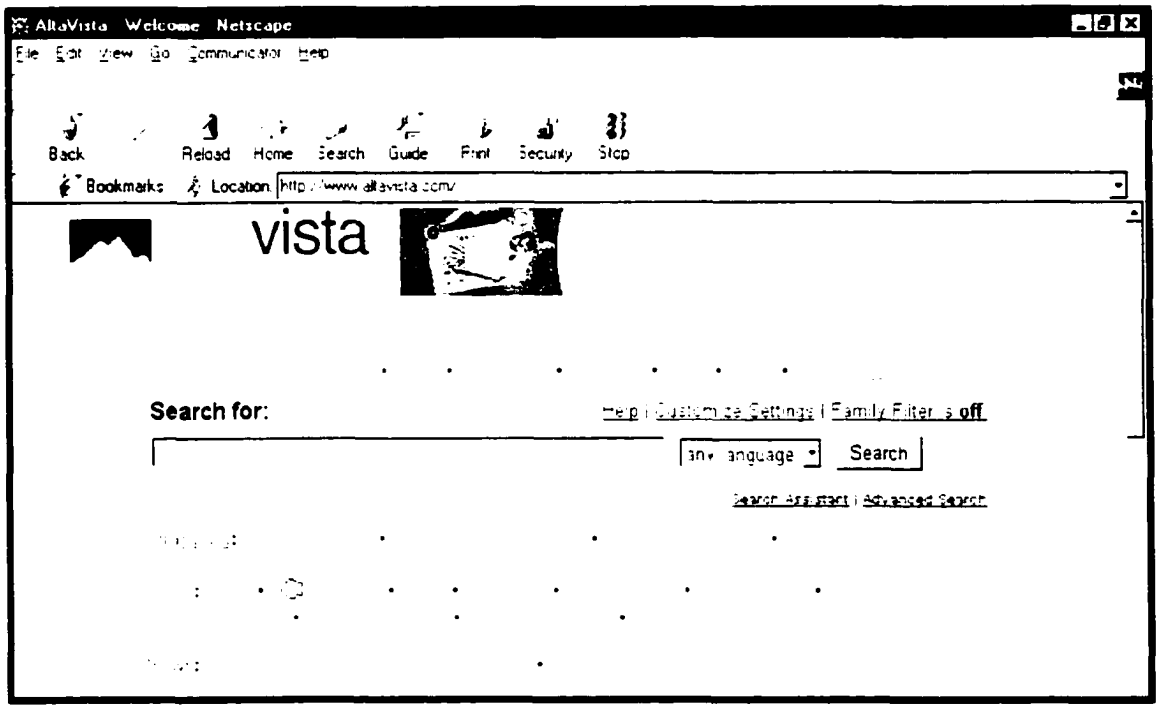
AUN	***Actual User's Need***
CEQ	***Close End Query***
CIN	***Concrete Information Need***
IN	***Information Need***
IR	***Information Retrieval***
IRIS	***Information Retrieval Interactive System***
LQSE	***Long Query Search Engine***
MPC	***Multiple Presentation Counter***
OEQ	***Open End Query***
OPAC	***On-Line Public Catalogue***
OQT	***Order Query Terms***
POIN	***Problem Oriented Information Need***
QPE	***Query Phrase Extractor***
QTF	***Query Terms File***
SAC	***Similarity Analysis Calculator***
SDR	***Spoken Document Retrieval***
SEF	***Single Engine Fusion***
SFQ	***Search Engine Formulated Query***
SUN	***Satisfied User's Need***
TREC	***Text REtrieval Conference***
UFQ	***User Formulated Query***
URL	***Uniform Resource Locator***

10 Appendices

Appendix A

Google and AltaVista Initial Windows





Appendix B

Sample Long Query

Visually impaired people are able to access digital information via a character interface by using screen readers [1][2]. But graphical user interfaces (GUIs) make it difficult for them to access information via computers. On the other hand, GUIs make it easy for sighted people to use computers, for two reasons. One is that they do not need to remember a lot of commands, and the other is that they can operate computers intuitively by manipulating graphical objects and metaphors with a mouse. These intuitive operations are not available to the visually impaired, since it is difficult for them to recognize two-dimensional information and graphical objects such as icons. There are screen readers for GUIs, such as Screen Reader/2 [3] for the Japanese version of OS/2 Warp [1], but they are not totally satisfactory. Screen Reader/2 reads all the text information on the screen, including icon labels, but it cannot represent visual information.

Nowadays, the Web is a useful information tool for computer users. However, visual data such as tables, image maps, and hierarchical sentence structures in World Wide Web (WWW) pages are big barriers for the visually impaired. Screen Reader/2 can only read the displayed text information from top to bottom, line by line, word by word, character by character, and so on, if there is no profile for an application written in profile access language (PAL) [4] of Screen Reader/2. In addition, Screen Reader/2 cannot read two-dimensional representations such as tables. This paper proposes a method for converting such visual representations into non-visual representations in Hyper Text Markup Language (HTML).

Appendix C

Long Query Search Engine Log

Log Number 196 - Program Used: Open End Query
Search Engine Results for *Google* 1

Initial Query String: Visually impaired people are able to access digital information via a character interface by using n readers ***(initial query deleted to fit the page)***
Parsed Query String: screen method image 12 discussion operations interfaces text recognize readers information wide ***(parsed query deleted to fit the page)*** 2
Total Number of words: 296 **Total Number of Stop Words Found:** 122

Creating Subquery #1
Terms 0 1 2 3 4 5 **only** returned 0... guis non-visual visually two-dimensional impaired intuitively

Terms 0 1 2 3 4 returned 9... guis non-visual visually two-dimensional impaired

Creating Subquery #2
Terms 1 2 3 4 5 6 returned 4... non-visual visually two-dimensional impaired intuitively representations

Creating Subquery #3
Terms 2 3 4 5 6 7 **only** returned 0... visually two-dimensional impaired intuitively representations graphical

Terms 2 3 4 5 6 **only** returned 0... visually two-dimensional impaired intuitively representations

Terms 2 3 4 5 returned 26... visually two-dimensional impaired intuitively 3

Creating Subquery #4
Terms 3 4 5 6 7 8 **only** returned 0... two-dimensional impaired intuitively representations graphical metaphors

Terms 3 4 5 6 7 **only** returned 0... two-dimensional impaired intuitively representations graphical

Terms 3 4 5 6 returned 35... two-dimensional impaired intuitively representations

Appendix D

Stop Words and Symbols

Stop Words: a above according across actually adj after afterwards again against almost alone along also although always am among amongst an and another any anyhow anyone anything anywhere are aren aren't around as at be became because become becomes been before beforehand begin behind being below beside besides between both but by can can't cannot co could couldn couldn't did didn didn't do does doesn doesn't don don't down during each eg else elsewhere end ending enough etc even ever every everyone everything everywhere except few for from further had has hasn hasn't have haven haven't he hence her here hereby herein hereupon hers herself him himself his how however I ie i.e. if in inc inc. indeed instead into is isn isn't its itself it last later latterly least less let like likely ll ltd made make makes many maybe me meantime meanwhile might miss more moreover most mostly mr mrs much must my myself namely next no nobody none nonetheless nor not nothing now nowhere of off often on one only onto or others otherwise our ours ourselves out over own per perhaps rather re recent recently said same seem seemed seeming seems several she should shouldn shouldn't since so some somehow someone something sometime sometimes somewhere still such taking than that the their them themselves then thence there thereafter thereby therefore therein thereupon these they this those though through throughout thru thus to together too toward towards under unless unlike unlikely until up upon us used using ve very via was wasn we we well were weren weren't what whatever when whence whenever where whereafter whereas whereby wherein whereupon wherever whether which while who whoever whole whom whomever whose why will with within without would wouldn't yes yet you your yours yourself yourselves

Symbols: + * / - = ~ [| { } ( ) < > | \ _ ^ , . ; : & % $ # @ !

Appendix E

Words and Documents Frequencies
(Actual Frequencies – Page 1/2)

	AV-11-0*	AV-11-0	G-11-0	AV-7-1	AV-7-1	G-7-1	F-7-1	N-7-1
WORDS	WC	DC	DC	WC	DC	DC	DC	IC
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*
gulfweeds	6	5	11	11	7	9	15	214
syllabifying	15	13	32	20	17	81	15	42
supereffective	38	18	63	52	32	104	79	642
pluperfects	52	25	54	50	30	60	66	3,860
uncrates	68	33	108	64	53	124	109	866
glamorises	86	61	178	123	114	261	195	1,106
dumpish	108	67	232	94	73	207	190	260
sarsaparillas	135	63	157	128	68	183	167	11,828
caraways	161	83	607	269	130	337	295	241
brassily	193	133	686	185	158	384	322	332
bellyachers	228	130	361	241	164	425	372	462
condescensions	315	140	742	275	187	875	577	26,599
abscise	421	190	1,100	430	362	932	591	463
overlight	555	170	1,250	653	397	1,060	784	678
lunchers	718	552	1,240	752	693	1,570	1,195	3,171
crunchiness	1,030	802	1,880	1,346	1,224	3,090	1,864	1,535
victuallers	1,155	784	1,980	1,717	1,273	2,370	1,736	3,196
wreathe	1,451	977	3,070	1,453	1,137	3,310	2,201	5,624
railheads	1,608	1,196	2,540	2,072	1,697	3,250	2,591	10,300
Brahmanism	2,425	1,523	3,620	3,063	2,044	4,980	3,750	2,372
pandemics	3,581	2,117	4,460	3,819	2,738	6,420	4,462	39,403
kookaburras	4,301	2,812	6,700	7,069	4,648	7,910	5,810	15,445
inducts	4,692	3,344	7,030	5,921	4,646	10,800	7,233	24,144
titillate	5,137	4,205	9,270	5,426	5,114	11,800	8,146	8,243
reticulated	12,785	7,915	17,700	15,022	10,125	22,900	16,647	11,153
kneecap	13,824	8,617	23,400	17,341	12,284	31,400	21,358	26,619
chiming	17,615	11,162	26,100	18,439	14,603	34,900	24,550	19,506
hulled	20,578	12,771	23,400	22,145	15,902	33,600	25,309	21,393
medalists	24,111	14,180	39,000	26,564	18,500	36,700	25,116	126,775
racehorse	30,599	17,766	35,300	41,824	29,434	41,900	31,446	38,138
condiment	33,080	22,467	41,600	38,357	27,401	54,200	42,837	96,749
scaly	42,187	27,381	61,200	43,170	32,724	78,000	56,971	39,926
artichokes	49,566	24,514	51,700	56,526	31,517	69,600	52,624	91,898

Words and Documents Frequencies
(Actual Frequencies – Page 2/2)

waffles	54,029	31,932	73,000	70,869	49,628	112,000	76,667	129,887
benediction	63,885	39,630	74,000	56,696	42,294	87,400	66,733	69,872
headroom	82,706	43,740	87,500	75,925	50,488	119,000	89,492	65,063
laureate	108,378	47,005	125,000	134,085	94,767	164,000	129,186	146,521
pharisees	146,059	50,752	132,000	134,421	56,503	189,000	109,456	85,655
initiator	161,572	58,878	201,000	185,557	108,851	217,000	111,914	95,110
vibrating	218,984	91,010	330,000	341,988	128,112	292,000	155,184	119,675
deprived	312,299	208,016	401,000	350,022	277,158	590,000	449,730	339,293
pulp	718,114	299,980	723,000	904,492	373,876	995,000	575,940	506,059
Webb	847,371	366,895	1,140,000	2,551,324	676,417	1,570,000	940,656	742,810
calculate	1,533,500	385,020	1,520,000	1,912,439	510,701	2,600,000	1,461,123	1,288,704
fellowship	1,970,165	947,330	1,650,000	2,156,848	1,045,050	2,250,000	1,584,144	1,313,841
Lisa	2,612,814	1,272,560	3,280,000	4,143,723	2,065,232	4,150,000	2,899,548	2,145,476
notices	3,676,265	1,271,670	5,000,000	11,585,822	2,295,553	5,810,000	3,965,551	10,924,678
fa	5,629,389	1,963,215	4,300,000	6,774,765	2,915,749	5,660,000	2,797,752	3,026,572
selected	10,946,862	4,242,405	9,700,000	13,610,407	5,795,025	16,400,000	9,228,420	7,256,286
text	61,687,221	35,637,400	67,300,000	130,176,754	80,017,697	48,300,000	21,017,772	17,926,943

Explanations:

Column #	*Frequency Type*	*Engine-Date*
1	WC (Word Count)	AltaVista – November 2000
2	DC (Document Count)	AltaVista – November 2000
3	DC (Document Count	Google – November 2000
4	WC (Word Count)	AltaVista – July 2001
5	DC (Document Count	AltaVista – July 2001
6	DC (Document Count	Google – July 2001
7	DC (Document Count	Fast – July 2001
8	IC (Item/Document Count)	Northern Light – July 2001

Words and Documents Frequencies
(Spearman Footrule Distance Calculation Page 1/2)

WORDS	RANKING*								1 & 2	1 & 3	1 & 4	1 & 5	1 & 6	1 & 7	1 & 8	3 & 6	2 & 5
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*									
gulfweeds	1	1	1	1	1	1	1	2	0	0	0	0	0	0	1	0	0
syllabifying	2	2	2	2	2	3	2	1	0	0	0	0	1	0	1	1	0
supereffective	3	3	4	4	4	4	4	8	0	1	1	1	1	1	5	0	1
pluperfects	4	4	3	3	3	2	3	16	0	1	1	1	2	1	12	1	1
uncrates	5	5	5	5	5	5	5	10	0	0	0	0	0	0	5	0	0
glamorises	6	6	7	7	8	8	8	11	0	1	1	2	2	2	5	1	2
dumpish	7	8	8	6	7	7	7	4	1	1	1	0	0	0	3	1	1
sarsaparillas	8	7	6	8	6	6	6	21	1	2	0	2	2	2	13	0	1
caraways	9	9	10	11	9	9	9	3	0	1	2	0	0	0	6	1	0
brassily	10	11	11	9	10	10	10	5	1	1	1	0	0	0	5	1	1
bellyachers	11	10	9	10	11	11	11	6	1	2	1	0	0	0	5	2	1
condescensions	12	12	12	12	12	12	12	26	0	0	0	0	0	0	14	0	0
abscise	13	14	13	13	13	13	13	7	1	0	0	0	0	0	6	0	1
overlight	14	13	15	14	14	14	14	9	1	1	0	0	0	0	5	1	1
lunchers	15	15	14	15	15	15	15	14	0	1	0	0	0	0	1	1	0
crunchiness	16	17	16	16	17	17	17	12	1	0	0	1	1	1	4	1	0
victuallers	17	16	17	18	18	16	16	15	1	0	1	1	1	1	2	1	2
wreathe	18	18	19	17	16	19	18	17	0	1	1	2	1	0	1	0	2
railheads	19	19	18	19	19	18	19	19	0	1	0	0	1	0	0	0	0
Brahmanism	20	20	20	20	20	20	20	13	0	0	0	0	0	0	7	0	0
pandemics	21	21	21	21	21	21	21	29	0	0	0	0	0	0	8	0	0
kookaburras	22	22	22	24	23	22	22	22	0	0	2	1	0	0	0	0	1
inducts	23	23	23	23	22	23	23	25	0	0	0	1	0	0	2	0	1
titillate	24	24	24	22	24	24	24	18	0	0	2	0	0	0	6	0	0
reticulated	25	25	25	25	25	25	25	20	0	0	0	0	0	0	5	0	0
kneecap	26	26	26	26	26	26	26	27	0	0	0	0	0	0	1	0	0
chiming	27	27	28	27	27	28	27	23	0	1	0	0	1	0	4	0	0
hulled	28	28	27	28	28	27	29	24	0	1	0	0	1	1	4	0	0
medalists	29	29	30	29	29	29	28	38	0	1	0	0	0	1	9	1	0
racehorse	30	30	29	31	31	30	30	28	0	1	1	1	0	0	2	1	1
condiment	31	31	31	30	30	31	31	36	0	0	1	1	0	0	5	0	1
scaly	32	33	33	32	33	33	33	30	1	1	0	1	1	1	2	0	0
artichokes	33	32	32	33	32	32	32	34	1	1	0	1	1	1	1	0	0

Words and Documents Frequencies
(Spearman Footrule Distance Calculation Page 2/2)

waffles	34	34	34	35	35	35	35	39	0	0	1	1	1	1	5	1	1
benediction	35	35	35	34	34	34	34	32	0	0	1	1	1	1	3	1	1
headroom	36	36	36	36	36	36	36	31	0	0	0	0	0	0	5	0	0
laureate	37	37	37	37	38	37	39	40	0	0	0	1	0	2	3	0	1
pharisees	38	38	38	38	37	38	37	33	0	0	0	1	0	1	5	0	1
initiator	39	39	39	39	39	39	38	35	0	0	0	0	0	1	4	0	0
vibrating	40	40	40	40	40	40	40	37	0	0	0	0	0	0	3	0	0
deprived	41	41	41	41	41	41	41	41	0	0	0	0	0	0	0	0	0
pulp	42	42	42	42	42	42	42	42	0	0	0	0	0	0	0	0	0
Webb	43	43	43	45	44	43	43	43	0	0	2	1	0	0	0	0	1
calculate	44	44	44	43	43	45	44	44	0	0	1	1	1	0	0	1	1
fellowship	45	45	45	44	45	44	45	45	0	0	1	0	1	0	0	1	0
Lisa	46	47	46	46	46	46	47	46	1	0	0	0	0	1	0	0	1
notices	47	46	48	48	47	48	48	49	1	1	1	0	1	1	2	0	1
fa	48	48	47	47	48	47	46	47	0	1	1	0	1	2	1	0	0
selected	49	49	49	49	49	49	49	48	0	0	0	0	0	0	1	0	0
text	50	50	50	50	50	50	50	50	0	0	0	0	0	0	0	0	0

Average Spearman Footrule Distance	0.2	0.4	0.5	0.4	0.4	0.4	3.6	0.4	0.5

* **Explanations:**

Columns: **1 through 8 (same as in table above)**

Ranking: **Relative position of each word within its own engine (based on actual frequencies in table above)**

a & b **distances between lists a and b - abs($a_i - b_i$) for each item (i) in the list**

Appendix F

Operational Parameters for the Close End Algorithm (sample query)

PART A

Y =	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	M=9	M=9	M=9	M=9	M=8	M=8	M=8	M=8	M=7	M=7	M=7	M=7	M=6	M=6	M=6
	L=3	L=4	L=5	L=6	L=3	L=4	L=5	L=6	L=3	L=4	L=5	L=6	L=3	L=4	L=5
Number of subqueries	466	382	256	130	219	163	93	37	99	64	29	8	42	22	7

PART B

Result-1	459	379	256	130	213	163	93	37	94	64	29	8	42	22	7
Result-2	192	155	93	37	81	63	29	8	31	22	7	1	12	6	1
Result-3	62	57	29	8	26	20	7	1	26	20	7	0	11	6	0
Result-4	30	22	7	1	15	6	1	0	14	6	1	0	5	1	0
Result-5	28	20	7	1	14	6	1	0	13	6	1	0	5	1	0
Result-6	26	19	7	1	13	6	1	0	11	5	1	0	4	1	0
Result-7	24	17	7	1	11	5	1	0	10	5	1	0	4	0	0
Result-8	23	17	6	1	10	5	1	0	8	4	1	0	3	0	0
Result-9	19	15	6	1	10	5	1	0	5	3	0	0	3	0	0
Result-10	19	14	6	1	8	5	1	0	5	3	0	0	1	0	0
Result-11	18	14	6	1	7	5	1	0	5	1	0	0	1	0	0
Result-12	17	13	5	1	6	5	1	0	4	1	0	0	1	0	0
Result-13	16	13	5	1	5	3	1	0	4	1	0	0	1	0	0
Result-14	16	13	5	1	5	3	1	0	4	1	0	0	1	0	0
Result-15	15	12	2	1	5	3	1	0	4	1	0	0	1	0	0
Result-16	13	11	1	1	4	3	1	0	3	1	0	0	1	0	0
Result-17	11	9	1	0	4	3	1	0	3	1	0	0	1	0	0
Result-18	10	8	1	0	4	3	1	0	3	1	0	0	1	0	0
Result-19	10	6	1	0	4	3	1	0	3	1	0	0	1	0	0
Result-20	8	6	1	0	4	3	1	0	3	1	0	0	1	0	0

Appendix G

Experiment Guide

DESCRIPTION: Instructions to conduct retrieval and evaluation experiments:

Execute Web searches using various Search Engines:

Evaluate search results:

Save experiment's results.

ATTACHMENTS: A diskette (zipped) with the following three subdirectories:

Assignment – containing MS Excel worksheet with assignments:

Queries – containing 50 MS Word files with the text of long queries:

Results – containing MS Excel workbook template to collect results.

PROCESSES:

I Preliminary process:

1. Unzip the diskette and save all files on your hard drive.
2. Find your *Group-ID* and *Participant-ID* in the assignment sheet and examine experiments you are assigned to conduct.

II Administrative Process:

3. Open the MS Excel template workbook and enter your *Group-ID* and *Participant-ID*, the *Query#* and *Experiment-ID* in the designated cells in the first worksheet called "ID". The sheet will return instructions to save your workbook and it will also provide you with a name to save it under (save it in the RESULTS directory).

III Retrieval and Evaluation Process:

4. Open the assigned MS Word file (e.g. ***AGR_1.doc*** in the QUERIES subdirectory):
5. Read the query text carefully. Spend time to understand the content and select search term in the text, which best describe its meaning.

III-A Short Query Process

6. Create a short query (any length) that you will submit to search engines:
7. Submit the short query, which you just created, to a designated search engine.
8. Evaluate every site (top 20 only) and determine relevancy of each site according to the following ternary scale:

Relevancy	*Description*
0	**Not relevant** (different topics, no common terms, includes references to missing or restricted access sites)
1	**Somewhat relevant** (mentions same topic as the query, uses common terms, has links to sites deemed relevant by prior searches)
2	**Very relevant** (original document, describes same or very similar topics, closely matches the query, uses query terms extensively, has many links to sites deemed very relevant by prior searches)

Table G-1 Relevancy Scale

9. Accurately copy into appropriate search engine's worksheet the top 20 URL addresses returned by the search engine. **Copy URLs for RELEVANT sites ONLY.** Enter the relevancy judgments in the column next to site addresses. If the search engine returns less then 20 results, use all of them. The spreadsheet will automatically "decide" that all "missing" sites are irrelevant.

III-B Long Query Process

10. Edit the text by deleting the file name (from the top of the page) and all Carriage Return (CR) characters (the majority of search engines interpret the first CR character as an end of the query).
11. "Select All" and "Copy" the entire text to the Clipboard. "Paste" it from the Clipboard into the search engine query window. Start the search.
12. If the query is too long, the search engine will come back with an error message. Copy this error message into the first URL field for that query.
13. To evaluate search results, continue as specified in paragraph 8 and 9 above.

III-C Long Query Process (LQSE)

14. Edit the text by deleting the file name only.
15. "Select All" and "Copy" the entire text to the Clipboard. "Paste" it from the Clipboard into the Long Query Search Engine large query window. First execute the search using the Open End Query (OEQ) and then repeat the search using the Close End Query (CEQ).
16. To evaluate search results, continue as specified in paragraphs 8 and 9 above.

III-D Feedback Process

17. Click the "Refine" button. You will be presented with a list of search terms and a small window to create or enter phrases. Use your judgment to re-order the search terms and or to create new phrases. Restart the search.
18. To evaluate the new search results, continue as specified in paragraphs 8 and 9 above.
19. Repeat the paragraphs 17 and 18 above as often as needed.

Appendix H

Collection of Long Queries (samples)

Family Life (FAM-2) A 23-year, seven-wave study of 867 families in the Detroit area found that children whose parents divorced were more likely to endorse premarital sex, approve of cohabitation, express negative attitudes towards marriage, and accept divorce, compared to children who lived with both parents. In another study, after evaluating a sample of 12,537 young people, the results showed that "parental divorce increases the likelihood that young men and women will have a child out-of-wedlock."

Medicine (MED-1) Nutrition behaviors are governed by health beliefs such as risk perceptions, outcome expectancies, and optimistic self-beliefs. The present study deals with the role that objective criteria such as age and body weight might play in forming subjective beliefs. The question is whether they can deter people from forming an overly optimistic judgment of their health risk.

Six kinds of verbal judgments were assessed, namely self-reported health, vulnerability towards cardiovascular diseases, nutrition outcome expectancies, nutrition self-efficacy, intentions to change one's diet, and reported nutrition behaviors. In a sample of 1,583 men and women between 14 and 87 years of age, these judgments were statistically related to age and body weight. It was found that people do take their objective risk status into account, but only to a certain degree. The self-serving bias continues to exist throughout all age groups and weight levels. Moreover, it was found that individuals report better intentions to adhere to healthy foods and better nutrition behaviors as they grow older and gain weight.

Environment & Natural Resources (ENR-2) This article explains the concepts of economic and resource rent in the area of natural resources as a basis for fiscal and tax policies by governments, and the relationship to US foreign tax credit policy and tax law. The article examines how the concepts of economic and resource rent are applied to tax the development of petroleum resources under several petroleum fiscal regimes. The changing parameters of a US creditable tax are discussed.

Appendix I

Categorization of Long Queries (Page 1 of 2)

CATE-GORY	DISCIP-LINE	FILE ID	QUERY NUMB.	NUMBER OF WORDS	NUMBER OF CHARACTERS
A Applied Science & Technology (9)	Agriculture	AGR_1	1	269	1,690
		AGR_2	2	137	867
	Computer Science	CSC_1	11	692	4,148
		CSC_2	12	889	6,215
		CSC_3	13	289	1,837
	Enginee-ring	ENG_1	14	223	1,702
		ENG_2	15	251	1,641
	Environm. & Nat. Rsr	ENR_1	16	53	343
		ENR_2	17	89	553
B Business & Finance (13)	Business	BUS_1	7	132	1,015
		BUS_2	8	176	1,189
	Finance	FIN_1	20	107	776
		FIN_2	21	160	1,069
	Manage-ment	MNG_1	39	147	969
		MNG_2	40	195	1,313
	Manufac-turing	MAN_1	31	87	513
		MAN_2	32	62	382
		MAN_3	33	144	872
	Marketing	MAR_1	34	481	3,466
		MAR_2	35	185	1,179
	Transpor-tation	TRN_1	47	152	1,104
		TRN_2	48	176	1,171

Categorization of Long Queries (Page 2 of 2)

CATE-GORY	DISCIP-LINE	FILE ID	QUERY NUMB.	NUMBER OF WORDS	NUMBER OF CHARACTERS
H Humanities (10)	Anthro-pology	ANT_1	3	130	930
		ANT_2	4	178	1,222
	Crime	CRM_1	9	248	1,609
		CRM_2	10	211	1,449
	Family Life	FAM_1	18	134	985
		FAM_2	19	75	497
	Psychology	PSY_1	43	79	622
		PSY_2	44	110	735
	Social work	SWK_1	45	100	690
		SWK_2	46	96	779
L Liberal Arts (9)	Arts	ART_1	5	139	934
		ART_2	6	81	608
	History	HIS_1	24	133	709
		HIS_2	25	91	616
	Literature	LIT_1	26	173	927
		LIT_2	27	95	539
		LIT_3	28	152	761
		LIT_4	29	282	1,662
		LIT_5	30	85	492
S Science & Medicine (9)	Geography	GEO_1	22	123	874
		GEO_2	23	199	1,279
	Medicine	MED_1	36	167	1,104
		MED_2	37	146	1,040
		MED_3	38	101	727
	Physiology	PHY_1	41	188	1,280
		PHY_2	42	197	1,192
	Zoology	ZOO_1	49	176	1,171
		ZOO_2	50	156	1,026

A, B, H, L, S - Category Codes

(N) Numeral in parenthesis indicates number of long queries in this category.

Appendix J

Data Collection Workbook (Id Page)

Enter your Group and Participant Id here ----------->>>	A Group	DE Prtcpnt
Enter 2-character Experiment Id (from the table below)----->>>		AS Exper Id
Enter 2-digit Query # here ----------->>>		33 Query #

Query Category --------->>>	**Business and Finance**
Query Category Code-->>>	**B**
Query Discipline -------->>>	**Manufacturing**
Query File XXX-N.doc->>>	**MAN_3.doc**

Save this worksheet as specified below:

AS_B_33_MAN_3_A_DE.xls

EXPERIMENTS TABLE		
EXP. ID	*SEARCH ENGINE*	*COMMENT*
AL	AltaVista	Long Query
AS	AltaVista	Short Query
CY	LQSE (close)	Long Query
FL	Fast	Long Query
FS	Fast	Short Query
GL	Google	Long Query
GS	Google	Short Query
NL	Northern Light	Long Query
NS	Northern Light	Short Query
OX	LQSE (open)	Long Query

Data Collection Workbook (Data Page)

Group Id	Participant Id	Test Id	Query Number	Query Category Code	Query File Name	SE Rank	Search Engine Results	Relevancy
A	DE	AS	33	B	MAN_3	1	http://www.brownmfgcorp.com/cutter/index.html	1
A	DE	AS	33	B	MAN_3	2		0
A	DE	AS	33	B	MAN_3	3	http://www.chelt.ac.uk/el/philg/gdn/jg/1998.htm	2
A	DE	AS	33	B	MAN_3	4	http://odur.let.rug.nl/~usa/E/dred_scott/scott02.htm	1
A	DE	AS	33	B	MAN_3	5		0
A	DE	AS	33	B	MAN_3	6		0
A	DE	AS	33	B	MAN_3	7	http://xroads.virginia.edu/~HYPER/DREISER	2
A	DE	AS	33	B	MAN_3	8	http://etext.lib.virginia.edu/etcbin	1
A	DE	AS	33	B	MAN_3	9		0
A	DE	AS	33	B	MAN_3	10	http://www.litrix.com/cristo/crist001.htm#1	1
A	DE	AS	33	B	MAN_3	11		0
A	DE	AS	33	B	MAN_3	12	http://www.vulcanpub.com/hen/	1
A	DE	AS	33	B	MAN_3	13		0
A	DE	AS	33	B	MAN_3	14		0
A	DE	AS	33	B	MAN_3	15		0
A	DE	AS	33	B	MAN_3	16		0
A	DE	AS	33	B	MAN_3	17	http://userpage.fu-berlin.de/~health/preprint.htm	2
A	DE	AS	33	B	MAN_3	18		0
A	DE	AS	33	B	MAN_3	19		0
A	DE	AS	33	B	MAN_3	20		0

Appendix K

List of Experiment Configurations

Configuration	*Search Engine Used*	*Comment*
AL	AltaVista (Long query)	Long Query is submitted by the user
AS	AltaVista (Short query)	Short Query is formulated by the user
CA	LQSE (Close end)	Close End algorithm (M = 9, L = 3)
CB	LQSE (Close end)	Close End algorithm (M = 9, L = 4)
CC	LQSE (Close end)	Close End algorithm (M = 9, L = 5)
CE	LQSE (Close end)	Close End algorithm (M = 8, L = 3)
CF	LQSE (Close end)	Close End algorithm (M = 8, L = 4)
CI	LQSE (Close end)	Close End algorithm (M = 7, L = 3)
CJ	LQSE (Close end)	Close End algorithm (M = 7, L = 4)
CM	LQSE (Close end)	Close End algorithm (M = 6, L = 3)
CS	LQSE (Close end)	With SAC algorithm
CU	LQSE (Close end)	Close End algorithm (with Feedback))
CY	LQSE (Close end)	Close End algorithm (without Feedback)
FL	Fast (Long query)	Long Query is submitted by the user
FS	Fast (Short query)	Short Query is formulated by the user
GL	Google (Long query)	Long Query is submitted by the user
GS	Google (Short query)	Short Query is formulated by the user
NL	Northern Light (Long query)	Long Query is submitted by the user
NS	NorthernLight (Short query)	Short Query is formulated by the user
O3	LQSE (Open end)	Open End Algorithm (3 subqueries)
O4	LQSE (Open end)	Open End Algorithm (4 subqueries)
O5	LQSE (Open end)	Open End Algorithm (5 subqueries)
O6	LQSE (Open end)	Open End Algorithm (6 subqueries)
OU	LQSE (Open end)	Open End algorithm (with Feedback)
OX	LQSE (Open end)	Open End algorithm (without Feedback)

Appendix L Position of the Original Document (Top 20 Search Results)

Configuration	*Top Document*	*Top 5 Documents*	*Top 10 Documents*	*Top 20 Documents*
AL				
AS				
CA				
CB				
CC				
CE				
CF				
CI				
CJ				
CM				
CS				
CU				
CY				
FL				
FS				
GL				
GS				
NL				
NS				
O3				
O4				
O5				
O6				
OU				
OX				

Appendix M

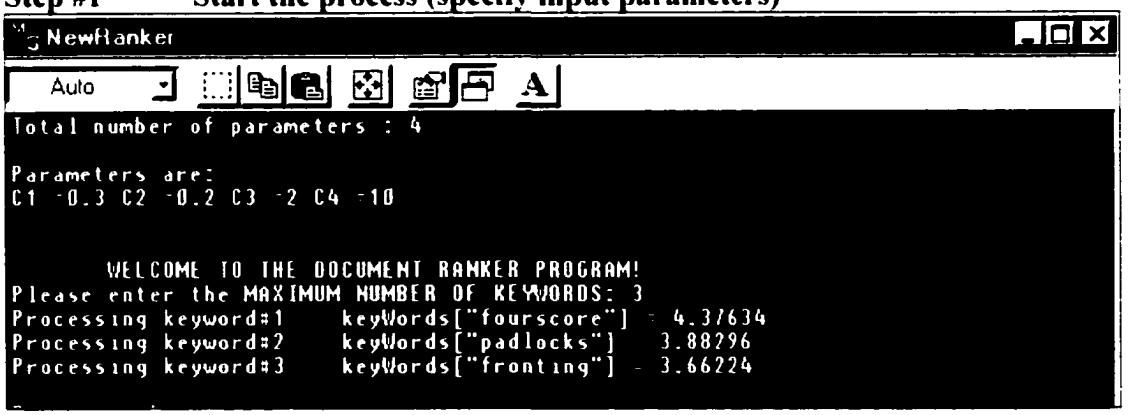
Precision @ 10 & Precision @ 20

	Precision @10	*Precision @20*
Configuration		
AL		
AS		
CA		
CB		
CC		
CE		
CF		
CI		
CJ		
CM		
CS		
CU		
CY		
CY		
FL		
FS		
GL		
GS		
NL		
NS		
O3		
O4		
O5		
O6		
OU		
OX		
OX		

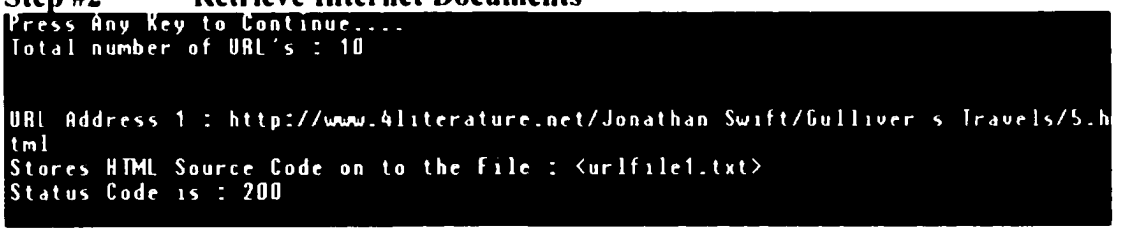
Appendix N

SAC Algorithm Interface

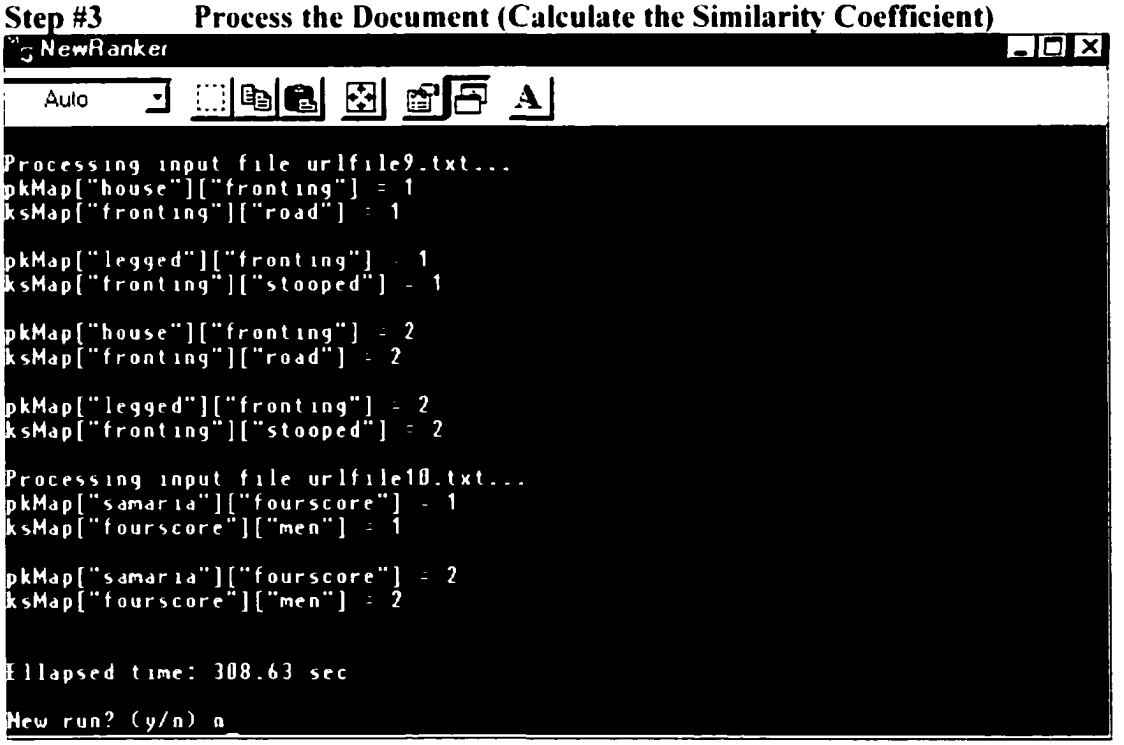
Step #1 Start the process (specify input parameters)



Step #2 Retrieve Internet Documents



Step #3 Process the Document (Calculate the Similarity Coefficient)



Appendix O

SAC algorithm (Input/Output Data)

Table # 1 *Input to the SAC algorithm (list of the URLs Produced by the MPC algorithm)*

1	http://www.4literature.net/Jonathan_Swift/Gulliver_s_Travels/5.html
2	http://www.lwc.edu/staff/mlund/gulliver.html
3	http://www.library.utoronto.ca/utel/fiction_u/swiftj_gt/gt1_all.html
4	http://www.geocities.com/soho/nook/7255/GULLIVER.html
5	http://www.jaffebros.com/lee/gulliver/bk1/chap1-1.html
6	http://ftp.jp.gnome.org/pub/anoncvs.gnome.org/notyop/lexicons/lexicon.crappy
7	http://www.inform.umd.edu/EdRes/ReadingRoom/Fiction/Swift/Gulliver/b1ch1.txt
8	http://www.library.utoronto.ca/utel/fiction_u/swiftj_gt/gt1_ch1.html
9	http://www.lang.nagoya-u.ac.jp/~matsuoka/EG-Grey.html
10	http://www.bryanstation.com/beard.htm

Table 2 *Output Produced by the SAC Algorithm (the list is ranked in reverse order of similarity coefficients)*

http://www.4literature.net/Jonathan_Swift/Gulliver_s_Travels/5.html	0.1830
http://www.jaffebros.com/lee/gulliver/bk1/chap1-1.html	0.1697
http://www.library.utoronto.ca/utel/fiction_u/swiftj_gt/gt1_ch1.html	0.1693
http://www.inform.umd.edu/EdRes/ReadingRoom/Fiction/Swift/Gulliver/b1ch1.tx	0.1392
http://www.library.utoronto.ca/utel/fiction_u/swiftj_gt/gt1_all.html	0.1341
http://www.geocities.com/soho/nook/7255/GULLIVER.html	0.1168
http://www.lwc.edu/staff/mlund/gulliver.html	0.1101
http://ftp.jp.gnome.org/pub/anoncvs.gnome.org/notyop/lexicons/lexicon.crappy	0
http://www.lang.nagoya-u.ac.jp/~matsuoka/EG-Grey.html	0
http://www.bryanstation.com/beard.htm	0

11 Bibliography

A00 **R. K. Ando**
Latent Semantic Space: Iterative Scaling Improves Precision of Inter–document Similarity Measurement.
Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval; Athens, Greece; July 24 - 28, 2000; pp. 216 - 223

B99 **C. Baumgarten**
A Probabilistic Solution in the Selection and Fusion Problem in Distributed Information Solution.
Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval; Berkeley, CA; August 15 - 19, 1999; pp. 246 - 253

BB98 **K. Bharat, A. Broder**
A Technique for Measuring the Relative Size and Overlap of Public Web Search Engines.
Proceedings of the 7th International World Wide Web Conference, Brisbane, Australia, April 14 - 18, 1998
http://www-sor.inria.fr/mirrors/www7/programme/fullpapers/1937/com1937.htm

BCCC93 **N. J. Belkin, C. Cool, W. B. Croft, J. P. Callan**
The Effect of Multiple Query Representation on Information Retrieval Performance.
Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval; Pittsburgh, PA; June 27 - July 1, 1993; pp. 339 - 346

BKFS95 **N. J. Belkin, P. B. Kantor, E. A. Fox, J. A. Shaw**
Combining the Evidence of Multiple Query Representations for Information Retrieval.
Information Processing and Management, 31(1), 1995; pp. 431–448

BP98 **S. Brin, L. Page**
The Anatomy of a Large-Scale Hypertextual Web Search Engine.
Proceedings of the 7th International World Wide Web Conference: Brisbane, Australia: April 14 - 18, 1998
http://www7.scu.edu.au/programme/fullpapers/1921/com1921.htm

BR99 **R. Baeza-Yates, B. Ribeiro-Neto**
Modern Information Retrieval.
Addison Wesley, 1999

BV00 **C. Buckley, E. M. Voorhees**
Evaluating Evaluation Measure stability
Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: Athens, Greece: July 24 - 28, 2000: pp. 33 - 40

CCH92 **J. P. Callan, W. B. Croft, S. M. Harding**
The InQuery Retrieval System.
Proceedings of the 3rd International Conference on Database and Expert System Applications: Valencia, Spain: September 1992.
http://santana.uni-muenster.de/Library/VirtualInformationRetrieval/

CDRRGK 98 **S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, J. Kleinberg**
Automatic Resource Compilation by Analyzing Hyperlinkage Structure and Associated Text.
Proceedings of the 7th International World Wide Web Conference: Brisbane, Australia: April 14 - 18, 1998
http://www7.scu.edu.au/programme/fullpapers/1898/com1898.html

CGP98 **J. Cho, H. Garcia-Molina, L. Page**
Efficient Crawling Through URL Ordering.
Proceedings of the 7th International World Wide Web Conference: Brisbane, Australia: April 14 - 18, 1998
http://www7.scu.edu.au/programme/fullpapers/1919/com1919.htm

D00 **B. D. Davison**
Topical Locality in the Web.
Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: Athens, Greece: July 24 - 28, 2000: pp. 272 - 279

DKNS01 **C. Dwork, R. Kumar, M. Naor, D. Sivakumar**
Rank Aggregation Methods for the Web.
Proceedings of the10th International World Wide Web Conference: Hong Kong, May 2 - 5, 2001
http://www10.org/cdrom/papers/577/index.html

F98 **S. Feldman**
Web Search Services in 1998: Trends and Challenges.
Searcher, Volume 6, Number 6: June 1998
http://www.infotoday.com/searcher/jun98/story2.htm

F99 **S. Feldman**
NLP Meets the Jabberwocky: Natural Language Processing in Information Retrieval.
ONLINE: May 1999
http://www.onlineinc.com/onlinemag/OL1999/feldman5.html

FM00 **M. Franz, J. S. McCarley**
Word Document Density and Relevance Scoring.
Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: Athens, Greece: July 24 - 28, 2000: pp. 345 - 347

FSTV99 **V. Frants, J. Shapiro, I. Taksa, V. Voiskunskii**
Boolean Search: Current State and Perspectives.
JASIS, 50(1) 1999: pp. 86 - 95

FSV97 **V. Frants, J. Shapiro, V. Voiskunskii**
Automated Information Retrieval: Theory and Methods.
Academic Press, 1997

G00 **J. Garofolo**
E–mail Correspondence.
National Institute of Standards and Technology: 2000
http: www.nist.gov itl div896 emaildir sdr_list msg00055.html
http: www.nist.gov itl div896 emaildir sdr_list msg00056.html
http: www.nist.gov itl div896 emaildir sdr_list msg00057.html

G01 http: www.google.com

GAV00 **J. Garofolo, C. Auzanne, E. Voorhees**
The TREC Spoken Document Retrieval Track: A Success Story.
National Institute of Standards and Technology: 2000
http: www.nist.gov speech tests sdr sdr2000 papers 01plenary1.pdf

GPS99 **G. Golovchinsky, M. N. Price, B. N. Schiff**
From Reading to Retrieval: Freeform Ink Annotations as Queries.
Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: Berkeley, CA: August 15 - 19, 1999: pp. 19 25

H90 **S. P. Harter**
Search Term Combinations and Retrieval Overlap: A Proposed Methodology and Case Study
JASIS, 41(2) 1990: pp. 132 146

H92 **D. K. Harman**
Relevance Feedback Revisited.
Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: Copenhagen, Denmark: June 21 - 24, 1992
http: www.acm.org pubs contents proceedings ir 133160

HB96 **S. Henninger, N. Belkin**
Interface Issues and Interaction Strategies for Information Retrieval Systems.
Proceedings of Human Factors in Computing Systems Conference: Vancouver, Canada: April 13 - 18, 1996
http: www.acm.org pubs contents proceedings chi 238386

HCH99 **D. Hawking, N. Craswell, D. Harman**
Results and Challenges in Web Search Evaluation.
Proceedings of the 8th International World Wide Web Conference: Toronto, Canada: May 11 - 14, 1999
http://www8.org/w8-papers-2c-search-discover/results/results.html

HHMN00 **M.R. Henzinger, A. Heydon, M. Mitzenmacher, M. Najork**
On Near-Uniform URL Sampling.
Proceedings of the 9th International World Wide Web Conference: Amsterdam, Holland: May 15 - 19, 2000
http://www9.org/w9cdrom/88/88.html

HHMN99 **M.R. Henzinger, A. Heydon, M. Mitzenmacher, M. Najork**
Measuring Index Quality Using Random Walks on the Web.
Proceedings of the 8th International World Wide Web Conference: Toronto, Canada: May 11-14, 1999
http://www8.org/w8-papers-2c-search-discover/measuring/measuring.html

HMIH99 **K. Hoashi, K. Matsumoto, N. Inoue, K. Hashimoto**
Query Expansion Method Based on word Contribution.
Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: Berkeley, CA: August 15 - 19, 1999: pp. 303 - 304

J00 **B. J. Jansen,**
The Effect of Query Complexity on Web Searching Results.
University of Maryland: 2000
http://www.shef.ac.uk/~is/publications/infres/paper87.html#ref2

JK00 **K. Järvelin, J. Kekäläinen**
IR Evaluation Methods for Retrieving Highly Relevant Documents.
Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: Athens, Greece: July 24 - 28, 2000: pp. 41 - 48

JP00 **B. J. Jansen(^), U. Pooch (*)**
A Review of Web Searching Studies and a Framework for Future Research.
(^)University of Maryland, (*)Texas A&M University: 2000
http://jimjansen.tripod.com/academic/pubs/wus.html

JS00 **B.J. Jansen, A. Spink**
Methodological Approach in Discovering User Search Patterns Through Web Log Analysis.
American Society for Information Science.
Bulletin of American Society for Information Science: Oct Nov 2000
http: jimjansen.tripod.com academic pubs asis1500.html

JS99 **S. Johnes, M. Staveley**
Phrasier: a System for Interactive Document Retrieval Using Keyphrases.
Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Berkeley, CA: August 15 - 19, 1999: pp. 160 - 167

JSS99 **B. J. Jansen, A. Spink, T. Saracevic**
The Use of Relevance Feedback on the Web: Implications for Web IR System Design.
Proceedings of World Conference of the WWW and Internet: Honolulu, Hawaii, Oct. 25 - 30, 1999
http: jimjansen.tripod.com academic pubs webnet99.html

K00 **J. Kennon**
Sizing the Internet.
Study by Cyveillance®: 2000
http: www.cyveillance.com newsroom pressr 000710.asp

L97 **J. H. Lee**
Analysis of Multiple Evidence Combination.
Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: Philadelphia, PA: July 27 - 31, 1997: pp. 267 - 275

LG98 **S. Lorence, C.L. Giles.**
Inquirus, the NECI meta search engine.
Proceedings of the 7th International World Wide Web Conference: Brisbane, Australia: April 14 - 18, 1998
http: www7.scu.edu.au programme fullpapers 1906 com1906.htm

LP99 **E. de Lima, J. O. Pedersen**
Phrase Recognition and Expansion for Short, Precision–biased Queries Based on a Query Log.
Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: Berkeley, CA: August 15 - 19, 1999: pp. 145 - 152

LS98 **W. Li, J. Shim**
Facilitating Complex Web Queries Through Visual User Interfaces and Query Relaxation.
Proceedings of the 7th International World Wide Web Conference: Brisbane, Australia: April 14 - 18, 1998
http://www7.scu.edu.au/programme/fullpapers/1936/com1936.htm

M90 **S. Miller**
Experimental Design and Statistics.
Routledge Kegan & Paul, 1990

M97 **M. Marchiori**
The Quest for Correct Information on the Web: Hyper Search Engines.
Proceedings of the 6th International World Wide Web Conference: Santa Clara, CA: April 7 - 11, 1997
http://www.w3.org/People/Massimo/papers/WWW6/

MB00 **R. McArthur, P.D. Bruza**
The Ranking of Query Refinements of Interactive Web–based Retrieval.
Proceedings of the Information Doors Workshop (held in conjunction with the ACM Hypertext and Digital Libraries Conferences): 2000
http://www.guidebeam.com/infodoors.pdf

MBDH98 **S. A. Macskassy, A. Banerjee, B. D. Davison, H. Hirsh**
Human Performance on Clustering Web Pages: A Preliminary Study.
Proceedings of 4th International Conference on Knowledge Discovery and Data Mining: New York, NY: August 27 - 31,1998.
http://www.cs.rutgers.edu/~davison/pubs/destr355.html

MBSC97 **M. Mitra, C. Buckley, A. Singhal, C. Cardie**
An Analysis of Statistical and Syntactic Phrases.
Proceedings of 5th RIAO Conference: Montreal, Canada: June 25 - 27, 199?
http: www.research.att.com singhal pubsorted.html

N99 **R. Nordlie**
"User Revealment" - a Comparison of Initial Queries and Ensuing Question Development in Online Searching and in Human Reference Interactions.
Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: Berkeley, CA: August 15 - 19, 1999: pp. 11 - 18

NO00 **M. Narita, Y. Ogawa**
The Use of Phrases from Query Text for Information Retrieval.
Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: Athens, Greece: July 24 - 28, 2000: pp. 318 - 320

PBRC00 **J. Prager, E. Brown, D. R. Radev, K. Czuba**
One Search Engine or Two Question-Answering.
TREC 9 QA-Track, Notebook Paper, NIST, Gaithersburg, MD, 2000
http: trec.nist.gov pubs trec9 papers Prager Trec9notebook.pdf

PH97 **A. Pollock, A. Hockley**
What's Wrong with Internet Searching?
D-Lib Magazine: March 199?
http: www.dlib.org dlib march97 bt 03pollock.html

PW00[1] **T. A. Phelps, R. Wilensky**
Robust Hyperlinks Cost Just Five Words Each.
UCB Computer Science Technical Report UCB CSD-00-1091: January 10, 2000
http: http.cs.berkeley.edu wilensky robust hyperlinks.html

PW00[2] **T. A. Phelps, R. Wilensky**
Robust Hyperlinks: Cheap, Everywhere, Now.
Lecture Notes in Computer Science
Springer-Verlag, 2000

RMMH00 **H. Reiterer, G. Mußler, T. M. Mann, S. Handschuh**
INSYDER – An Information Assistant for Business Intelligence.
Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: Athens, Greece; July 24 - 28, 2000: pp. 112 - 119

S01 **D. Sullivan, Editor**
Search Assistance Features.
Search Engine Watch: 2001
http://searchenginewatch.com/facts/math.html

S98 **A. F. Smeaton**
Independence of Contributing Retrieval Strategies in Data Fusion for Effective Information Retrieval.
The 20th BCS Colloquium on Information Retrieval: Grenoble, France, March 25-27, 1998
http://www.compapp.dcu.ie/~asmeaton/pubs/DataFusionCRC.PDF

S99 **D. Sullivan, Editor**
AltaVista's Automatic Phrase Searching.
Search Engine Watch: 1999
http://searchenginewatch.com/sereport/99/02-avphrase.html

SBC97 **B. Shneiderman, D. Byrd, W. B. Croft**
Clarifying Search: A User-Interface Framework for Text Searches.
D-Lib Magazine, January 1997
http://www.dlib.org/dlib/january97/retrieval/01shneiderman.html

SHMM98 **C. Silverstein, M. Henzinger, H. Marais, M. Moricz**
Analysis of a Very Large AltaVista Query Log.
SRC Technical Note: 1998-014
http://gatekeeper.dec.com/pub/DEC/SRC/technical-notes/abstracts/src-tn-1998-014.html

SK88 **T. Saracevic, P. Kantor**
A Study of Information Seeking and Retrieving.
Part III. Searchers, Searches, and Overlap.
JASIS, 39(3) 1988: pp.197 - 216

SP99 **A. Singhal, F. Pereira**
Document Expansion for Speech Retrieval.
Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: Berkeley, CA: August 15 - 19, 1999: pp. 34 - 41

SX00 **A. Spink, J.L. Xu**
RESEARCH NOTE: Selected Results from a Large Study of Web Searching: the Excite Study.
Information Research, Volume 6 No. 1: October 2000
http://www.shef.ac.uk/is/publications/infres/paper90.html

SYAS97 **R. G. Sumner, Jr., K. Yang, R. Akers, W. M. Shaw, Jr.**
Interactive Retrieval using IRIS: TREC-6 Experiments.
Proceedings of the Sixth Text REtrieval Conference (TREC-6): Gaithersburg, MD: November 19 - 21, 1997
http://www.ils.unc.edu/unc_trec6

USPTO99 **Infoseek Corporation (Sunnyvale, CA)**
United States Patent & Trademark Office.
Real-time document collection search engine phrase indexing.
US Patent 5,920,854: July 6, 1999

V00 **P. Vakkari**
Relevance and Contributing Information Types of Searched Documents in Task Performance.
Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval: Athens, Greece: July 24 - 28, 2000: pp. 2 - 9

VH97 **E. Voorhees, D.K. Harman**
Overview of the Sixth Text Retrieval Conference TREC-6.
Proceedings of the Sixth Text REtrieval Conference (TREC-6): Gaithersburg, MD: November 19 - 21, 1997.
http://trec.nist.gov/pubs/trec6/t6_proceedings.html

W00 **D. Wolfram**
A Query-Level Examination of End User Searching Behavior on the Excite Search Engine.
Proceedings of the 28th Annual Conference CAIS 2000: Edmonton, Alberta, Canada: May 28 - 30, 2000
http://www.slis.ualberta.ca/cais2000/wolfram.htm

WJ99 **P. William, M. John**
Statistical Design and Analysis of Experiments.
Society for Industrial & Applied Mathematics (SIAM), 1999

WP97 **M.R. Wulfekuhler, W. F. Punch.**
Finding Salient Feature for Personal Web Page Categories.
Computer Networks and ISDN Systems, 29(1997): pp. 1147 - 1156

X99 **J. L. Xu**
Internet Search Engines: Real World IR Issues and Challenges.
Proceedings of CIKM 9: Kansas City, MI: October 31 - November 4, 1999
http://www.acm.org/pubs/contents/proceedings/cikm/319950

YMMS98 **K. Yang, K. Maglaughlin, L. Meho, R.G. Sumner, Jr.**
IRIS at TREC-7.
Proceedings of the seventh Text REtrieval Conference (TREC-7): Gaithersburg, MD: November 4 - 6, 1998:
http://trec.nist.gov/pubs/trec7/t7_proceedings.html

ZM98 **J. Zobel, A. Moffat**
Exploring the Similarity Space.
ACM SIGIR Forum, 35(1) 1998: pp. 18 - 34