The Pennsylvania State University

The Graduate School

College of Engineering

USING TOPOLOGICAL CONSTRUCTS TO MODEL INTERACTIVE INFORMATION RETRIEVAL DIALOGUE IN THE CONTEXT OF BELIEF, DESIRE, AND INTENTION THEORY

A Thesis in

Industrial Engineering

by

Craig Alan Nowack

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2005

This thesis of Craig Alan Nowack was reviewed and approved* by the following:

Soundar R.T. Kumara
Distinguished Professor of Industrial Engineering
Thesis Co-Adviser
Chair of Committee

Christopher C. Byrne
Assistant Professor of Mathematics
Thesis Co-Adviser
Special Member

Natarajan Gautam
Associate Professor of Industrial Engineering

Shankar Sundaresan
Assistant Professor of Supply Chain and Information Systems

Richard J. Koubek
Professor of Industrial Engineering
Head of the Department of Industrial and Manufacturing Engineering

*Signatures are on file in the Graduate School

ABSTRACT

An internet search with a modern search engine involves iterations of composing a query, submitting the query to the search engine via a text box, reviewing the returned search results list for relevant web pages, and visiting potentially relevant web pages. Often times, one iteration of search steps does not satisfy a search engine user's information need, especially if the user knows few keywords that are relevant to the information need. After a few search iterations, the user's patience and determination levels will be tested.

For the purpose of making information searches on the internet more efficient and productive, in this thesis a distance measure that takes advantage of past user queries is developed. In presenting the distance measure, the internet search process is described as a dialogue between the user and the search engine and methods are defined in the context of Belief, Desire, and Intention Theory. The distance measure is formulated using topology and the measure calculates distances between queries based on the search results of those queries. The tested hypothesis is that mathematical modeling can be used to express relationships between user behavior and search engine performance..

To test the hypothesis, analysis was performed on data from a survey administered to 39 subjects. The survey gathered relevant subject background information, queries related to two different search tasks that were presented to the subjects, and subjects' intentions behind their queries with regards to recall and precision. A total of 390 queries were collected from the subjects, of which 334 queries were

unique. The proposed distance measure was applied to the search results of those queries and statistical analysis was performed on the survey data to find correlations between query distances and subject background information, experience, and intentions. The results indicate that query surveys show potential for learning more about search engine user behavior and that the proposed distance measure shows potential as a basis for developing internet search tools.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I wish to acknowledge all of my committee members for their invaluable assistance on this thesis. Natarajan Gautam, Shankar Sundaresan, and Richard J. Koubek all provided constructive and inspiring feedback. I am especially appreciative of my co-advisers, Soundar R.T. Kumara and Christopher C. Byrne. Their encouragement and unique insight provided support without which this thesis would not have been possible.

I wish to acknowledge my brother and sister, Steve and Jeannine, both of whom I have looked up to since childhood.

Most of all I wish to acknowledge my parents, Patricia and Ronald. Countless times during my work on this thesis I have been thankful for their guidance and love.

This thesis was partially funded by DARPA project number MDA 972-01-1-0038.

CHAPTER 1:

INTRODUCTION

A typical internet search engine has a database of documents, or web pages, from the internet. On June 20, 2002, the Google search engine indexed approximately 2,073,418,204 such web pages. On March 20, 2005, Google indexed approximately 8,058,044,651 web pages. Given such a large and growing collection, it is impossible for a person to manually find web pages that are relevant to a particular interest. Therefore, search engines were designed to find quickly a relatively small number of web pages that are highly relevant to a user's interest.

A typical search engine home page includes a text box where a user can type keywords, or terms, that the user believes are most relevant to their interest. Words, phrases, characters, and numbers supplied individually or collectively by the user form the keywords. The user then submits those keywords to the search engine. The keywords, as the user submits them, compose a query. There are other possible components of a query, including Boolean operators, keyword proximity constraints, etc., but keywords form the basis of any query. When a query is submitted to a search engine, the search engine applies an algorithm for finding web pages that are somehow measured to be most relevant to the user's query. Based on some minimum relevance criteria,

websites are selected for inclusion in a search results list. Websites in the search results list are ranked in order of decreasing level of relevance and the list is then presented to the user. The search results list typically includes the title of each web page in the list along with a brief description of the web page and the URL of the web page.

1.1. PROBLEM DESCRIPTION

Often times, a search engine user will not know what to look for or how to translate their information needs into a search engine query [89]. Even so, a user will do the best he can to compose an initial query. Based on website titles and descriptions in a user's initial search results list, the user might conclude that there are no websites in the list that are relevant to the user's interest. Consequently, the user will not be satisfied with their initial search results list, even though the search engine may be highly capable of returning relevant search results when given the appropriate query. The user will then conclude either that the query was not a good query, that the search engine is poorly designed, or that there are no web pages on the internet that are relevant to the user's interest. If the user believes the problem is with the query, then he will need to submit a different query to the search engine in order to retrieve a different search results list. Multiple queries are often required in order to find adequately relevant websites, with each query analogous to a fisherman casting a line into the ocean with few clues as to what lies in the ocean's depths. This is a source of inefficiencies in the search process.

Recall and precision are two popular measures of search engine performance. From the user's point of view, the most important measure is the one that better addresses

the user's background and information needs. For instance, users that know exactly what they are searching for will believe they know the most relevant and effective keywords to include in their queries. The user will then submit those queries with the intention of retrieving precisely the web pages that are highly relevant without retrieving irrelevant web pages.

This thesis posits that there is a potential for information exchange that is currently unexploited by today's search engines. *In particular, not only do search engines possess information about web pages on the internet, but they also have information about queries that have been submitted to the search engine and about the search results obtained by those queries*.

A problem addressed in this thesis is the formulation of mathematical constructs that calculate distances between queries and that enable improved search engine dialogue using the concepts that users attempt to communicate with search engine queries and from the search results of those queries.

The hypothesis tested is that mathematical modeling can be used to express relationships between user behavior and search engine performance.

1.2. APPROACH

In this thesis, the approach to the problem is to treat the search process as a dialogue scenario involving two dialogue agents, namely the user and the search engine. In this type of scenario, the beliefs of each agent are affected by the dialogue actions of the other agent. Consequently, an agent will adjust its actions according to the actions of

the other agent as the dialogue progresses. Currently, the user is the only agent that adjusts its beliefs over the course of the search process. The search engine, on the other hand, responds only to the current user query without any regard for historical user behavior. In this thesis, concepts from topology are employed to formulate a mathematical distance measure that is applied to the search results of historical user queries. The distance measure is designed for analyzing and comparing user queries and, with an eye towards increasing the dialogue nature of internet searches, to act as a mediator between the search engine and search engine users.

To test and verify the measure, a survey that collected queries related to two different search tasks was administered to a sample of internet users. Data collected with the surveys included queries, subject background information, and query descriptions. The distance measure was applied to the search results of those queries after submitting the queries to a search engine. Statistical analysis was performed on the survey data and the calculated query distances to look for relationships between the survey data and the calculated distances.

1.3. UNIQUENESS AND CONTRIBUTIONS

The uniqueness of approach in this thesis lies in the development of a mathematical model in the context of describing the search process as a dialogue between the user and the search engine. In describing the problem this way, search results lists become a product of the interaction between the user and the search engine which can be compared to the interaction products between other users and the search engine. As

described in Chapter 7, it is believed that the distance measure has potential as a basis for future query tools and as an analysis tool for understanding search engine users and for search engine comparison and development.

1.4. THESIS ORGANIZATION

In Chapter 2, literature is surveyed to summarize research that has been done in fields related to this thesis. In Chapter 3, the methodology is described, including describing the problem in terms of Belief, Desire, and Intention Theory and speech acts. In Chapter 4, the mathematics of the proposed distance measure are described. In Chapter 5, the survey for collecting user backgrounds and queries is described. Statistical analysis of the survey results is in Chapter 6. Conclusions and comments on future work are in Chapter 7. Appendix A contains the survey described in Chapter 5. Appendix B is a description of a simulation that was built to model users performing a search with a search engine. The model is difficult to validate and no simulation analysis is performed in the thesis. Appendix C describes the main tasks performed during the statistical analysis of the survey results.

1.5. NOTE

This work is original in content. We tried to be faithful to the literature and included all the references that have been used in the current work. Whenever we have used a source, we have referenced the original work. Any omission is only accidental and not intentional. No research work is ever complete, so is this one. In this work we

tried exploring a new path for searching the internet; search through analyzing distance between user queries.

CHAPTER 2:

LITERATURE REVIEW

In this chapter, literature is reviewed to summarize topics that are relevant to this thesis and to information retrieval. The chapter is organized into four sections, which are Knowledge and Information Need, Search Engines for Information Retrieval, Presentation of Retrieval Results, User Search Behavior, and Query Assistance.

2.1. KNOWLEDGE AND INFORMATION NEED

[98] defined an information need as being visceral, conscious, formalized, or compromised. A visceral information need is the actual information need of a person, although it may be unrecognized by the person as the actual information need. A conscious information need is that which the user tries to formalize in some fashion, e.g. in a sentence. A compromised information need is the information need formed in an expression that can be represented in an information system, e.g. a query.

As described in [78], there is “explicit” user modeling, which constructs models “explicitly by the user”, and “implicit” user modeling, which constructs models that are “abstracted by the system on the basis of the user’s behavior”. The problem with explicit modeling systems, and even some implicit modeling systems, is that the user is required

to provide explicit feedback, e.g. a numerical rating of viewed documents. This is a problem, because it increases the cognitive load on the user and because single-valued numerical ratings are often inadequate descriptions of a document. Implicit feedback about a user can include such evidence as whether the user read or ignored a document, whether the user saved or deleted a document, and whether or not the user replied to a message [94]. Other implicit evidence has been found in reading duration [72].

A model for anticipating contextually-motivated inferences addressees are likely to draw from information that is presented to them was developed by [49]. Their model is in the context of consecutively uttered propositions, e.g. “It rained yesterday” followed by “Mary took an umbrella to work”, and they cite empirical evidence ([60], [37], and [100]) suggesting that humans draw causal inferences during reading to close gaps left implicit in narrative texts as support and motivation for their approach. Inference reasoning is presumably done by building forward-oriented expectations and by drawing backward-driven inferences.

Regression was used in [47] in a home video recommendation service. In that study, users supplied numerical evaluations of movies they had seen and regression was performed on the evaluations of users with correlated interests to recommend unseen movies. One peculiar aspect of that study was that the movie database and user interests were relatively stable.

2.2. SEARCH ENGINES FOR INFORMATION RETRIEVAL

This thesis is concerned with information retrieval as opposed to information filtering. As described in [12], information filtering (IF) systems are designed for large streams of unstructured data that must be filtered based on individual or group preferences, called profiles, which typically represent long-term interests. After a stream of data is filtered, the user of the system is presented with the unfiltered data, not the extracted data. For example, some email systems filter out spam from a user's email box. Information retrieval (IR) systems, on the other hand, differ from IF systems in the following ways:

a. The user typically has a one-time information goal that the user tries to achieve during a single information seeking episode.
b. IR systems recognize the inadequacies of queries as representations of information needs.
c. IR systems typically are concerned with the collection and organization of texts.
d. IR systems are concerned with the selection of texts from a relatively stable database.

Internet search engines are described in the following two sub-sections, which are Indexing and Retrieval.

2.2.1. INDEXING

Internet search engines today automatically compile their web page indexes through the use of computer programs called "spiders", although the web page publisher

can also manually submit web pages to the search engines. A spider traverses the web by following links from web site to web site. The search engine then indexes web sites that the spider finds along the way. Each search engine has its own criteria for deciding which web sites to index. For example, some search engines are programmed for depth, so they index not only main sites, but also subsidiary pages to main sites. Search engines that are programmed for breadth are concerned with indexing more main web pages, but are not as concerned with indexing subsidiary web pages.

Different search engines index different parts, or fields, of web pages. Some search engines claim to index the entire text from every web page, while others only index keywords from fields such as a web page's title, URL, metatags, etc. Most search engines index the "high value" fields that appear at the top of a web page, such as the title and the URL.

Often times, "stop words" are not indexed at all. Stop words are common words that appear frequently in web pages and they are considered insignificant by the search engine because they contribute very little information about the content of a web page. For example, words like "it", "the", "of", etc., are often listed as stop words by search engines. Some search engines also do not index words less than three characters long [53].

Inverted File Structures (IFS) are common in information retrieval and database systems. An IFS tracks which documents contain which index terms, or keywords, by organizing information into an abbreviated list of terms, which then reference a specific set of documents. In a web search engine, for example, a word that is indexed may be

indexed along with the URLs in which it appears. The word may also be indexed along with the number of times the word appears in each web page, the position(s) the word holds in each web page, and the fields where the word appears in each web page. The position of a word is defined according to the number of words, including stop words, which appear before the word in the web page. For example, in the sentence "The buck stops here", the word "buck" holds position 2 [13].

Given a document collection and its corresponding inverted file structure (IFS), vector space models can be used to represent both terms and documents in the text collection. In vector space models, a document collection composed of n documents indexed by m terms can be represented by an $m \times n$ *term-by-document matrix* A. The n columns of A represent the n documents in the collection and are interpreted as the *document vectors*. The m rows of A are interpreted as the *term vectors*. The matrix element, a_{ij}, is the weighted frequency with which term i occurs in document j. With large document collections such as the WWW, the number of terms is much smaller than the number of documents, i.e. $m << n$. Consequently, a typical document vector will contain mostly 0's, because the document will use only a small portion of the English language (assuming English documents). In order to avoid the storage and processing of zero elements, sparse matrix storage formats have been developed that require only $2nnz + m + 1$ storage (array) locations compared with mn for the complete matrix A, where nnz is the number of nonzero values in A. Sparse matrix storage formats include compressed row storage (CRS) and compressed column storage (CCS).

As the semantic content of each document in a collection is represented by the *relative* frequencies of terms, the elements in A are sometimes scaled so that the Euclidian norm of each column is 1. The Euclidian norm of column j, $a_{.j,}$ is then defined by Equation 1, i.e. Eq. 1.

$$\left\| a_j \right\|_2 = 1 = \sqrt{\sum_{i=1}^{m} a_{ij}^2} \qquad \text{Eq. 1}$$

Each matrix element, $a_{ij,}$ is defined to be the *weighted* frequency with which term i occurs in document j and the purpose of weighted frequencies is to improve retrieval performance. Retrieval performance is measured by a search engine's ability to retrieve relevant information (recall) and its ability to dismiss irrelevant information (precision).

There are a number of ways to compute each element a_{ij}. Let each element a_{ij} be defined by Eq. 2,

$$a_{ij} = l_{ij} g_i d_j, \qquad \text{Eq. 2}$$

where l_{ij} is the local weight for term i occurring in document j, g_i is the global weight for term i in the collection, and d_j is a document normalization factor. Table 1, Table 2, and Table 3 contain some popular weight formulas used in automated indexing systems. For convenience, let

$$\chi(r) = \begin{cases} 1 & \text{if } r > 0, \\ 0 & \text{if } r = 0 \end{cases},$$

define f_{ij} as the number of times that term i appears in document j, and let

$p_{ij} = f_{ij} \Big/ \sum_j f_{ij}$.

Symbol	Name	Formula
B	Binary	$\chi(f_{ij})$
l	Logarithmic	$\log(1 + f_{ij})$
N	Augmented normalized Term frequency	$\left(\chi(f_{ij}) + \left(f_{ij} / \max_k f_{kj}\right)\right)/2$
T	Term frequency	f_{ij}

Table 1. Local term weight (lij) formulas in automated indexing systems. Source: [13].

Symbol	Name	Formula
X	None	1
E	Entropy	$1 + \left(\sum_j \left(p_{ij} \log(p_{ij})\right) \Big/ \log n \right)$
F	Inverse document frequency (IDF)	$\log\left(n \Big/ \sum_j \chi\left(f_{ij}\right) \right)$
G	Gfldf	$\sum_j f_{ij} \Big/ \sum_j \chi\left(f_{ij}\right)$
N	Normal	$1 \Big/ \sqrt{\sum_j f_{ij}^2}$
P	Probabilistic Inverse	$\log\left(\left(n - \sum_j \chi(f_{ij}) \right) \Big/ \sum_j \chi(f_{ij}) \right)$

Table 2. Global term weight (gi) formulas in automated indexing systems. Source: [13].

Symbol	Name	Formula
x	None	1
c	Cosine	$\left(\sum_j (g_i l_{ij})^2 \right)^{-1/2}$

Table 3. Document normalization (dj) formulas in automated indexing systems. Source: [13].

2.2.2. RETRIEVAL

Given a user query, a search engine retrieves documents that are calculated to be relevant to the user's interest. Judgment of a document's relevance is based on keywords, although the mere presence of matching keywords is not enough for a web page to earn a high ranking. In addition to keywords, a web page's ranking can be based on link popularity, click popularity, stickiness, and themes [53].

The contribution of keywords to a web page's ranking is a function of keyword prominence, proximity, density, and frequency. Keyword prominence refers to keyword location in the web page. The two most prominent places to locate keywords are the URL and the title, which are located at the top of a web page. Relevant keywords, i.e. query terms, located at the top of a web page will significantly boost a web page's ranking. Keyword proximity refers to how close keywords are to each other in a web page. The closer two keywords are to each other, the higher the ranking for the web page. Keyword density and frequency are related to each other. The more frequently a keyword appears in a web page and, consequently, the higher the keyword density for the

web page compared to a web page of the same length, the higher the ranking for the web page [53].

Given a user query, a search engine must perform some query matching technique to find documents in the database that are most similar to the query. With vector space models, this can be viewed as searching the columns of the term-by-document matrix A for the most similar documents. A common similarity measure for query matching is the *cosine* of the angle between the query vector and the document vectors. If the *j*th document vector is defined as a_j and the query vector $q = (q_1, q_2,..., q_m)^T$, then the cosine between a_j and q is defined by Eq. 3.

$$\cos\theta_j = \frac{a_j^r q}{\|a_j\|_2 \|q\|_2} = \frac{\sum_{i=1}^{m} a_{ij} q_i}{\sqrt{\sum_{i=1}^{m} a_{ij}^2}\sqrt{\sum_{i=1}^{m} q_i^2}} \qquad \text{Eq. 3}$$

Documents that produce cosines with the query vector that are greater than some threshold value, e.g. ≥ 0.5, are judged to be relevant to the user query. This technique, however, does not always adequately model the similarity in semantic content between queries and documents. Term weights are an approach to improve the results. Another approach is based on low-rank approximations to the original term-by-document matrix A. Improved retrieval results using low-rank approximations are achieved through *noise* reduction due to problems like synonymy and polysemy [53].

A web page's link popularity refers to links to the web page from other web pages on the internet. It is important to note that link popularity is not just about the number of links to a web page. Although more links to a web page is better, the relevance of those

links is also significant. For example, if Joe's web page is about his furniture manufacturing business, links from other furniture companies will contribute a great deal more to his web page's ranking than will a link from his mother's cinema review web site. Another factor in link popularity is the text that is used by other web sites to describe the link. For example, consider these two links to the same World Cup of Soccer web site.

FIFA World Cup

Click here

Keywords in the text of a link that are used to describe a web site are considered to be relevant to that web site, thus increasing that web site's ranking with regard to those keywords. The publisher of the example World Cup of Soccer web site would probably much prefer the first link over the second one, even though the publisher has little control over the link text that is used by other publishers [53].

Click popularity is a measure of how often users select a web page for viewing whenever the web page is listed among the documents in the search results list. The idea is that a web page that is often selected from among all other retrieved documents is probably a good document that is highly relevant to that particular query, or topic. One argument against click popularity is that low scoring documents will never achieve high click popularity, because they appear low on the search results list. However, programmers claim to counter that problem by placing higher weight on the first clicks that a web page receives.

When users spend a lot of time on a web page, the web page is said to have high stickiness, although the amount of time spent by a user on a web page must be monitored. Web pages with high stickiness are assumed to be good web pages. Consequently, some search engines will give higher rankings, if possible, to web pages with high stickiness.

Themes are another criteria used by some search engines for ranking web pages. The presence of a theme in a web page is signaled to a search engine by the presence of the same keywords in the title, metatags, text, links, and the sites that links lead to. The presence of a theme that is relevant to the user's query will boost the web page's ranking [53].

2.3. PRESENTATION OF RETRIEVAL RESULTS

Conventional search engines commonly present their search results in the form of a list that is sorted in decreasing order of relevance to the user's query [26]. Alternatives to conventional search results lists include graphical displays of interdocument similarity [22] [35] [99] relationship to fixed attributes[61] [93], query term distribution patterns [45], and clustering [28] [46], e.g. the Northern Lights search engine [39]. Clustering applied to entire corpora has also been investigated [28] [29] [58] [62] [75] [103] [105] and [107]. Methods for category specific web searching [38] [39] [40] and [50] organize or narrow search results to specific categories, e.g. personal homepages, product reviews, research papers, etc. Clustering, query modification, and relevance feedback have been employed in these methods.

2.4. USER SEARCH BEHAVIOR

After studying a large corpus of web search queries extracted from recorded server logs, [63] constructed Bayesian networks for predicting user search behavior over time. The networks were modeled in terms of probabilistic relationships among temporal patterns of activity, i.e. the length of time between query submissions by a particular user, informational goals, and classes of query refinement. The authors defined seven intervals of time, ranging from the interval of 0 to 10 seconds to the interval of greater than 20 minutes. They also defined 14 categories of informational goals, such as *Entertainment* and *Career Opportunities*, and seven query refinement classes, such as *Reformulation* and *Specialization*.

Among the key statistics derived form the server log data were that users made an average of 4.28 queries over the course of a day, they averaged 1.31 informational goals per day, 3.27 queries per goal, and the average length of queries was 2.30 words. Distinct relationships were found between the time interval separating adjacent queries and the refinement class of the successive query. For example, the probability that a user will specialize a previous query rises to a maximum in the 20-30 second interval. Such relationships may have implications for anticipating user interactions with search engines.

A study was performed by [92] on an AltaVista query log that contained approximately 1 billion entries for search requests, composed of approximately 150 million unique queries, and over 285 million user sessions over a six-week period. It was found that of all non-empty requests, i.e. requests that contained at least one query term, 32% consisted of a request for a new result screen and 68% consisted of requests for the first result screen of a new query. 63% of all sessions consisted of only one query and one result screen examined. The average number of queries per session was 2.02, with 77.6% of all sessions having only one query, and the average number of screens per session was 1.39. The 25 most common queries in the query log formed 1.5% of the total number of queries asked over the six week period covered by the query log, despite being only 0.00000016% of the unique queries. The average number of times a query appeared in the query log was 3.97, with a minimum of 1, a maximum of 1,551,477, and a standard deviation of 221.31. 63.7% of all unique queries were submitted only once over the six week period. Another study of a different query log [57] found the average number of

queries per session to be 2.8 and the average number of screens per session to be 2.21. [92] found that in situations where a query is modified, about 12% of the queries were modified by adding or deleting terms or operators. In about 35% of the cases the queries were totally changed and in about 53% of the cases query terms were replaced.

The relatively low number of queries per session found in [57] and [92] would seem to indicate the possibility that users' information needs require the submission of only one or two queries to a search engine. However, user surveys from other sources seem to indicate that user information needs are frequently not satisfied. For example, a British survey found that 72% of British adult internet users were frustrated by the quality of information from searching, while 31% of all respondents stated that they "often" do not find what they need on the internet [52] [54]. A different survey [51] [52] found that 20% give up when they are unable to find what they are looking for on a specific search site, but most simply try another search site. Based on these statistics, there is a good possibility that two significant contributors to the low average number of queries per session found by [57] and [92] are that users sometimes change search engines and they sometimes give up on their search out of frustration.

Studies addressing user strategies and usability of closed hypermedia systems, databases, and library information systems [20] [27] distinguish between browsing and searching by categorizing two to three browsing strategies:

1. Search browsing: a directed search where the goal is known
2. General purpose browsing: consulting sources that have a high likelihood of items of interest

3. Serendipitous browsing: purely random

Marchionini [67] designates open and closed tasks. Closed tasks have a specific answer, while open tasks are more subject-oriented and less specific. Bates [11] challenges the classic model of information retrieval, in which user queries are matched with representations of relevant documents in a document database. Bates' argues that searches are more often like berrypicking. When a person picks huckleberries or blueberries in a forest, the berries are scattered on the bushes, rather than growing in bunches. According to [11], searching for information is done in a similar fashion, with the user picking up bits of information at different points in their search. As a user progresses through their search, each new piece of information gives the user new ideas and directions to follow and, consequently, the user's conception of their query changes. As the user's conception of their query changes, the query itself is continually shifting, e.g. query terms are changed. Not only can the user's conception of their query change, but the user's information need can change, too. Bates calls this type of search an *evolving search*. At the end of an evolving search, the information need is not necessarily satisfied by a single final retrieved set of documents. Instead, bits of information are gathered from different locations along the path of the user's search. Bates' [11] review of research by [66], [48], [95], [96] and [33] attests to the popularity of this approach in a variety of environments. However, these studies were conducted in closed systems, while the world wide web (WWW) is an open and dynamic system. The first study of user browsing strategies on the web was performed by [21], which supplemented knowledge of user navigation strategies. Since [21], there have been

additional studies of user patterns in web-based information seeking, e.g. use of web-browser history lists to re-visit web pages [97] and of web browser interfaces, e.g. [24]. Various browsing advisors, including agent-based advisors have been offered, e.g. [56] and [65].

2.5. QUERY ASSISTANCE

There are a few approaches to automatically assisting the user with formulating queries, including query expansion, query refinement, phrase browsing, intelligent agents, relevance feedback, and query distance/similarity, each of which is discussed below.

Query expansion is an automatic process that modifies a query based on the relevance of documents in a preliminary retrieval system output, e.g. search results list. Query expansion systems have used the Rocchio algorithm [81], the technique of Local Context Analysis [108], probabilistic models [80], the cluster hypothesis [1] and [101], word co-occurrence in documents [84], information theory [19], and Hidden Markov Models [69], among other techniques.

Query refinement is the incremental process of transforming a query into a different query through iterative term changes in previous queries. A *query refinement facility* automatically recommends query terms to the user for the user to add to, subtract from, or otherwise modify a previous query. For example, a domain-specific thesaurus may be used for automatically suggesting terms to users, such is in [18], which uses a semantic distance function for calculating distances between words for linguistic support in an information broker. The Hyperindex Browser (HiB) works in conjunction with a

search engine and produces reformulations of a query in the form of linguistically well-formed phrases [16]. With the HiB, user queries are passed to the associated search engine and the search results are then analyzed by the HiB. Using a shallow natural language parsing technique, phrases are derived from the titles of web pages in the search results list and a hyperindex of refinement possibilities is computed and presented to the user. A hyperindex is a union of lattices, in which a lattice is a collection of expressions. Each expression is composed of terms connected with *operators*, e.g. prepositions such as *of*, *by*, *in*, etc. The refinement possibilities are more specific than the user's current query. *Query by navigation* is the process of navigation through the hyperindex, such as HiB. At any point, the user can choose a desired phrase that is then submitted to the associated search engine. The search engine's results for the chosen query are then presented to the user.

The Paraphrase Search Assistant [2] is similar to HiB in that it supports query reformulation through interactive use of linguistic phrases that are derived from documents in a search results list. [70] take the approach of using WordNet to expand internet queries.

A study in [17] included performing experiments with 54 human subjects to compare the effectiveness of

1. standard internet query search as supported by the Google search engine
2. directory browsing as supported by Yahoo
3. phrase-based query reformulation as supported by HiB.

Comparisons between the three internet search mechanisms listed above were based on three measures. The three measures were

1. independent relevance ratings of documents perused by subjects during their search
2. the length of time before the user first bookmarked a relevant page during their search
3. the demands placed on the user while interacting with the search mechanisms. Demands were measured by the amount of time users spent in each phase their search, e.g. reading through document summaries, and by the cognitive load experienced by the users. Cognitive load was measured using a dual task methodology.

Experimental results indicated that using the HiB versus the standard internet query search can improve the relevance of documents perused by users, but at the cost of increased search time and cognitive load.

Phrase browsing [9] [31] [74] [106] is a technique for exploring the document database for words that occur in the context of query terms and phrases. Words that are found in the context of query terms and phrases can then be shown to the user as suggestions for additional or alternative query terms.

Various intelligent agents [3] [10] [23] [36] [43] [58] [65] [68] [73] have been developed whose purpose is to search the web and retrieve documents that are relevant to the user's interest. These agents are often, although not always, geared towards users with a persistent interest in a particular domain and they use a number of tools such as

term weighting schemes, neural networks, competitive learning, and fuzzy logic. The agents themselves are autonomous and/or interface agents. Autonomous agents operate in parallel with the user. Interface agents are capable of affecting objects in a direct manipulation graphical interface without explicit instruction from the user.

Relevance feedback is a process that automatically reformulates a user's query based on relevance judgments about documents that are initially retrieved. Relevance feedback from the user about query terms is has also been proposed [102]. Reformulations of user queries include changing terms and/or changing the weights of query terms. Relevance judgments may be automatic [87] or fixed directly by the user [44]. Salton's [83] contribution was a system that would successively modify and improve a user's query formulation based on user feedback about search results. However, the presumption was that the information need remains unchanged, regardless of new information that is obtained from retrieved documents by the user.

With regards to query distance/similarity, [77] proposes two similarity measures between queries based on the retrieval output of queries. The similarity measures proposed by [77] were developed in the context of so-called *optimal queries*. An *optimal query* is defined to be a linear classifier, e.g. [79], [85], and [86], that uses relevance feedback from the user to separate relevant documents from irrelevant documents. Although the similarity measures proposed by [77] may not necessarily be restricted to optimal queries, they are defined on queries that retrieve the same set of documents and is based on the rankings of those documents. If the set of retrieved documents is not defined to be the entire corpus, then the similarity measures are useless for queries that do

not retrieve the same documents. If the set of retrieved documents is defined to be the entire corpus, then the retrieval output of a query would be the ranking of all documents in the corpus with respect to that query. Since users will typically only view the first one or two pages of a search results list, the similar measures may not reflect what the user actually sees. Another weakness of the measures is that they were developed in the context of a static information need. The assumption of relevance feedback is also a drawback.

Another query similarity measure is offered by [34]. However, their similarity measure has numerous weaknesses. First of all, their measure is not symmetric and, therefore, is not a metric. Second, for two queries that do not share documents in their search results lists, the similarity equation does not differentiate between them with respect to a third reference query. Third, it assumes the same number of documents, i.e. the top 200, in each of the two results lists used to compute the similarity between queries. Fourth, it essentially is a query refinement tool, i.e. it adds terms to the user's query, which amounts to a search within a search results list. This assumes a static information need.

CHAPTER 3:

RESEARCH METHODOLOGY

3.1. INTRODUCTION

When a user initiates an internet search session, the user's motivation is to find information, which implies an information retrieval scenario. However, a search session is often lengthier and more complicated than a single query followed by a single search results list. A search session often involves multiple iterations of information retrieval events. Consequently, internet searches with search engines are analogous to a game of "20 Questions", in which search engines only offer information involuntarily in response to queries, never voluntarily. From a user perspective information is pulled and never pushed. In contrast, in human-human interactions each person learns about the other person as their dialogue progresses and each participant often volunteers information based on their own past beliefs and understanding of the other participant. For example, a helpful librarian will often query an inquiring patron about specific topics and subtopics that may be unknown to the patron in order to identify the patron's particular area of interest and to retrieve the appropriate material. By querying the patron, the librarian is voluntarily offering knowledge to the user to make the patron-librarian interaction more productive. The patron-librarian interaction is therefore more accurately described as a

dialogue session of information exchange rather than a sequence of information retrieval events.

Information exchange between people involves not only the information itself, but also the speech acts humans employ to aid in exchanging information. For example, a simple "Yes" is a very informative response to the question "Are you interested in Abraham Lincoln speeches only from the year 1864?", even though the content of the response, i.e. the word "Yes", is minimal. Of course, the information in the response is dependent on the question, which is part of the nature of dialogue, and the meaning of each statement must be interpreted in the context of the dialogue. Although speech acts play an important role in human communication, information is inherent in the purpose of communication and it must be possible for information to be represented linguistically or graphically before it can be exchanged. If information can be iteratively exchanged in two directions between two agents in a dialogue, e.g. between a helpful librarian and an inquiring patron, then searching for relevant information can be more efficient than when it is limited to information retrieval in one direction, e.g. a game of "20 Questions".

To increase the dialogue nature of interactions between two agents, it is necessary to increase the understanding that at least one actor has of the other. For example, a librarian is more helpful if the librarian has more knowledge of subjects that are of interest to library patrons and if the librarian is familiar with the type of questions that patrons ask and the intentions behind those questions. In the user/search engine scenario, a search engine could be more helpful if the search engine had more knowledge about the intentions behind user queries.

For a search engine to have more knowledge of user intentions, it would be necessary for search engine designers to code such a capability into search engine software. This brings up the issue of what to automate and what to leave entirely to the people, or users. Anything that is coded into the search engine is automated and is an attempt to add "understanding" of users to a search engine. Of course, the coding of any capability into a search engine must be based on search engine designer knowledge and understanding of that capability. All automated and non-automated search tasks, when taken together, comprise something that is analogous to an information search system. The purpose of that system is to fulfill user information needs.

If a task is not automated, then it is up to the users to better understand how to perform the task with or without the aid of the search engine. For example, a person using a search engine might be unhappy with the search results obtained with the first few queries of a search. Because search engines only return search results that are relevant to a single query and search engines are not capable of deducing a user's intentions from a single query, it is entirely up to the user to improve the search results and the success of the search by adjusting his understanding of the search engine and his expectations of future search results. The performance of the system, of which the users are a part, then depends on the performance of current system users when they are performing non-automated tasks. When users have only themselves on which to rely when interfacing with the search engine, they cannot benefit from the experience of others. If people want to share knowledge about non-automated tasks, then it is up to the

people to communicate knowledge and understanding about that task, including experience, between each other.

Without a search engine, the task of finding websites and sifting through them to find relevant information would be so enormous as to be practically impossible. Among the benefits of search engines is that they have an algorithm for automatically finding websites, indexing them, and ranking them in response to user queries. However, there is much of the search process that is not automated by search engines, although the search engine/user interface dictates the common steps involved in a search. The search process involves a number of steps that form an information-finding heuristic comprised of both automated and non-automated tasks. The steps are listed below in the left hand column of Table 4 and the actors who perform the steps are listed in the right hand column. Table 4 is not meant as a description of a search heuristic, but as a listing of components, or steps, in a system that extracts information from the internet and funnels that information to the user partly via interactions between the user and the search engine. Any step performed by the search engine is automated. Some user steps are mechanical in nature, e.g. Step 5. User steps of a cognitive nature are tasks that typically are not aided by automated tools. For example, Step 4, formulating a query, is a cognitive exercise approached in a manner of the user's own choosing.

Search Step	Actor
1. Define the information need	1. User
2. Go to a search engine website	2. User
3. Present the user/search engine interface	3. Search Engine
4. Formulate a query	4. User
5. Type the query into the query text box	5. User
6. Submit the query	6. User
7. Receive the query	7. Search Engine
8. Rank documents relative to query	8. Search Engine
9. Present search results	9. Search Engine
10. Read search results	10. User
11. From search results list, visit websites that seem relevant	11. User
12. Read contents of visited websites	12. User
13. From within visited websites, follow links that seem relevant	13. User
14. Decide whether information need is satisfied	14. User
15. If necessary, repeat the process	15. User

Table 4. Steps in the information search process.

The goal of automating a search task is to make the overall system more efficient to the benefit of both the user and the search engine. The effects of automating a task may include, but is not limited to, the following:

1. Consistency and predictability: everybody does the same thing.
2. Once a good automation is found, the automated task is performed for everyone, including those who would normally be incapable.
3. Established automations do not allow new and better ways of doing things.
4. Massive tasks can be greatly simplified for individuals.
5. The efficiency of the population can be improved by providing an automatic tool that facilitates either coordination or knowledge sharing among the population.
6. The efficiency of the population can be decreased if a far-from-optimal program is automated.

Given the number of tasks that currently are performed by the user, there is room for further automation in the information retrieval system with the potential benefit being more efficient searching. However, choosing tasks to automate should be based on which tasks are better automated and which tasks are better left to the users.

3.1.1. PRECISION VS. RECALL

It is assumed here that the usefulness of a query, as well as a search engine, is determined ultimately by the search results that it obtains. The assumption is based on, arguably, the most visible performance characteristic of any search engine, which is the degree of relevance to a query of the documents that a search engine retrieves and ranks highly in response to a query. In relation to this performance characteristic, the two most common performance measures in information retrieval are precision and recall. It is precision and recall that search engine designers are trying to improve when perfecting their search engines. Precision is the ability to retrieve documents that are relevant to a user's interest at the maximum exclusion of irrelevant documents. Recall is the ability to retrieve the maximum number of documents that are relevant to a user's interest. Typically, there is a tradeoff between precision and recall with any search engine [8]. As an example, a generic precision-recall graph is illustrated in Figure 1.

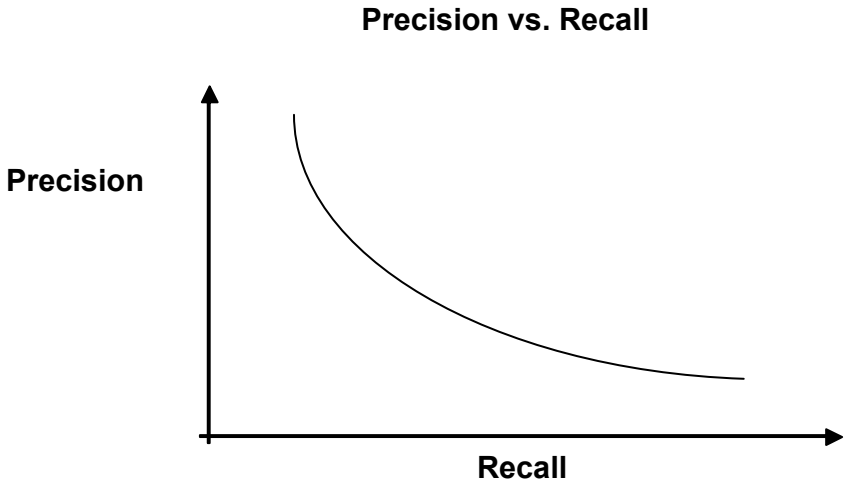


Figure 1. The tradeoff between precision and recall.

For example, search engines often give document ratings between 0 and 100 based on the relevance of a document to a particular query, with 0 being least relevant and 100 being most relevant. To achieve higher precision, a search engine will sometimes increase the minimum rating required for documents to be included in a search results list. To achieve higher recall, the minimum rating will be lowered, e.g. a minimum rating of 70 will result in lower precision than a minimum rating of 80. A lower minimum rating will result in lower precision, because a lower minimum rating decreases the average document rating in a search results list. The benefit of higher recall is that more documents are retrieved, thus increasing the probability of retrieving the documents that best satisfy the user's information requirements. On the other hand, the benefit of higher precision is that fewer irrelevant documents are retrieved, thus reducing the amount of information the user needs to sift through manually to find the best documents within the search results list.

3.1.2. TYPES OF SEARCHES

Precision and recall are relevant to the type of search being performed by the user. In general, there are two types of information searches. One type of search is analogous to finding a needle in a haystack. In that type of search, the user knows precisely the information he or she is searching for, e.g. a particular website, but the size of the database is enormously large. In such cases, precision is more desirable than recall, especially since the user likely knows the specific terms to include in the query for best retrieval performance. The second type of search is a comprehensive search in which the user doesn't know precisely the information they're searching for, e.g. the user's goal is to educate himself on a particular subject, such as the causes of the American Civil War. For a comprehensive search, the user typically requires more recall than precision from the search engine, partly because the user might expect multiple websites to be necessary for accomplishing the user's goal.

No matter the type of search being performed, when a series of queries are required for a search on a particular subject, the desired degree of precision and recall will fluctuate over the sequence of query submissions. Therefore, the degree of precision and recall desired for the overall search session may be different from the degree of precision and recall desired for a particular query that is submitted during that search session.

3.1.3. BELIEFS, DESIRES, & INTENTIONS

There are dialogue characteristics inherent in human-search engine (HSE) interactions and those characteristics present an opportunity for improving the search process through the application of intelligent agents, although such agents are not developed in this research. Belief, Desire, and Intention Theory (BDI Theory) [15] is often employed in the development of intelligent agents and the terms "belief", "desire", and "intention" are useful here, because they aid in describing the phenomena exhibited in HSE interactions.

The beliefs of an agent are those propositions that represent the state of the world to the agent. For example, an agent might have the belief that all birds fly. The desires of an agent represent those states that an agent finds positive and would like to achieve. For example, a thirsty person might have the desire to drink water. The intentions of an agent arise from rational deliberation and are those actions that an agent plans to follow. For example, a golfer might have the intention of hitting a golf ball down the middle of a fairway. The terms "belief", "desire", and "intention" are used in the following description of the search process, which leads to the five steps in the proposed methodology.

On average, a typical user submits more than two queries to a search engine during a search on a particular theme [92] [57]. The total number of queries during a search can be as many as 10 or more with query modifications often serving the purpose of gaining either more precision, more recall, or of shifting the focus of the search. The sequence of queries submitted by a user during a search session usually will exhibit a

user adapting query formulations according to new information the user obtains during a search. Such behavior is often referred to as a search tactic [104]. A search tactic will be exhibited by such characteristics as adding terms, deleting terms, substituting terms, etc. Some steps in that sequence will be more drastic than others, e.g. the number of common terms between two consecutive queries may be relatively few or many. For example, a person doing a comprehensive search on the relationship between indoor carpets and allergies might submit a sequence of queries like those in Table 5.

Query Number	Query
1	carpet allergies
2	indoor carpet allergens
3	dust mites
4	carpet animal dander
5	hepa filter

Table 5. An example sequence of queries on the relationship between indoor carpets and allergies.

All else being equal, it might be concluded that fewer common terms between consecutive queries indicates a more drastic shift in the user's search intentions, i.e. the distance between the queries is large. This is in contrast to a shift in the user's information desires. For example, a user that submits the query "Battle of Gettysburg" and then submits the query "Emancipation Proclamation" might be intending to switch

their focus from one aspect of the American Civil War to another, although it might remain their desire to do a comprehensive search on the American Civil War.

If it is true that the usefulness of a query is determined ultimately by the search results that it obtains, then the degree of change from query to query typically will be some type of a reflection on the user's beliefs about previous search results. For instance, the degree of change might be directly proportional to user beliefs about the relevance of the previous search results. On the other hand, a significant change in term usage might indicate a shift in the user's intentions. The exact terms that a user includes in their next query will depend additionally on the user's beliefs about term usage in the language, term usage in the database, and search engine performance.

3.1.4. DIALOGUE

An important and basic claim, especially for purposes of human-computer interaction, is that communication presupposes asymmetries of knowledge and participation of various kinds. For this particular problem, the dialogue of interest is that which takes place between the user and the search engine at a search engine website. On one hand, the user has knowledge about the theme of the search. On the other hand, the search engine has knowledge about documents in the database, i.e. internet websites. The user communicates knowledge via queries and the search engine communicates knowledge via search results. Although both the search engine and the user are communicating, their communications are not eliminating asymmetries of knowledge and participation if the dialogue leaves the user unsatisfied, i.e. the user quits the search.

Human-machine oral dialogue is one potential solution to eliminating asymmetries of knowledge, e.g. see [14], with efforts that are based on one of several dialogue models, such as dialogue grammars, e.g. see [76], plan-based models, e.g. see [3], and joint action models, e.g. see [24]. In this thesis, however, human-machine oral dialogue is not attempted. Instead, the attempt is to develop a tool that will promote interactions between search engines and users that exhibit more dialogue characteristics.

With respect to user knowledge about how a search tool works, common knowledge [6] [64] [88] between a user and a search engine about the basis for a search tool is a potential factor that influences the way a user interacts with the search tool.

For example, consider a query suggestion tool that is based on the number of shared documents between the search results of queries. Such a basis is simple, intuitive, and easy for the user to understand, so controlling common knowledge about the query suggestion tool would be possible in human experiments. If the user is informed of the basis, then common knowledge is established, because the search engine and the user would both know, the search engine and the user would both know that the other knows, and the search engine and the user would both know that the other knows that they know about the basis for the query suggestion tool. If a particular mathematical construct doesn't capture any semantic meaning between queries, it still might capture some meaning in the way humans interact with search tools that feature common knowledge. For example, if a person chooses not to submit a suggested query, it's possible that the user purposely avoids the suggested query because, due to common knowledge, the user knows the type of search results the query will retrieve.

In human dialogue, there are three speech acts performed whenever something is spoken [7] [91] . The first act is a locutionary act. A locutionary act is the act of uttering a sequence of words, e.g. "That apple pie looks delicious". The second act is an illocutionary act. An illocutionary act is the act that the speaker intends to perform when saying words. For example, the remark "That apple pie looks delicious" may be intended as only a compliment to the person that baked the apple pie. If the remark achieves its intended purpose, the baker might reply with something like "Thank you". However, a speaker's locutionary act might not necessarily result in the intended illocutionary act. The speech act that actually results from a locutionary act is the third type of speech act, which is a perlocutionary act. For example, instead of interpreting "That apple pie looks delicious" as just a compliment, the baker of the pie might interpret the remark as a request for a slice of the apple pie and reply with "Thank you. You may have a slice."

The importance of precision and recall manifests itself in the dialogue nature of HSE interactions. The goal of search engine designers is to retrieve documents that are the most relevant to a user's query, but not to overwhelm the user with irrelevant documents. This goal is captured in Grice's Maxims, which are listed in Table 6. Grice's Maxims were introduced as a set of guiding principles that underlie all communication [41]. Of particular interest are the Maxims of Quantity and Relation, which bear a strong resemblance to the definitions of precision and recall. Search engines achieve the Maxim of Quality, because search results are based largely on term frequencies in the database itself, i.e. internet websites. The Maxim of Manner is important, because users must avoid submitting queries that may have two or more

possible interpretations. For example, the phrase "Rochester hotels" has at least two interpretations, one being "hotels in Rochester, New York" and the other being "hotels in Rochester, Minnesota". The most useful interpretation to the user will depend on the city of interest to the user.

Maxim	Description
1. Maxim of Quantity	Make your contribution as informative as required, but not overly informative.
2. Maxim of Quality	Do not say things for which you lack evidence.
3. Maxim of Relation	What you say should be relevant to the current topic.
4. Maxim of Manner	Avoid obscurity of expression and ambiguity.

Table 6. Grice's Maxims.

When a user submits a query to a search engine, the content and form of the query is chosen by the user based on the user's beliefs about search results that will be given in response to the query. However, what the user believes and what the search engine's response will actually be are not necessarily the same thing. Therefore, the degree of change between consecutive search results might not mirror exactly the degree of change between the respective queries. Making such comparisons between query differences and search results differences requires some measure that can be applied equally to both search results and queries. Given that there is not yet a direct measure of the difference between search results, nor a direct measure of the difference between queries, it would be more appropriate to develop these measures before developing a unified measure that

can be applied to both search results and queries. A direct measure of the difference between queries would most likely employ linguistic tools involving syntax and semantics. Such a measure is not the interest here. Instead, the focus here is on a direct measure between search results and to indirectly apply that measure to queries.

3.2. POSTULATES

In this research, it is claimed that acts performed by users and search engines are analogous to speech acts. A query is a locutionary act. The illocutionary act behind a query is a request, although there may be additional illocutionary acts behind the query. For example, a user that is dissatisfied with the results of their first query might submit a second query, or request, with the hope of receiving a more relevant response. In this case, the second query is not only a request, but it is also a retraction of the first query. Since a search engine is capable of performing only one type of speech act, as described next, a user is capable of producing only one type of perlocutionary act, which is a request. When search engines become capable of distinguishing between user illocutionary acts, users will then be capable of producing more than one perlocutionary act.

Following the submission of a user query, the search engine will perform a locutionary act in the form of a search results list. The illocutionary act behind the search results list is a response. Because a search engine is a search tool that is designed to be submissive to the user, a response is the only possible illocutionary act from a search engine. Therefore, even if search engines were capable of perceiving the consequences

of their acts, any resulting perlocutionary acts would neither be expected nor unexpected by a search engine. That is an important reason for the one-dimensional nature of HSE interactions.

3.3. METHODOLOGY STEPS

The five main methodology steps are listed here.

1. Formulate a Query Distance Measure.
2. Survey User Queries and Intentions.
3. Statistically Analyze User Queries and Intentions.
4. Apply Distance Measure to Search Results.
5. Compare Statistical Analysis with Query Distances.

The following five sections describe the methodology steps in more detail.

3.3.1. STEP 1: FORMULATE A QUERY DISTANCE MEASURE

The distance measure is based on search results lists. Search results lists are the primary mode of communication employed by search engines and, therefore, act as a form of language between users and search engines. By comparing search results lists, the usage of terms in queries is taken into account, because a search results list is a response to a query. Comparing search results lists also takes into account term usage in the database, because a search engine includes websites in a search results list based on the usage of the query terms throughout the database.

Applying set theory to information retrieval, [32] and [82] studied the topology of retrieval systems and their precision and recall. To compare queries, set theory is similarly applied here to the search results obtained by those queries. At first, a simple but intuitive idea was employed for comparing the search results of queries, which resulted in an initial equation. From that initial equation, a more useful equation was developed. The inputs into both formulations are the number of shared and unshared websites between the search results lists of the queries being compared. A calculated distance between two search results lists is the output of the expressions. That distance is used to represent the distance between the queries that retrieved the search results. If the mathematical distance captures a relationship between search results lists and their respective queries, the captured relationship may or may not be a semantic one and it is not hypothesized that any semantic relationships are being captured. Instead, the captured relationship, if there is one, will be one that is based at least on term usage in the database, term usage in queries, and the performance of the particular search engine.

Mathematical formulations are discussed in Chapter 4.

3.3.2. STEP 2: SURVEY USER QUERIES AND INTENTIONS

It's difficult to judge accurately a user's meaning and intentions, e.g. whether the user is trying to achieve more or less precision with a particular query, when they submit a query to a search engine, thus making search engine query logs insufficient for that purpose. Therefore, a survey was composed containing two different search tasks and subjects were asked to compose queries for each search task. By providing subjects with

a search task, the subjects were given an information need, or desire. To gain some knowledge about each subject's beliefs, each subject taking the survey also was asked to provide relevant background information, queries for the search tasks, and some information about their queries. To gain some knowledge about subjects' intentions, each subject was asked to write sentence descriptions of each of their queries and to rate each query with regards to the recall and precision the subject hoped to achieve with each query. Gathering this information was an attempt to better understand how subjects compose queries and to look for relationships between the gathered information and the proposed distance measure.

The survey is discussed in Chapter 5.

3.3.3. STEP 3: STATISTICALLY ANALYZE USER QUERIES AND INTENTIONS

The goal of this methodology step was to summarize subject data and query data and to find any relationships between subjects' beliefs, e.g. background, and intentions, e.g. recall and precision goals. To do so, after collecting query surveys and entering queries from the surveys into an Excel spreadsheet, a number of statistics on the queries were extracted. The statistics were based on the number of keywords in the queries, the number of keywords in the search task descriptions, and answers to questions about each subject's background. To look for relationships, statistical correlations were calculated for different pairs of data, e.g. background vs. number of keywords in a query.

The statistical analysis of user queries and intentions is discussed in detail in Chapter 6.

3.3.4. STEP 4: APPLY DISTANCE MEASURE TO SEARCH RESULTS

The goal of this methodology step was to find relationships between the proposed distance measure and subjects' beliefs, desires, and intentions. To do so, after obtaining user queries and intentions with a survey, user queries were submitted to a search engine to obtain search results for each query. The search results were saved and the web pages in those search results were used to calculate query distances with the distance measure from Step 1. For each query, all other queries were clustered based on calculated distances and statistical comparisons were made between query clusters and subjects' beliefs, e.g. background, and desires, e.g. search task.

Distance comparisons and clustering are discussed in more detail in Chapter 6.

CHAPTER 4:

MATHEMATICAL FORMULATIONS

4.1. DISTANCE METHOD I

This section describes mathematical terminology and an example of one possible method for measuring distances between queries.

4.1.1. TERMINOLOGY

The distance method will be based on calculating the distance between search results. Comparisons between search results will be made by looking at the number of common documents and the total number of documents in the search results lists.

A topic is defined in this thesis as a set of one or more documents. Therefore, a document can then be referred to as either a document or a topic, although a document is simply referred to as a document in this thesis. A search results list can be referred to interchangeably as either a topic or a set of topics. Theoretical distances are based on measures applied to topics.

Notation used is as follows:

I is the set of all documents available

S is a search engine

Q is the set of all possible queries

$|A|$ denotes the cardinality of A, where A is any set

q_i is a particular query

T is the set of all topics = $\mathcal{P}(I)$ (power set of I, $|\mathcal{P}(I)|=2^{|I|}$)

T_i is a particular topic

t_i^k is the *k*th document in topic *i*

$D(T_i, T_j)$ is the distance between topics T_i and T_j.

$d(q_i, q_j) = D(T_i, T_j) = D(S(q_i), S(q_j))$ is the distance between queries q_i and q_j.

The search engine S maps the set of queries Q to the set of topics T, i.e. $S(Q) \rightarrow T$; S is not usually surjective or injective. A search results list returned by S is a set of documents. Therefore, a search results list is a topic. For query q_i, S maps q_i to T_i, where T_i is the search results list returned by S in response to the query q_i being submitted to S, i.e. $S(q_i) \rightarrow T_i$. T_i is referred to as query q_i's topic. The cardinality of T_i, written as $|T_i|$, is equal to the number of documents returned by S in response to a user's submission of query q_i to S. Typically, a search results list is sorted from 1 to $|T_i|$ in decreasing level of rating, where the method of rating is dependent on the search engine. If t_i^k is the k^{th} rated document in the topic T_i, then T_i is equal to

$$\bigcup_{k=1}^{|T_i|} t_i^k .$$

The distance between topics T_i and T_j, $d(T_i, T_j)$, will be expressed in terms of the intersection and union of topics. The intersection of topics T_i and T_j, written as $T_i \cap T_j$,

is the set of documents that T_i and T_j have in common. Since $T_i \cap T_j$ is a set of documents, $T_i \cap T_j$ is a topic. The union of topics T_i and T_j, written as $T_i \cup T_j$, is the set of documents contained in T_i and/or T_j. Since $T_i \cup T_j$ is a set of documents, it, too, is a topic.

4.1.2. DISTANCE MEASURE I

One possible use of a distance measure is to suggest alternative queries to the user based on the search results of queries. For example, let's say a comparison between queries q_A and q_B is to be made by calculating a distance between them. Using search results and the definitions above, one possible way to calculate $D(T_A, T_B)$ is with Eq. 4.

$$D(T_A, T_B) = \frac{|T_A \cup T_B|}{|T_A \cap T_B|} \qquad \text{Eq. 4}$$

Eq. 4 is a theoretical measure of the distance between topics. After calculating $D(T_A, T_B)$, it must be decided whether or not $D(T_A, T_B)$ is significant, e.g. whether the distance between T_A and T_B is close enough for q_A and q_B to be considered similar to each other. If $D(T_A, T_B)$ is significant, then for purposes of aiding the user with his overall search, the user might be referred q_B if the user submits query q_A to the search engine, and vice versa. The form of the reference chosen could be the topic T_A itself. However, this would be similar to increasing the size of the search results list presented to the user in response to q_A by simply adding T_B to T_A. However, since search results lists often contain thousands of documents, adding T_B to T_A would most likely cause T_B to be practically invisible to the user due to the resultant large search results list.

Alternatively, if the user has a working knowledge of the language in which the documents in the database are written, then it can assumed that the user can form a hypothesis about the content of a query's topic if the user is given the query itself. Hence, instead of employing a brute force approach of adding a topic to a search results list, an alternative approach of suggesting alternative queries to the user in conjunction with the search results of the user's query would need to be tested. This would offer a more visible and manageable search tool that takes advantage of the user's ability to form a hypothesis about a query's topic if given the query itself.

Eq. 4 has the advantage of being intuitive as well as symmetric. The following example illustrates its application.

4.1.3. EXAMPLE

Numerous environmental conditions are capable of inducing allergic reactions in people. For those who are particularly susceptible to allergic reactions, it is necessary to control their environment as much as possible to prevent allergic reactions in themselves. For example, a household carpet is a potential source of allergens such as dust mites and animal dander. Therefore, it is important that a person who is susceptible to allergies be informed about the association between household carpets and allergens.

For this particular example, 13 different queries related to carpets and allergies were submitted to the Google search engine and their topics, or search results, were compared. The topics were obtained in one afternoon. The 13 queries are listed below.

1. Allergens Carpet
2. Allergies Carpet Cleaning
3. Carpet Allergies
4. Carpet Allergy
5. Carpet Allergy Prevention
6. Dander Carpet
7. Dust Mite
8. Dust Mite Allergies
9. Dust Mite Allergy
10. Dust Mite Carpet
11. HEPA Allergy
12. HEPA Filter
13. HEPA Vacuum

The comparison of topics involved a pair wise manual review of topics and the counting of the number of websites shared between topics. Since the number of websites in any given topic numbered in the thousands, only the top 100 ranked documents were reviewed for each comparison. Therefore, it is assumed that $\left| T_i \right| = 100$ for each of the 13 queries. Although this assumption gives inaccurate results, the example is still useful for illustrative purposes.

A total of 202 websites were found to appear in two or more of the 13 T_i's. As already mentioned, these shared websites were among the top 100 ranked websites in the

search results. Table 7 contains the number of shared websites between the 13 T_i's. The row and column headings are of the form "Qi", which refers to query i, e.g. "Q11" refers to the query "HEPA Allergy". Each number in Table 7 represents a number of shared websites, e.g. the 13 in the Q4 row and Q3 column means that 13 websites are shared between T_3 and T_4. From the data in Table 7, calculated distances between query topics are given in Table 8. When two topics do not share any documents, then the theoretical distance between them is infinite.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13
Q1													
Q2	6												
Q3	10	20											
Q4	11	7	13										
Q5	2	0	12	14									
Q6	13	5	5	7	3								
Q7	0	0	1	1	0	0							
Q8	0	0	1	2	2	0	41						
Q9	0	0	0	2	2	0	47	40					
Q10	3	1	2	3	4	2	32	37	23				
Q11	0	0	0	6	2	0	0	0	0	0			
Q12	0	0	0	0	0	0	0	0	0	0	4		
Q13	0	0	0	1	0	0	0	0	0	0	16	5	

Table 7. The number of shared websites between pairs of query topics.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13
Q1													
Q2	32.3												
Q3	19	9											
Q4	17.2	27.6	14.4										
Q5	99	∞	15.7	13.3									
Q6	14.4	39	39	27.6	65.7								
Q7	∞	∞	199	199	∞	∞							
Q8	∞	∞	199	99	99	∞	3.9						
Q9	∞	∞	∞	99	99	∞	3.3	4					
Q10	65.7	199	99	65.7	49	99	5.3	4.41	7.7				
Q11	∞	∞	∞	32.3	99	∞	∞	∞	∞	∞			
Q12	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	49		
Q13	∞	∞	∞	199	∞	∞	∞	∞	∞	∞	11.5	39	

Table 8. Distances between query topics.

There are some problems with Eq. 4. One problem is that it does not satisfy reflexivity, because the calculated distance between any topic and itself is equal to one, not zero. Another problem with Eq. 4 is that it calculates the same distance for any two topics that do not intersect. Specifically, the calculated distance between any two non-intersecting topics is always equal to infinite. Finally, Eq. 4 does not satisfy the triangle inequality. For example, consider the following case for queries q_A, q_B, and q_C.

$| T_A | = 100$ $\quad | T_B | = 100$ $\quad | T_C | = 60$

$\left| T_A \cup T_B \right| = 150$ $\quad \left| T_A \cup T_C \right| = 150$ $\quad \left| T_B \cup T_C \right| = 110$

$\left| T_A \cap T_B \right| = 50$ $\quad \left| T_A \cap T_C \right| = 10$ $\quad \left| T_B \cap T_C \right| = 50$

$\Rightarrow d(q_A, q_B) = 150/50 = 3; \quad d(q_B, q_C) = 110/50 = 11/5; \quad d(q_A, q_C) = 150/10 = 15.$

$\Rightarrow d(q_A, q_B) + d(q_B, q_C) = 3 + 11/5 = 26/5$

$\Rightarrow d(q_A, q_B) + d(q_B, q_C) \leq d(q_A, q_C).$

To effectively measure the distance between queries, need a more useful distance equation is needed. Other candidates include Jaccard's [55] index and the Marczewski-Steinhaus metric [82], although both are incapable of differentiating between two non-intersecting topics of the same cardinality.

4.2. DISTANCE METHOD II

This section describes the chosen method in this thesis for measuring mathematically the distances between queries.

4.2.1. INTERSECTING TOPICS

A distance measure should be consistent with the notion that the distance between similar objects is less than that between dissimilar objects. The documents that appear in search results lists offer such a measure of the similarity between topics. When comparing the topics of two queries, assume that each unshared document increases the distance between the two topics and thus increases the distance between the two queries. Likewise, assume that each shared document between two topics decreases the distance between the two topics and thus decreases the distance between the two queries. This relationship between queries with intersecting topics is captured in Eq. 5.

$$d(q_A, q_B) = D(T_A, T_B) = \left| T_A \cup T_B \right| - \left| T_A \cap T_B \right| \qquad \text{Eq. 5}$$

4.2.2. Eq. 5 AND THE TRIANGLE INEQUALITY

Eq. 5 is symmetric and it satisfies reflexivity. It also satisfies the triangle inequality. To prove that the triangle inequality is satisfied, consider the topics T_A, T_B, and T_C, in which T_A intersects T_B, T_B intersects T_C, and T_A intersects T_C. Eq. 6 needs to be shown true.

$$d(q_A, q_B) + d(q_B, q_C) \geq d(q_A, q_C) \qquad \text{Eq. 6}$$

To begin, let's rewrite Eq. 6 in the same form as the right side of Eq. 5. The result is Eq. 7.

$$|T_A \cup T_B| - |T_A \cap T_B| + |T_B \cup T_C| - |T_B \cap T_C| \geq |T_A \cup T_C| - |T_A \cap T_C| \qquad \text{Eq. 7}$$

Next, refer to Figure 2.

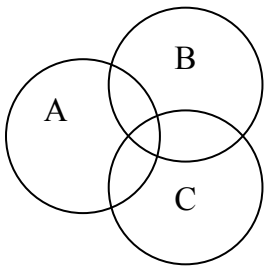


Figure 2. A diagram representing the topics A, B, and C.

In Figure 2, topics A, B, and C are represented as a group of intersecting circles. The portions of circles that overlap represent documents that are shared between topics. Although the diagram shows a set of documents that are shared among all three topics, this need not be the case.

As illustrated in Figure 2, a document can appear in only one of eight possible regions. For example, an unshared document in T_A will appear in the region $T_A \cap \overline{T}_B \cap \overline{T}_C$. The eight possible regions in which a document may appear are listed below.

$$T_A \cap \overline{T}_B \cap \overline{T}_C \qquad \overline{T}_A \cap \overline{T}_B \cap \overline{T}_C$$

$$T_A \cap T_B \cap \overline{T}_C \qquad \overline{T}_A \cap T_B \cap \overline{T}_C$$

$$T_A \cap \overline{T}_B \cap T_C \qquad \overline{T}_A \cap \overline{T}_B \cap T_C$$

$$T_A \cap T_B \cap T_C \qquad \overline{T}_A \cap T_B \cap T_C$$

Eq. 5 can now be written in terms of the eight regions listed above. The result is Eq. 8.

$$d(q_A, q_B) = \left|T_A \cap \overline{T}_B \cap \overline{T}_C\right| + \left|T_A \cap \overline{T}_B \cap T_C\right| + \left|\overline{T}_A \cap T_B \cap \overline{T}_C\right| + \left|\overline{T}_A \cap T_B \cap T_C\right| \quad \text{Eq. 8}$$

Rewriting Eq. 7 with Eq. 8, the left side of Eq. 7 becomes Eq. 9.

$$\left|T_A \cap \overline{T}_B \cap \overline{T}_C\right| + \left|T_A \cap \overline{T}_B \cap T_C\right| + \left|\overline{T}_A \cap T_B \cap \overline{T}_C\right| + \left|\overline{T}_A \cap T_B \cap T_C\right|$$

$$+ \left|T_A \cap T_B \cap \overline{T}_C\right| + \left|T_A \cap \overline{T}_B \cap T_C\right| + \left|\overline{T}_A \cap T_B \cap \overline{T}_C\right| + \left|\overline{T}_A \cap \overline{T}_B \cap T_C\right| \quad \text{Eq. 9}$$

The right side of Eq. 7 becomes Eq. 10.

$$\left|T_A \cap \overline{T}_B \cap \overline{T}_C\right| + \left|T_A \cap T_B \cap \overline{T}_C\right| + \left|\overline{T}_A \cap \overline{T}_B \cap T_C\right| + \left|\overline{T}_A \cap T_B \cap T_C\right| \quad \text{Eq. 10}$$

By inspection, it can be seen that Eq. 9 is always greater than or equal to Eq. 10. Therefore, Eq. 5 satisfies the triangle inequality.

4.2.3. NON-INTERSECTING TOPICS

Recall that Eq. 4 always calculates an infinite distance for queries that do not share documents. Eq. 5, on the other hand, calculates a distance equal to $|T_A \cup T_B|$ whenever $|T_A \cap T_B| = 0$. This is not very useful, because it will result in a relatively short calculated distance between any two queries with relatively small topics, regardless of how irrelevant the queries are to each other. Therefore, Eq. 5 needs to be modified to improve its distance calculation performance for queries with nonintersecting topics.

4.2.4. TOPIC CHAINS

Assume that T_A and T_B are two nonintersecting topics. Although T_A and T_B do not intersect, there may exist other queries, stored in a query database, with topics that do intersect with either T_A or T_B. Because of this, there may exist a "chain" of one or more topics, or "links", between T_A and T_B that connects T_A with T_B via a series of intersections between links in that chain. In addition, there may exist multiple chains that connect T_A with T_B, although the restriction will be placed on chains that any given topic may appear no more than once in a given chain. As an example, see Figure 3 below. Chains will be considered to be unique only if they differ by at least one topic, or link, regardless of the sequence of topics in the chain. Although each link between T_A and T_B must intersect with at least two other links, there is no upper limit on the number of links with which a particular link may intersect.

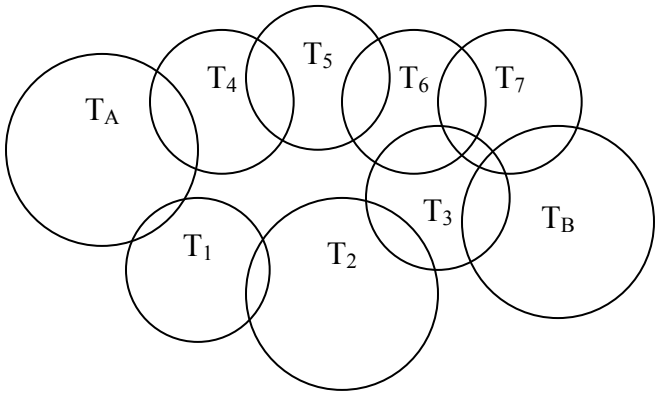


Figure 3. A graphical example of a set of topics. Each circle represents a topic and intersecting circles represent intersecting topics

In Figure 3, there are six unique chains that connect T_A with T_B. The six chains are listed in Table 9. In Table 9, note that the topics T_A and T_B are included in each of the chains. This practice will be followed when defining chains and when counting the number of links in a chain.

Chain Number	Chain Links
1	T_A-T_1-T_2-T_3-T_B
2	T_A-T_1-T_2-T_3-T_7-T_B
3	T_A-T_1-T_2-T_3-T_6-T_7-T_B
4	T_A-T_4-T_5-T_6-T_7-T_B
5	T_A-T_4-T_5-T_6-T_7-T_3-T_B
6	T_A-T_4-T_5-T_6-T_3-T_B

Table 9. The six unique chains in Figure 3 that connect T_A with T_B.

As another example, let's say there exists a chain between T_A and T_B that contains only topic T_{10}. Because T_{10} connects T_A with T_B, it must be true that $T_{10} \cap T_A \neq \{\}$ and

$T_{10} \cap T_B \neq \{\}$, where $\{\}$ is the empty set. It is certainly possible that $T_{10} \cap T_A \neq T_A$ and $T_{10} \cap T_B \neq T_B$, as in diagram (i) of Figure 4. It is also possible that $T_{10} \cap T_A = T_A$, $T_{10} \cap T_B = T_B$, or $T_{10} \cap (T_A \cup T_B) = T_A \cup T_B$, as in diagrams (ii), (iii), and (iv), respectively in Figure 4. Right now, the case when $T_A \cap T_B = \{\}$ is considered; however, a chain is not restricted to connecting only those topics that do not intersect. In the T_A, T_B, and T_{10} example, if $T_A \cap T_B \neq \{\}$, then it becomes possible that $T_{10} \cap T_A = T_{10}$ or $T_{10} \cap T_B = T_{10}$.

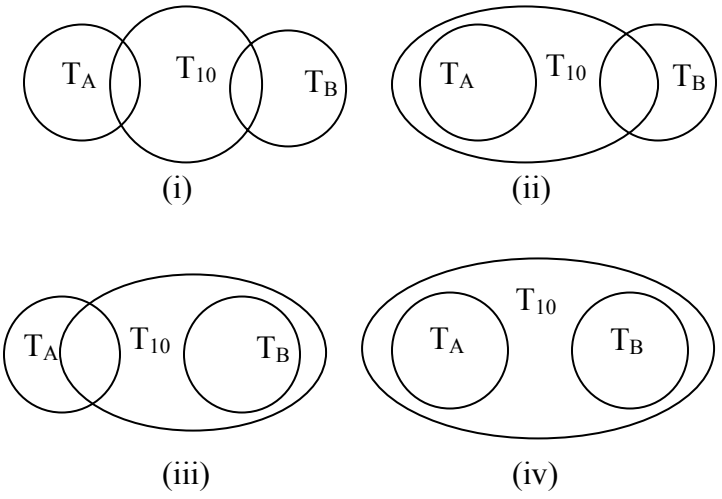


Figure 4. Possible scenarios when T_A and T_B do not intersect.

Given a chain between the topics T_A and T_B, the links in that chain will be labeled from 1 to L, beginning with T_A, in the sequence that they appear in the chain. T_B will then be labeled Link L. The chain can then be listed as $\{T_1, T_2,...,T_L\}$. If so chosen, links could be labeled beginning with T_B as Link 1 and ending with T_A as Link L. Either way,

the topic represented by the chain $\{T_1, T_2,...,T_L\}$ is equal to the union of the topics in the chain. The cardinality of this topic is written as $| T_1 \cup T_2 \cup \ldots \cup T_L |$.

4.2.5. CHAIN LINKS

There are two general classes of topics within any chain of topics, the exterior topics and the interior topics. The exterior topics are the topics T_1 and T_L. The interior topics are the topics T_2 through T_{L-1}. Since a chain is composed of intersecting topics, any document in the chain can appear in anywhere from 1 to L topics. If a document appears in only 1 topic then it is an unshared document in that chain. If it appears in 2 or more topics, then it is a shared document in that chain. The 11 possible locations of a document in a chain are listed below.

1. 1 end topic
2. 2 end topics
3. 1 interior topic
4. 2 or more, but not all, interior topics
5. All interior topics
6. 1 end topic and 1 interior topic
7. 1 end topic and 2 or more, but not all, interior topics
8. 1 end topic and all interior topics
9. 2 end topics and 1 interior topic
10. 2 end topics and 2 or more, but not all, interior topics
11. 2 end topics and all interior topics

4.2.6. CHAIN LENGTH

The concept of chain "length" is now introduced. The length of a chain between T_A and T_B will be a mathematical concept and will be denoted as $X\{T_A,\ldots,T_B\}$. If T_A is labeled as Link 1 and T_B is Link L, then the length of the chain between T_A and T_B will be equal to Eq. 11.

$$X\{T_A,\ldots,T_B\} = \left| \bigcup_{k=1}^{L} T_k \right| - \left| T_1 \cap T_L \right| \qquad \text{Eq. 11}$$

Note that if $\left| T_A \cap T_B \right| > 0$ and L = 2, then Eq. 11 simplifies to Eq. 5.

As mentioned earlier, it is possible that there exists more than one chain linking T_A with T_B. The number of chains linking T_A with T_B will be denoted as nc_{AB}. Chain *i* between T_A and T_B will be denoted as $\{T_A,\ldots,T_B\}^i$. The length of chain *i* between T_A and T_B will be donoted as $X\{T_A,\ldots,T_B\}^i$, which is defined by Eq. 11. The distance between queries A and B will now be defined as the length of the shortest chain between T_A and T_B, which is written mathematically as Eq. 12.

$$d(q_A, q_B) = D(T_A, T_B) = \min_i \{X\{T_A,\ldots,T_B\}^i\} \qquad \text{Eq. 12}$$

Eq. 12 is symmetric, satisfies reflexivity, and calculates distances for queries that have either intersecting or nonintersecting topics. Eq. 12 also satisfies the triangle inequality, the proof of which follows.

4.2.7. Eq. 12 AND THE TRIANGLE INEQUALITY

Proving that Eq. 12 satisfies the triangle inequality is begun by showing that Eq. 11 satisfies the triangle inequality for any particular chain between topics T_1 and T_L.

Assum there exists three topics T_A, T_B, and T_C in the chain $\{T_A,...,T_B,...,T_C\}$. After labeling T_A as T_1, T_B as T_R, and T_C as T_L, with $1 < R < L$, Eq. 11 gives the length of this chain as

$$X\{T_1,\ldots,T_L\} = \left|\bigcup_{k=1}^{L} T_k\right| - \left|T_1 \cap T_L\right|.$$

Let's now split the chain into two pieces, $\{T_1,...,T_R\}$ and $\{T_R,...,T_L\}$. According to Eq. 11, the length of $\{T_1,...,T_R\}$ is equal to

$$X\{T_1,\ldots,T_R\} = \left|\bigcup_{k=1}^{R} T_k\right| - \left|T_1 \cap T_R\right|$$

and the length of $\{T_R,...,T_L\}$ is equal to

$$X\{T_R,\ldots,T_L\} = \left|\bigcup_{k=R}^{L} T_k\right| - \left|T_R \cap T_L\right|.$$

To show that Eq. 11 satisfies the Triangle Inequality for a particular chain, Eq. 13 needs to be shown true.

$$X\{T_1,\ldots,T_R\} + X\{T_R,\ldots,T_L\} \geq X\{T_1,\ldots,T_L\} \qquad \text{Eq. 13}$$

According Eq. 11, the right hand side of Eq. 13 is equal to

$$\left|\bigcup_{k=1}^{L} T_k\right| - \left|T_1 \cap T_L\right|. \qquad \text{Eq. 14}$$

The left hand side of Eq. 13 is equal to

$$\left|\bigcup_{k=1}^{R} T_k\right| - \left|T_1 \cap T_R\right| + \left|\bigcup_{k=R}^{L} T_k\right| - \left|T_R \cap T_L\right|. \qquad \text{Eq. 15}$$

In Eq. 15 there appears the quantity $\left|\bigcup_{k=1}^{R} T_k\right| + \left|\bigcup_{k=R}^{L} T_k\right|$.

Since T_R is included in each of $\bigcup_{k=1}^{R} T_k$ and $\bigcup_{k=R}^{L} T_k$, Eq. 16 must be true.

$$\left|\bigcup_{k=1}^{R} T_k\right| + \left|\bigcup_{k=R}^{L} T_k\right| \geq \left|\bigcup_{k=1}^{L} T_k\right| + \left|T_R\right| \qquad \text{Eq. 16}$$

Therefore, Eq. 15 is greater than or equal to

$$\left|\bigcup_{k=1}^{L} T_k\right| + \left|T_R\right| - \left|T_1 \cap T_R\right| - \left|T_R \cap T_L\right|$$

$$= \left|\bigcup_{k=1}^{L} T_k\right| + \left|\overline{T}_1 \cap T_R\right| - \left|T_R \cap T_L\right|$$

$$= \left|\bigcup_{k=1}^{L} T_k\right| + \left|\overline{T}_1 \cap T_R \cap T_L\right| + \left|\overline{T}_1 \cap T_R \cap \overline{T}_L\right| - \left|T_1 \cap T_R \cap T_L\right| - \left|\overline{T}_1 \cap T_R \cap T_L\right|$$

$$= \left|\bigcup_{k=1}^{L} T_k\right| + \left|\overline{T}_1 \cap T_R \cap \overline{T}_L\right| - \left|T_1 \cap T_R \cap T_L\right|. \qquad \text{Eq. 17}$$

Since $T_1 \cap T_R \cap T_L$ is a subset of $T_1 \cap T_L$, Eq. 17 is clearly greater than or equal to the right hand side of Eq. 13. Therefore, Eq. 11 satisfies the triangle inequality for any particular chain between topics T_1 and T_L.

The next step is to show that Eq. 12 satisfies the triangle inequality. To begin, assume two queries, q_A and q_B and apply Eq. 11 and Eq. 12 to find the shortest chain between T_A and T_B. The shortest chain will be donoted as $\{T_A,\ldots,T_B\}^*$. Since Eq. 11

satisfies the triangle inequality for any particular chain, Eq. 18 is true for any R in $\{T_A,\ldots,T_B\}^*$ such that $\{T_A,\ldots,T_R\} \cup \{T_R,\ldots,T_B\} = \{T_A,\ldots,T_B\}^*$.

$$X\{T_A,\ldots,T_R\} + X\{T_R,\ldots,T_B\} \geq X\{T_A,\ldots,T_B\}^* \qquad \text{Eq. 18}$$

In addition, by Eq. 12, $X\{T_A,\ldots,T_B\}^* \leq X\{T_A,\ldots,T_B\}^i$ for all i. Once again applying the property that Eq. 11 satisfies the triangle inequality for any particular chain, Eq. 19 is true for any T_R such that $\{T_A,\ldots,T_R\} \cup \{T_R,\ldots,T_B\} = \{T_A,\ldots,T_B\}^i$.

$$X\{T_A,\ldots,T_B\}^* \leq X\{T_A,\ldots,T_B\}^i \leq X\{T_A,\ldots,T_R\} + X\{T_R,\ldots,T_B\} \qquad \text{Eq. 19}$$

Therefore, Eq. 12 satisfies the triangle inequality.

4.2.8. Eq. 12 AND INTERSECTING TOPICS

Another property of Eq. 12 is that, given $\left|T_A \cap T_B\right| > 0$, the shortest possible chain connecting T_A with T_B will be the chain $\{T_A,\ldots,T_B\}$. This is true, because if $L \geq 2$, then

$$\left|\bigcup_{k=1}^{L} T_k\right| \geq \left|T_A \cup T_B\right|.$$

Therefore, if $L \geq 2$, then

$$\left|\bigcup_{k=1}^{L} T_k\right| - \left|T_A \cap T_B\right| \geq \left|T_A \cup T_B\right| - \left|T_A \cap T_B\right|.$$

4.2.9. Eq. 12 ALTERNATIVES

Although Eq. 11 and Eq. 12 may be mathematically satisfactory, the task of finding the minimum chain between topics may be so difficult computationally as to

make implementation impractical. Therefore, alternative and more practical distance equations are desired that will approximate the theoretical distances supplied by Eq. 11 and Eq. 12. An alternative to Eq. 11 is Eq. 20.

$$X\{T_A,\ldots,T_B\} = \sum_{i=1}^{L-1} d(q_i, q_{i+1}) \qquad \text{Eq. 20}$$

In Eq. 20, $d(q_i, q_{i+1})$ is equal to Eq. 5 for intersecting topics. A more general form of Eq. 20 is Eq. 21,

$$X\{T_A,\ldots,T_B\} = \left[\sum_{i=1}^{L-1} d(q_i, q_{i+1})^r\right]^{1/r}, \qquad \text{Eq. 21}$$

where the parameter r could possibly be chosen by the user. This use of the Minkowski functional for chain length is essentially Schvaneveldt's Pathfinder semantic distance [90], with user interregations replaced by the intersecting topic distance formula for individual link weights. Increasing r would allow the user to better differentiate between queries of similar distances, which might be useful to a user that prefers to make more gradual changes in their queries. The distance between two queries can then defined according to Eq. 12, which is the length of the shortest chain between the two queries' associated topics. Shortest path algorithms such as Dijkstra [42] could be applied to find the shortest chains between topics and query distances could be easily updated when new queries are added to the query database.

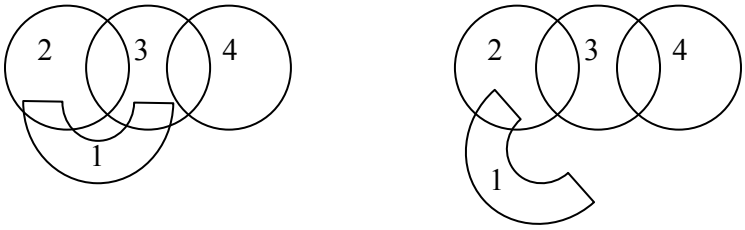
An equivalent approach to Eq. 20 is to treat each topic as an object with a set of characteristics. Each characteristic of an object would be the presence or absence of a database document, e.g. website. Each website would be represented by a binary

variable, with the variable equal to 1 if the website is retrieved by a query and equal to 0 if the website is not retrieved by a query. The total number of possible characteristics would then be equal to the total number of documents in the database. With this approach, an equation that is equivalent to Eq. 20 would be the Euclidean squared distance between two topics, *i* and *k*, which is equal to

$$\sum_{j}^{p}(x_{ij} - x_{kj})^2 ,$$

where *p* is the total number of documents in the database and x_{ij} is equal to 1 if document *j* is retrieved by query *i* and equal to 0 otherwise; likewise for x_{kj}.

Although Eq. 20 and Eq. 21 share with Eq. 11 the same properties, i.e. reflexivity, symmetry, and the triangle inequality [18], they are approximations of Eq. 11. Therefore, there may be a tradeoff between practicality and accuracy and/or precision. For instance, refer to the 1-2-3-4 chains in the two examples in Figure 5. Eq. 11 will count each document in the chain only once, which will allow Eq. 11 to distinguish between the two examples. However, if r = 1, then Eq. 21 will count all documents in interior topics twice, in addition to counting all documents in exterior topics once. This will not allow Eq. 21 to distinguish between the two examples. This may or may not be of practical significance.



Example 1 Example 2

Figure 5. Two examples of 1-2-3-4 chains. The only difference between the chains is that topic 1 intersects with topic 3.

4.2.10. EXAMPLE

Returning now to the carpet and allergy example, Eq. 4, Eq. 12, and Eq. 21 are compared. For this example, Table 7 is reproduced below as Table 10.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13
Q1													
Q2	6												
Q3	10	20											
Q4	11	7	13										
Q5	2	0	12	14									
Q6	13	5	5	7	3								
Q7	0	0	1	1	0	0							
Q8	0	0	1	2	2	0	41						
Q9	0	0	0	2	2	0	47	40					
Q10	3	1	2	3	4	2	32	37	23				
Q11	0	0	0	6	2	0	0	0	0	0			
Q12	0	0	0	0	0	0	0	0	0	0	4		
Q13	0	0	0	1	0	0	0	0	0	0	16	5	

Table 10. The number of shared websites between pairs of query topics.

With the data in Table 10, chain lengths and distances between queries can be calculated. For comparison, Eq. 4, Eq. 12, and Eq. 21 are applied with r = 1. Due to the large number of different chains between topics, it was not possible to manually calculate

distances between Q1 and all the other queries. However, sample distances between Q1 and some of the other queries are in Table 11.

	Eq. 4	Eq. 12	Eq. 21
Q2	32.33333	188	188
Q3	19	180	180
Q4	17.18182	178	178
Q5	99	196	196
Q6	14.38462	174	174
Q7	∞	265	330
Q10	65.66667	194	194
Q12	∞	379	558

Table 11. Sample distances between Q1 and some of the other 12 queries.

T_1 intersects with all the other 12 topics, except for T_7 and T_{12}. According to all three equations, the six closest queries to Q1 are, from closest to most distant, Q6, Q4, Q3, Q2, Q10, and Q5. Eq. 4 does not distinguish between Q7 and Q12, but Eq. 12 and Eq. 21 both rank Q7 as the 7^{th} closest and Q12 as the 8^{th} closest to Q1. The Eq. 12 value for $d(q_1, q_7)$ is equal to the length of chain $\{T_1, T_{10}, T_7\}$. The Eq. 12 value for $d(q_1, q_{12})$ is equal to the length of chain $\{T_1, T_4, T_{11}, T_{12}\}$. Distances from Q1 to queries Q8, Q9, Q11, and Q13 were not calculated. In order to find $d(q_1, q_8)$, $d(q_1, q_9)$, $d(q_1, q_{11})$, and $d(q_1, q_{13})$, chain lengths would have to be calculated for all possible chains between T_1 and the topics T_8, T_9, T_{11}, and T_{13}.

In the next two sections, the search process is described in terms the distance function's mathematical properties and a Cartesian product of sets.

4.2.11. SEPARATING QUERIES AND DOCUMENTS

The distance function from Distance Method 2 is a metric. Therefore, the distance function provides a basis β for a topology on the set of queries Q and β is defined as

$$\beta = [B_q(a);\ q \in Q \text{ and } a \text{ is a nonnegative real number}],$$

where $B_q(a)$ is the ball of radius a about query q. Since the topology on Q defined by β is a metric topology, Q has the following properties.

1. Q is Hausdorff (T_2)
2. Q is $T_1 \Rightarrow$ each point of Q is closed as a subset of Q
3. Every compact subset of Q is closed
4. Q is normal (T_4).

Topological separation properties are analogous to a user performing a search on the internet. As a user performs a search, the user is searching for documents that are relevant to the user's information need. If the user's initial query retrieves an insufficient number of relevant documents, then the user's concept function will react with a different query. As part of this search process, a search engine provides a tool to the user for separating irrelevant queries from relevant queries and, hence, irrelevant documents from relevant ones. The search process is then an iterative review of topics in T in an effort to separate the user's initial irrelevant query, i.e. the user's initial irrelevant search results list, from a set of relevant queries of a topic. The distance measure in Distance Method 2 is a tool for offering to the user a list of queries from the open set containing the user's previous query or queries, the open set being that set defined by the ball of some radius a

about the user's previous query or queries. It is the concept function describing the user that maps each search results list to the next query in the user's search. The concept function thus transports the user from their initial query to queries that retrieve more precise search results containing documents relevant to the user's information need. A distance measure could be a tool for influencing that concept function by further quantifying user behavior.

4.2.12. CONCEPT FUNCTIONS

A query selection policy is similar to a user concept as a function. A concept function could be defined by a set of user parameters, including those seven characteristics used in the simulation (Appendix B). The function parameters include all human characteristics, both psychological and intellectual, that affect a user's search behavior. As a user proceeds with their search, the concept function adjusts according to the search results and websites that the user observes.

A concept function can be defined in terms of a Cartesian product of sets. To do so, let W be an indexing set of -tuples that include topics (search results) returned by a search engine and user beliefs, desires, and intentions. Let $\{S_w\}_{w \in W}$ be a collection of sets where each S_w is a set of queries. Note that it's possible for $S_i \bigcap S_j \neq \{\}$ for any i and j. The Cartesian product of sets $\{S_w\}_{w \in W}$ is defined as the set

$$P = \prod_{w \in W} S_w = \left[f : W \to \bigcup_{w \in W} S_w ; f(w) \in S_w \text{ for all } w \in W \right].$$

A concept function is then an element of P.

An example for illustrating the idea of a concept function is a search for websites on the relationship between indoor carpets and allergies. Assume that a person submits "carpet allergies" as their initial query to begin the search. Call the initial query Query 1. After reviewing the search results list and downloading websites that are listed in the search results, the user's concept function reacts or determines a response according to the information obtained by the user from the search results list and the downloaded websites. For example, the user might obtain information about allergens that are commonly found in indoor carpets, such as dust mites and animal dander. Theoretically, there would then be a new and limited set of queries from which the user will select Query 2, such as queries that contain the terms "dust mites" and "animal dander". The set of possible next queries might not be completely new, because the user might still consider a query that he did not choose as Query 1, such as "carpet allergens". It's also possible new information obtained by the user will cause the user to no longer consider some queries that were potential initial queries. In any case, the set of candidates for Query 2 will be included in some set S_j, say S_2. The user then submits a Query 2, such as "eliminating dust mites". From the resultant search results list and from websites listed in the search results, the user will obtain new information and the user's concept function will adjust accordingly. If the user's information need is still not satisfied, then the user will then select a Query 3 from another set of possible queries, say S_3. This iterative process of submitting queries and reviewing search results will continue to be driven by the concept function until the user's information need is satisfied. This formalization of

users' behavior as concept functions has not yet been applied and tested, but is included here to foster further development by the research community at large.

CHAPTER 5:

SURVEYING USER QUERIES

While searching the literature, no work was found that addresses issues related to how users compose queries or if user beliefs and intentions are reflected in the form of user queries. To gain insight into those issues, a survey to gather queries related to specific information needs was composed. The survey is in Appendix A.

5.1. SURVEY GOALS

The goals of the survey were to address a list of issues, or questions, which are listed below. The actual questions or tasks included on the survey were designed to address one or more of the issues listed below.

1. Is a person's belief about their own level of expertise consistent with their actual level of expertise?
2. What are the differences between users in the number of keywords they use per query?
3. What are the differences between queries given by the same user?

a. Do some users have more variety in their queries? In other words, will the documents retrieved by all of their queries cover the information need? This might be similar to a breadth-first search.

b. Do some users focus their queries on a particular theme, or sub-topic? This might be similar to a depth first search.

4. Are there statistical correlations between the distance measure and the answers to the above questions or between the distance measure and user beliefs and/or intentions?

5.2. SPECIFYING AN INFORMATION NEED

For subjects to compose queries, they first must have an information need. For example, one information need might be to learn about any connections between indoor carpeting and allergies. Since an aim was to compare queries across subjects, it was necessary to present the same information needs to all subjects.

In selecting information needs for the survey, a balance had to be struck between choosing information needs that are familiar to many people versus information needs on which there are few experts. Information needs that are familiar to many people have the possible benefit of causing users to formulate many queries. However, the drawback of such information needs is that they might not generate queries that highlight differences between novices and experts. Although less familiar needs might highlight differences between novices and experts, they might result in many identical or very similar queries from novices.

A balance also had to be struck between needs that are associated with many words and/or synonyms versus needs that are commonly associated with very few and specific words. Needs that are more complex are often associated with varieties of words and/or synonyms. For example, finding information on the proper exercise program for a college student might result in queries with such words as exercise, fitness, weight lifting, aerobics, etc. In contrast, needs such as finding prices on a particular consumer product, e.g. a Trek 520 touring bicycle, are very specific and not very complex, thus resulting in not very interesting queries.

Finally, an effort was made to choose information needs that subjects could relate to in some capacity, even if they were not experts on the stated information need. For example, a person with little background in mathematics would have a difficult time formulating multiple queries for an abstract need such as chaos theory. However, more concrete information needs such as those related to the weather or automobiles can be related to by novices because they encounter the weather and automobiles on an every day basis and they play a significant role in their lives.

When wording the information need in the survey, an effort was made to avoid as much as possible words that a person would include in a query. The reason for doing so was to minimize the wording's influence on keywords that subjects included in their queries.

5.3. SUBJECT BACKGROUND INFORMATION

Some questions were included on the survey to gather background information on each subject that completed the survey. The background questions were geared towards gaining information on the subject's past experience with searching the internet on the information need that was specified in the survey, the subject's beliefs about their level of expertise on the specified information need, and the subject's frequency with which they use internet search engines on a weekly basis.

5.4. SURVEY TASKS

Each survey contained two search tasks for which subjects were asked to provide 5 queries each. After providing their queries for both search tasks and without seeing the search results of those queries, subjects were then asked to rate their queries with regards to the subject's intended level of recall and precision for the query. Subject's were asked to give their ratings on a scale of 1 to 5, with 1 being a low level of intended recall or precision and 5 being a high level of intended recall or precision.

After composing their queries, subjects were also asked to write descriptions of their queries in sentence form. The order of tasks was varied from survey to survey, so as to randomize any effects that might be caused by task order. For example, if a subject has difficulty rating their first search task queries, the subject might not put in the same amount of effort in rating their second search task queries.

CHAPTER 6:

SURVEY RESULTS AND ANALYSIS

6.1. NUMBER OF SUBJECTS AND QUERIES

A total of 39 subjects were administered the survey. Subjects were either undergraduate or graduate college students. Each survey contained questions and tasks related to two search tasks. One search task was related to the effects of weather on the United States economy. The other search task was related to cell phone plans. The search task descriptions for the weather/economy and cell phone tasks are below.

> Weather/economy task: "Find out what yearly weather trends do to the performance of the United States' economy."
>
> Cell phone task: "Find information on selecting a cell phone plan based on expected cell phone usage."

For the weather/economy task, 35 of the 39 subjects supplied 5 queries in response to the search task. For the cell phone task, 36 of the 39 subjects supplied 5 queries in response to the search task. In total, 188 queries were collected for the

economy task, of which 169 were unique (89.89%). For the cell phone task, a total of 192 queries were collected, of which 165 were unique (85.93%).

6.2. STATISTICAL TESTS, P-VALUES, AND TYPE I & II ERRORS

To look for relationships in the survey data, hypothesis tests were formulated and either regression analysis was performed or sample tests of means were performed for each hypothesis test. Formulas used in performing each of those statistical tests are included in Appendix C.

Regression analysis was performed when testing for trends in data over sequences of data values, e.g. 1, 2, 3, 4, etc. Sample tests of means were performed when comparing means of samples, e.g. mean information need ratings for the weather economy task vs. mean information need ratings for the cell phone task, or groups of data values in a sequence, e.g. mean precision ratings when the recall rating is 2 vs. mean precision ratings when the recall rating is either 3, 4, or 5.

For each hypothesis test, a null hypothesis, H_0, and one or more alternative hypotheses, H_i, were formulated. The null hypothesis is the hypothesis that is tested, e.g. $\mu_X = \mu_Y$, by looking for evidence in the data that contradicts the null hypothesis and supports the alternative hypothesis, e.g. $\mu_X > \mu_Y$. A real-world example is a murder trial in which the defendant is considered innocent until proven guilty. In this case, the null hypothesis is that the defendant is innocent of murder and the alternative hypothesis is the defendant is guilty of murder. Presenting sufficient evidence to convict the defendant is the responsibility of the prosecution. The two possible types of errors when making

conclusions about a hypothesis test are Type I and Type II errors. A Type I error is committed by rejecting H_0 when H_0 is true, i.e. the jury finds the defendant guilty when in fact the defendant is innocent. A Type II error is committed by not rejecting H_0 when H_0 is false, i.e. when the jury finds the defendant innocent when in fact the defendant is guilty. When performing statistical tests in this thesis, the probability of committing Type I errors is of most concern. The probability of committing a Type I error is commonly denoted as α and a statistical test's estimate of α is given as a p-value. The p-value is estimated from the statistical test's estimated t-value, which is the estimated number of standard deviations that the sample statistic is away from the population statistic stated in the null hypothesis, assuming the null hypothesis to be true. In this thesis, p-values are considered statistically significant if they are less than the critical value of 0.05, i.e. if it can be stated with at least 95% confidence that the null hypothesis is false.

6.3. BACKGROUND AND INFORMATION NEED STATISTICS

The following statistics are listed in Table 12. For the weather/economy task, no subjects had performed an internet search on the task prior to taking the survey. For the cell phone task, 12 of the 39 subjects, or approximately 30.7%, had performed an internet search on the task prior to taking the survey. For the weather/economy task, only one of the 39 subjects, or approximately 2.6%, claimed to have some background related to the search task. For the cell phone task, 23 of the 39 subjects, or approximately 59.0%, claimed to have some background related to the search task.

Task	Task Search History	Task Background
Weather/Economy	0%	2.6%
Cell Phone	30.7%	59.0%

Table 12. Percentage of subjects with some task background.

For the weather/economy task, the average information need given by the subjects was 4.49 on a scale of 1 to 5, with 1 being no information need, i.e. the subject considered himself to be an expert on the search task, and 5 being the greatest information need, i.e. the subject considered himself to be a novice on the search task. For the cell phone task, the average information need was 3.13. Average information need is illustrated graphically in Figure 6.

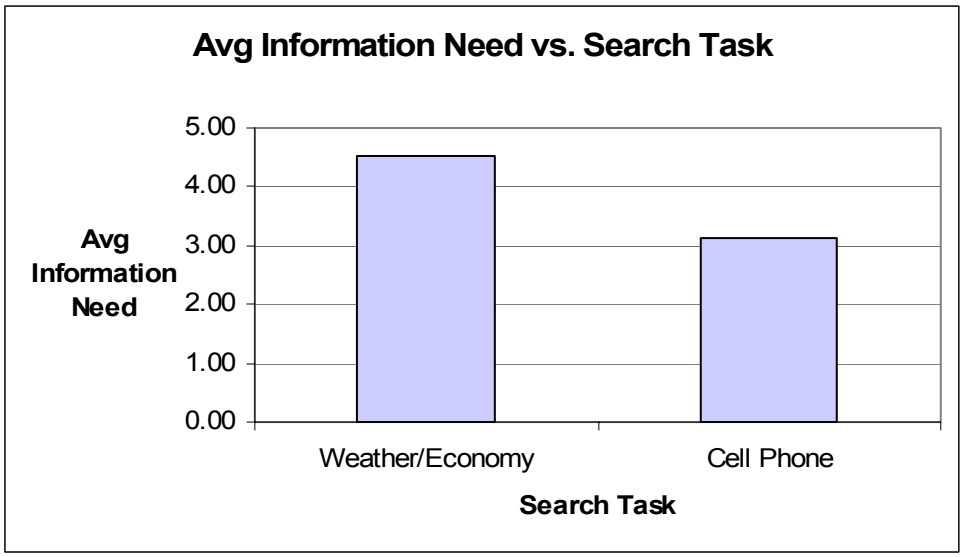


Figure 6. Average subject information need by search task.

Out of the 39 subjects, 8 claimed to perform from 0 to 5 internet searches per week with a search engine, 18 claimed to perform from 6 to 10, and 13 claimed to perform more than 10. These numbers are illustrated graphically in Figure 7.

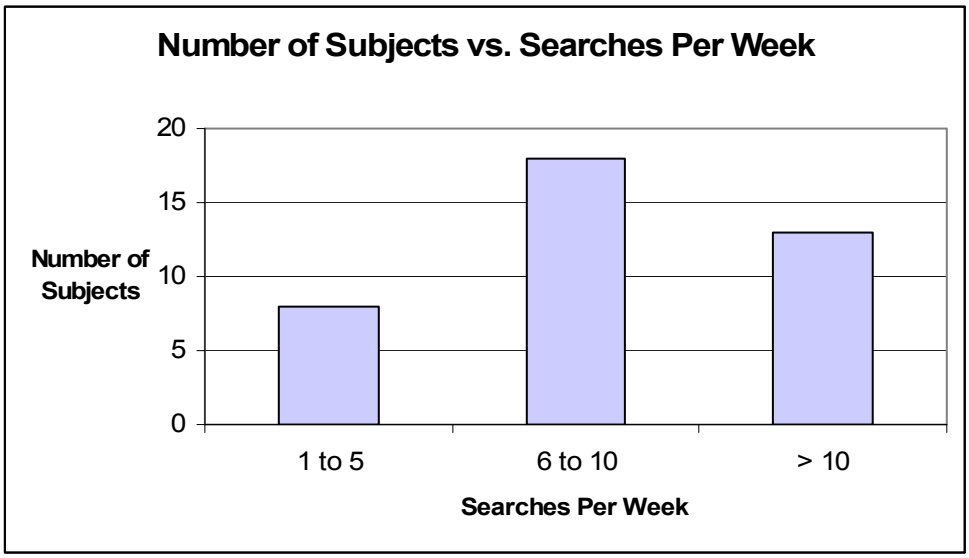


Figure 7. Estimated number of internet searches per week performed with a search engine by subjects.

Average Information Need was plotted against the number of Searches per Week for the cell phone task in Figure 8. Data from the weather/economy task was not plotted, because for the weather/economy task there was little variation in the information need ratings, i.e. ratings of either 4 or 5.

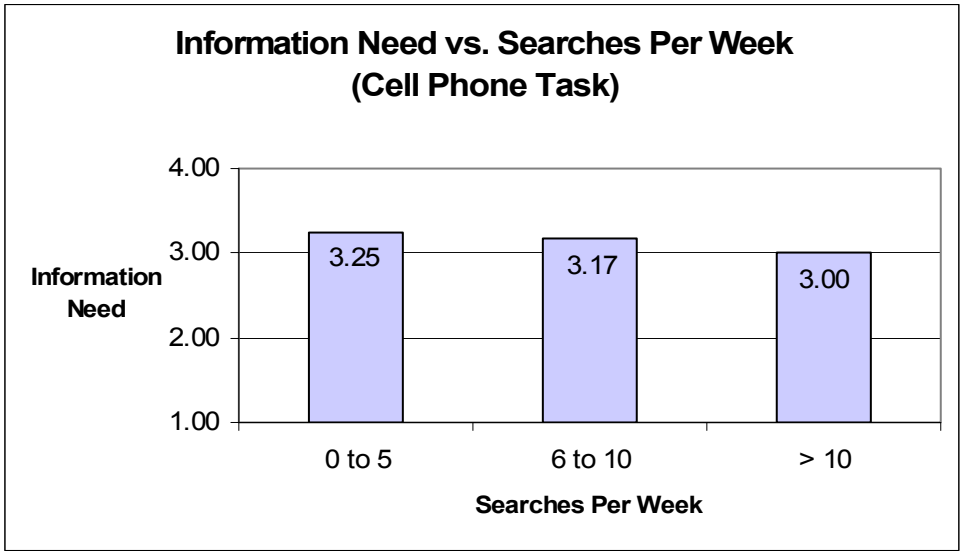


Figure 8. Information Need vs. Searches Per Week for the cell phone task.

In Figure 8, average information need decreases slightly as the number of searches per week increases. To test the statistical significance of the data in Figure 8, three tests were performed to compare average information need for subjects who perform 0 to 5 searches per week, μ_X, average information need for subjects who perform 6 to 10 searches per week, μ_Y, and average information need for subjects who perform greater than 10 searches per week, μ_Z. The first hypothesis test was to compare μ_X and μ_Z, as stated by the following null and alternative hypotheses.

H_0: $\mu_X = \mu_Y$

H_1: $\mu_X > \mu_Y$

The survey data provided estimates of μ_X and μ_Y and those estimates were 3.25 and 3.17, respectively. A one-sided t-test assuming unequal variances resulted in a t-value of 1.7709 (13 degrees of freedom) with a p-value equal to 0.4133. The p-value is greater than the critical value of 0.05; therefore the null hypothesis is not rejected.

The second hypothesis test on Figure 8 data was to compare μ_X and μ_Z, as stated by the following null and alternative hypotheses.

H_0: $\mu_X = \mu_Z$

H_1: $\mu_X > \mu_Z$

The survey data provided an estimate of μ_Z equal to 3.00. A one-sided t-test assuming unequal variances resulted in a t-value of 1.7531 (15 degrees of freedom) with a p-value equal to 0.2721. The p-value is greater than the critical value of 0.05; therefore the null hypothesis is not rejected.

The third hypothesis test on Figure 8 data was to compare μ_Y and μ_Z, as stated by the following null and alternative hypotheses.

H_0: $\mu_Y = \mu_Z$

H_1: $\mu_Y > \mu_Z$

A one-sided t-test assuming unequal variances resulted in a t-value of 1.7081 (25 degrees of freedom) with a p-value equal to 0.3057. The p-value is greater than the critical value of 0.05; therefore the null hypothesis is not rejected.

In conclusion with regards to Figure 8, there was no statistically significant relationship between the number of internet searches per week performed by subjects and their stated information need for the cell phone task.

6.4. QUERY STATISTICS

As an example, listed below are some weather/economy queries collected from the surveys.

1. United States economy factors
2. weather United States economy
3. temperature and “gross national product” data
4. weather effects US economy
5. weather trends and US economy

As another example, listed below are some cell phone queries collected from the surveys.

1. student plans cellular services U.S.
2. cell phone plan expected usage
3. cell phone minutes price
4. verizon wireless calling plans
5. “cell phone plan” and “expected usage”

A number of query statistics were extracted from the survey queries, including those listed below. Table 13 contains numbers for the statistics listed below and those numbers are illustrated graphically in Figure 9, Figure 10, and Figure 11. Following each figure, a statistically test is performed on the data represented in the figure.

1. Average number of keywords per query
2. Average number of query keywords that appear in the search task description
3. Average percent of query keywords that appear in the search task description

Task	Keywords Per Query	Keywords in Task Description	% Keywords in Task Description
Weather/Econ.	4.63	3.44	75.01
Cell Phone	3.94	2.64	66.63

Table 13. Query keyword averages by task.

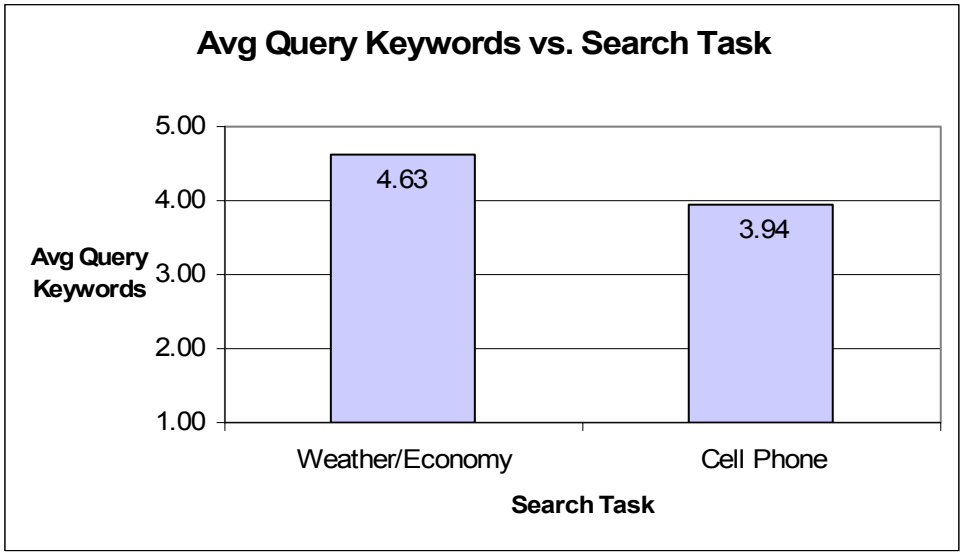


Figure 9. Average number of query keywords by search task

To analyze statistically whether subjects use more keywords for the weather/economy task than for the cell phone task, the following null and alternative hypotheses were tested with regards to the Average Number of Keywords per Query for the weather/economy task, μ_X, and the average Number of Keywords per Query for the cell phone task, μ_Y.

H_0: $\mu_X = \mu_Y$

H_1: $\mu_X > \mu_Y$

Assuming unequal variances, a one-tailed two-sample test of means was performed. The resulting t-value was equal to 4.27 (366 degrees of freedom), which corresponds to a p-value of 1.27×10^{-5}. The p-value is less than the critical value of 0.05; therefore the null hypothesis is rejected.

The average number of keywords for the weather/economy search task should not be greater due to the phrase "United States" in the search task description. Note that keywords such as "U.S." and "US" are abbreviations of "United States" and those keywords were counted as two words when counting query keywords and when counting query keywords that appear in the weather/economy search task description. Any statistical implications caused by handling such abbreviations as two words may be offset by considering the prevalence of the phrase "cell phone" in the cell phone task queries. The phrase "cell phone" appeared in 139 of the 192 total cell phone queries and 112 of the 165 unique cell phone queries. Some form of "United States" or its abbreviations appeared in 102 of the 188 total weather/economy queries and 94 of the 169 unique weather/economy queries.

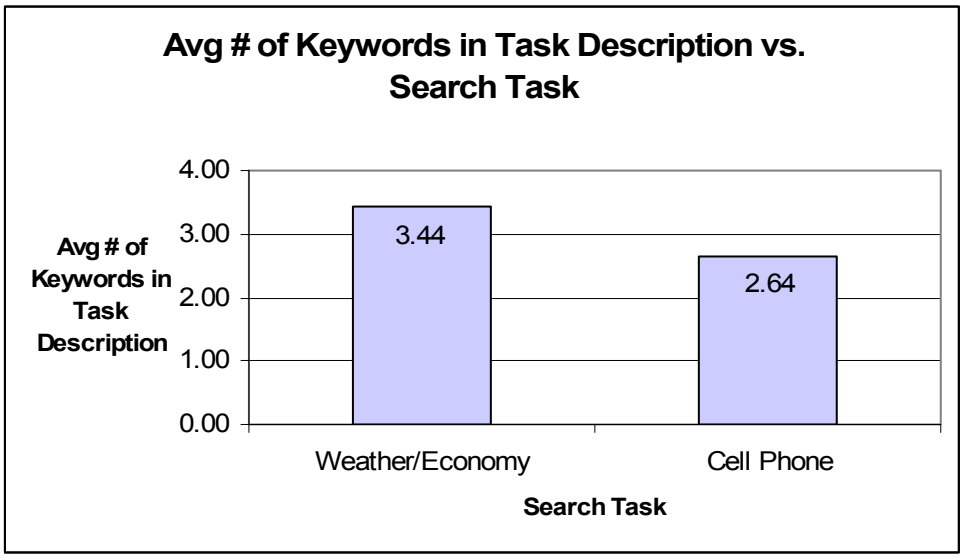


Figure 10. Average number of query keywords in the search task description by task.

To analyze statistically whether subjects use more keywords from the task description in their weather/economy queries than in their cell phone queries, the following null and alternative hypotheses were tested with regards to the Average Number of Keywords per Query in the Task Description for the weather/economy task, μ_X, and the Average Number of Keywords per Query in the Task Description for the cell phone task, μ_Y.

H_0: $\mu_X = \mu_Y$

H_1: $\mu_X > \mu_Y$

Assuming unequal variances, a one-tailed two-sample test of means was performed. The resulting t-value was equal to 5.66 (360 degrees of freedom), which corresponds to a p-value of 1.52×10^{-8}. The p-value is less than the critical value of 0.05; therefore the null hypothesis is rejected.

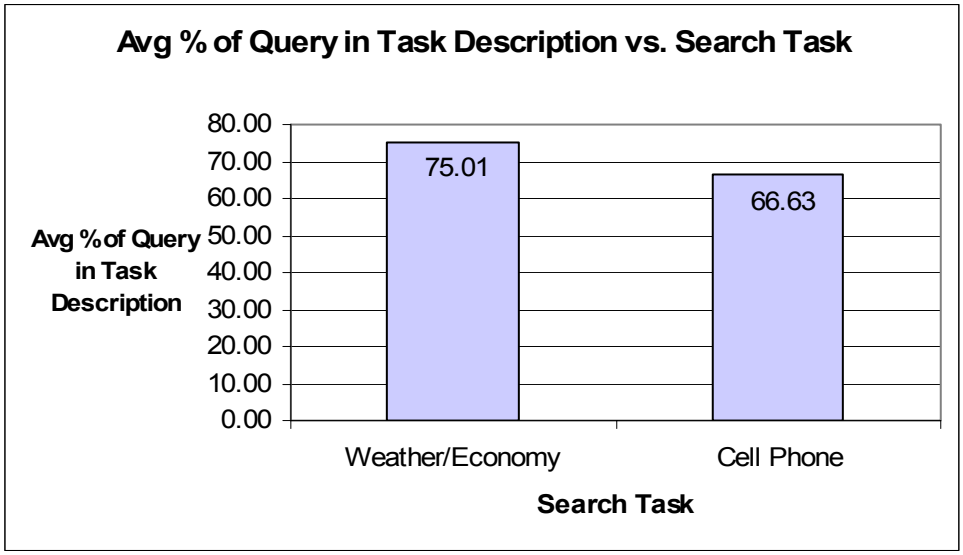


Figure 11. Average percent of query keywords in the search task description by task.

As an example of the percent of query keywords in the search task description, let a particular query be composed of five keywords. If three of those keywords are from the search task description, then 60% of the query keywords are in the search task description. To analyze statistically whether subjects take a larger percentage of their query keywords from the weather/economy task description than from the cell phone task description, the following null and alternative hypotheses were tested with regards to the Average Percent of Query Keywords in the Task Description for the weather/economy queries, μ_X, and the Average Percent of Query Keywords in the Task Description for the cell phone queries, μ_Y.

H_0: $\mu_X = \mu_Y$

H_1: $\mu_X > \mu_Y$

Assuming unequal variances, a one-tailed two-sample test of means was performed. The resulting t-value was equal to 3.38 (369 degrees of freedom), which

corresponds to a p-value of 0.00041. The p-value is less than the critical value of 0.05; therefore the null hypothesis is rejected.

As described previously, each subject was asked to provide five queries for each search task. The average percent of query keywords in the search task description was calculated across search tasks by subject query number. The data is listed in Table 14 and the data is illustrated graphically in Figure 12.

Query #1	Query #2	Query #3	Query #4	Query #5
84.47%	69.92%	69.25%	65.05%	64.45%

Table 14. Average percent of query keywords in the search task description by subject query number.

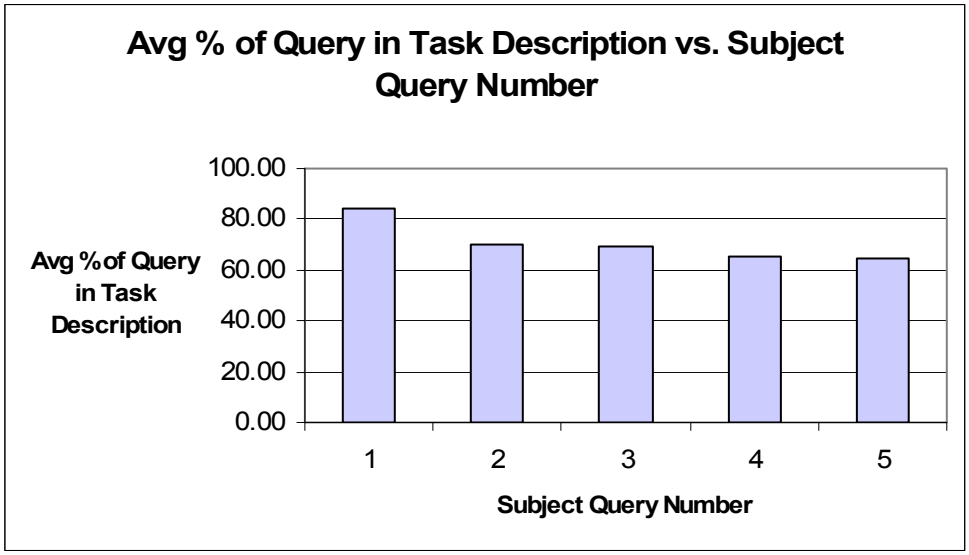


Figure 12. Average percent of query keywords in the search task description vs. subject query number.

To analyze statistically whether subjects compose successive queries by using more of their own words rather than search task words, regression analysis was performed on the following model.

$$y = \beta_o + \beta_1 x_1$$

In the model, y is the percent of query keywords in the search task description and x_1 is the subject query number, where x_1 takes on values 1, 2, 3, 4, and 5. Regression analysis resulted in β_o and β_1 estimates of 84.21 and -4.54, respectively. To test the statistical significance of the β_1 estimate, the following null and alternative hypotheses were tested.

H_0: $\beta_1 = 0$

H_1: $\beta_1 < 0$

The 95% confidence interval for β_1 is (-6.25, -2.84), which does not include 0. The t-value for β_1 was equal to -5.24 with an associated p-value of 2.7×10^{-7}. The p-value is less than the critical value of 0.05; therefore the null hypothesis is rejected. However, the R-value for the model was only 0.065, suggesting that little variation was accounted for with the model. Although the estimated value of β_1 appears small, it is statistically significant and indicates that across both the weather/economy and cell phone search tasks, with each successive query subjects showed a tendency to decrease the percentage of their query keywords from the search task description. As a result, the average percent of a query's keywords that came from the search task description dropped from 84.47% in subjects' first queries to 64.45% in subjects' fifth queries.

6.5. RECALL AND PRECISION STATISTICS

For each of their queries, each subject was asked to rate the degree of their intended precision and recall behind their queries. They were asked to rate their queries on a scale of 1 to 5, with 1 being a low intended degree of recall or precision and 5 being a high degree of recall or precision. For example, a recall rating of 5 would be appropriate if retrieving all relevant documents was of the utmost importance. At the same time, if a person wasn't concerned at all about retrieving irrelevant documents, a precision rating of 1 would be appropriate. Data for the numbers of times each query rating was given for any query is listed in Table 15, which includes data from each search task. Totals are plotted in Figure 13.

		Rating				
Property	**Task**	**1**	**2**	**3**	**4**	**5**
Recall	Weather/Economy	2	19	61	58	43
	Cell Phone	5	14	42	68	59
	Total:	*7*	*33*	*103*	*126*	*102*
Precision	Weather/Economy	21	52	56	31	18
	Cell Phone	26	32	71	33	20
	Total:	*47*	*84*	*127*	*64*	*38*

Table 15. The number of times each precision rating was given for any query. Total tallies are given and they are also broken down by search task.

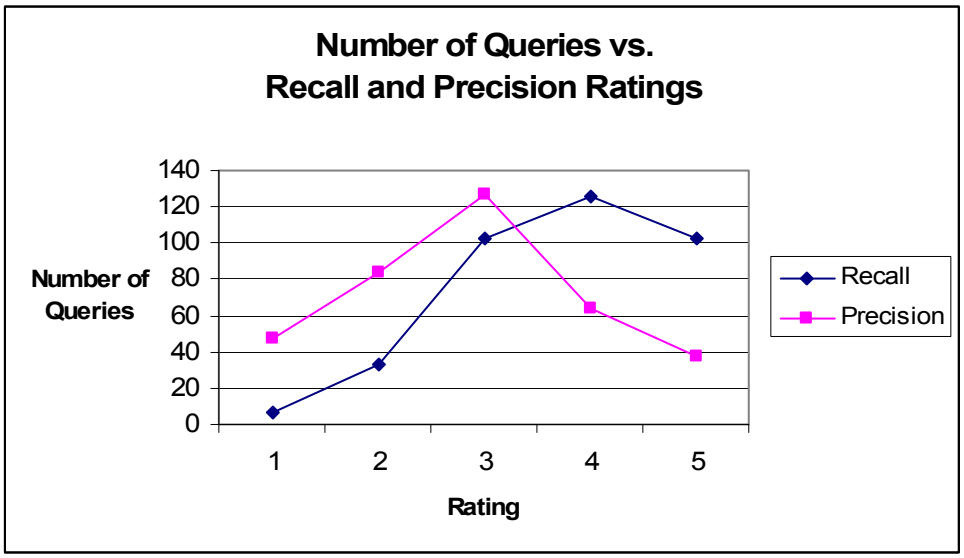


Figure 13. Tallies of recall and precision ratings. Tallies are summed over both search tasks.

Figure 14 shows average Recall − Precision ratings versus the number of internet searches per week. There appear to be similarities in precision and recall ratings for subjects that perform 0 to 5 searches per week and subjects that perform greater than 10 searches per week, with subjects that perform 6 to 10 searches per week showing slightly different precision and recall ratings.

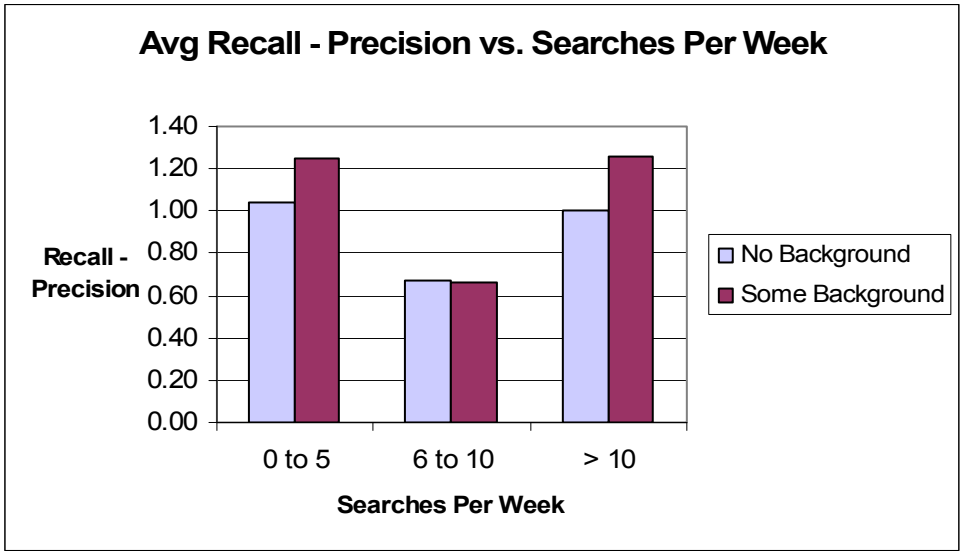


Figure 14. Average recall – precision rankings vs. the number of internet searches per week performed by a subject.

To test the statistical significance of the data in Figure 14, three tests were performed to compare Average Recall – Precision ratings for subjects who perform 0 to 5 searches per week, μ_X, Average Recall – Precision ratings for subjects who perform 6 to 10 searches per week, μ_Y, and Average Recall – Precision ratings for subjects who perform greater than 10 searches per week, μ_Z. The first hypothesis test was to compare μ_X and μ_Z, as stated by the following null and alternative hypotheses.

H_0: $\mu_X = \mu_Z$

H_1: $\mu_X \neq \mu_Z$

The survey data provided estimates of μ_X and μ_Z and those estimates were 1.10 and 1.09, respectively. A two-sided t-test assuming unequal variances resulted in a t-value of 0.0838 (130 degrees of freedom) with a p-value equal to 0.9333. The p-value is greater than the critical value of 0.05; therefore the null hypothesis is not rejected.

The second hypothesis test on Recall – Precision ratings was to compare μ_X and μ_Y, as stated by the following null and alternative hypotheses.

H_0: $\mu_X = \mu_Y$

H_1: $\mu_X > \mu_Y$

The survey data provided an estimate of μ_Y equal to 0.67. A one-sided t-test assuming unequal variances resulted in a t-value of 2.1043 (147 degrees of freedom) with a p-value equal to 0.0185. The p-value is less than the critical value of 0.05; therefore the null hypothesis is rejected.

The third hypothesis test on Recall – Precision ratings was to compare μ_Z and μ_Y, as stated by the following null and alternative hypotheses.

H_0: $\mu_Z = \mu_Y$

H_1: $\mu_Z > \mu_Y$

A one-sided t-test assuming unequal variances resulted in a t-value of 2.7209 (273 degrees of freedom) with a p-value equal to 0.0035. The p-value is less than the critical value of 0.05; therefore the null hypothesis is rejected.

To analyze further the data from Figure 14, precision and recall ratings were plotted independently versus the number of searches per week in Figure 15 and Figure 16. The data Figure 15 and Figure 16 were tested for statistical significance and a description of those tests follows Figure 16.

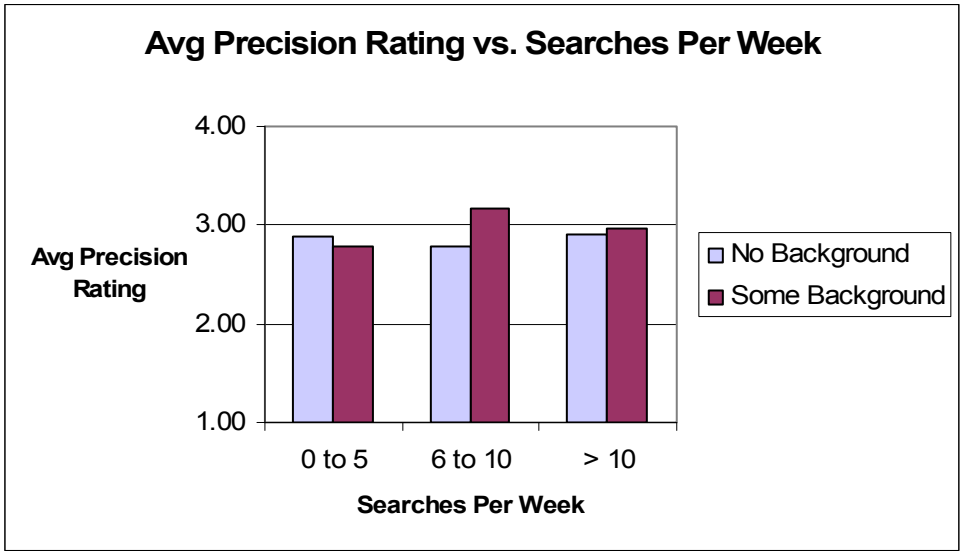


Figure 15. Subjects' average precision rating of their queries vs. the number of internet searches they perform per week.

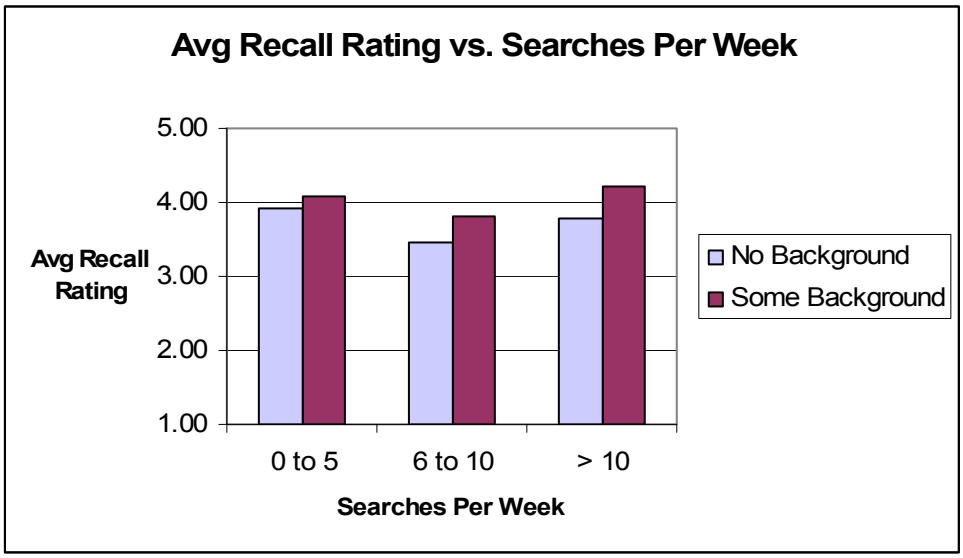


Figure 16. Subjects' average recall rating of their queries vs. the number of internet searches they perform per week.

Figure 15 suggests that, across both search tasks, subjects who perform 0 to 5 searches per week and subjects who perform greater than 10 searches per week show

similar precision ratings, with both groups showing possibly lower precision ratings than subjects who perform 6 to 10 searches per week. To test the statistical significance of the data, three tests were performed to compare precision ratings for subjects who perform 0 to 5 searches per week, μ_X, precision ratings for subjects who perform 6 to 10 searches per week, μ_Y, and precision ratings for subjects who perform greater than 10 searches per week, μ_Z. The first hypothesis test was to compare μ_X and μ_Z, as stated by the following null and alternative hypotheses.

H_0: $\mu_X = \mu_Z$

H_1: $\mu_X \neq \mu_Z$

The survey data provided estimates of μ_X and μ_Z and those estimates were 2.86 and 2.92, respectively. A two-sided t-test assuming unequal variances resulted in a t-value of 1.9754 (155 degrees of freedom) with a p-value equal to 0.7087. The p-value is greater than the critical value of 0.05; therefore the null hypothesis is not rejected.

The second hypothesis test on Precision ratings was to compare μ_X and μ_Y, as stated by the following null and alternative hypotheses.

H_0: $\mu_X = \mu_Y$

H_1: $\mu_X < \mu_Y$

The survey data provided an estimate of μ_Y equal to 2.89. A one-sided t-test assuming unequal variances resulted in a t-value of 1.9765 (145 degrees of freedom) with a p-value equal to 0.8375. The p-value is greater than the critical value of 0.05; therefore the null hypothesis is not rejected.

The third hypothesis test on Precision ratings was to compare μ_Z and μ_Y, as stated by the following null and alternative hypotheses.

H_0: $\mu_Z = \mu_Y$

H_1: $\mu_Z < \mu_Y$

A one-sided t-test assuming unequal variances resulted in a t-value of 1.9701 (235 degrees of freedom) with a p-value equal to 0.8219. The p-value is greater than the critical value of 0.05; therefore the null hypothesis is not rejected.

Figure 16 suggests that, across both search tasks, subjects who perform 0 to 5 searches per week and subjects who perform greater than 10 searches per week show similar recall ratings, with both groups showing possibly higher recall ratings than subjects who perform 6 to 10 searches per week. To test the statistical significance of the data, three tests were performed to compare recall ratings for subjects who perform 0 to 5 searches per week, μ_X, recall ratings for subjects who perform 6 to 10 searches per week, μ_Y, and recall ratings for subjects who perform greater than 10 searches per week, μ_Z. The first hypothesis test was to compare μ_X and μ_Z, as stated by the following null and alternative hypotheses.

H_0: $\mu_X = \mu_Z$

H_1: $\mu_X \neq \mu_Z$

The survey data provided estimates of μ_X and μ_Z and those estimates were 3.97 and 3.93, respectively. A two-sided t-test assuming unequal variances resulted in a t-value of 1.9730 (183 degrees of freedom) with a p-value equal to 0.7434. The p-value is greater than the critical value of 0.05; therefore the null hypothesis is not rejected.

The second hypothesis test on recall ratings was to compare μ_X and μ_Y, as stated by the following null and alternative hypotheses.

H_0: $\mu_X = \mu_Y$

H_1: $\mu_X > \mu_Y$

The survey data provided an estimate of μ_Y equal to 3.56. A one-sided t-test assuming unequal variances resulted in a t-value of 1.9733 (179 degrees of freedom) with a p-value equal to 0.0011. The p-value is less than the critical value of 0.05; therefore the null hypothesis is rejected.

The third hypothesis test on recall ratings was to compare μ_Z and μ_Y, as stated by the following null and alternative hypotheses.

H_0: $\mu_Z = \mu_Y$

H_1: $\mu_Z > \mu_Y$

A one-sided t-test assuming unequal variances resulted in a t-value of 1.9695 (250 degrees of freedom) with a p-value equal to 0.0027. The p-value is less than the critical value of 0.05; therefore the null hypothesis is rejected.

In conclusion with regards to Figure 14, Figure 15, and Figure 16, either the data in Figure 14 shows evidence that a person's internet search experience has an effect on their recall and precision judgments or the degree to which a person's judgments on precision and recall differ is similar to the degree to which they perceive their actual internet search engine experience. Further analysis of Figure 15 and Figure 16 indicates that this effect due to search experience is manifests itself in subjects' recall ratings.

Subjects showed some interesting tradeoffs between precision and recall ratings, as illustrated graphically in Figure 17. The graph in Figure 17 shows a precision/recall tradeoff in user intentions that is slightly different from the typical precision/recall tradeoff seen in search engine performance that is illustrated in Figure 1.

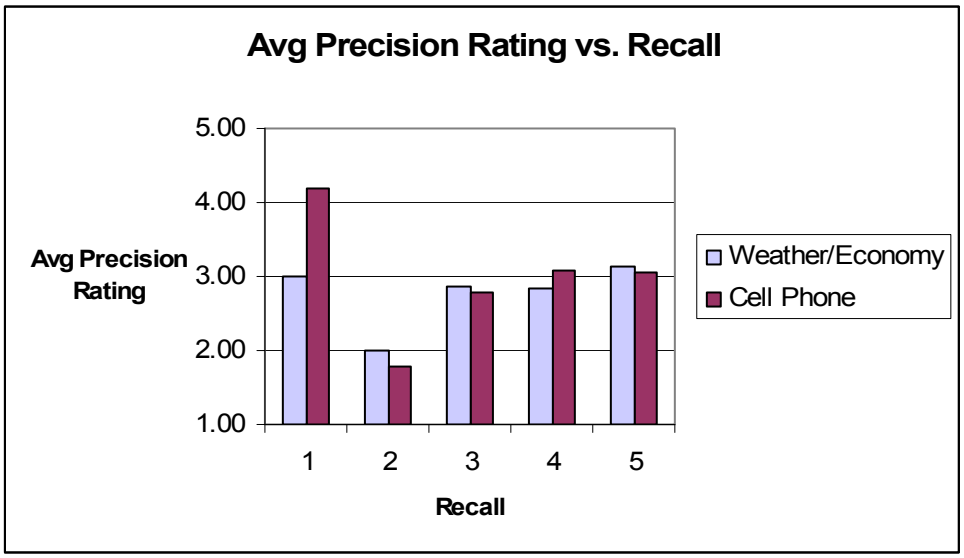


Figure 17. Subjects' average precision rating of their queries vs. their recall ratings for their queries.

To test the significance of the data illustrated in Figure 17 across both search tasks, the following were defined.

μ_A = a subject's Average Precision Rating when the subject's Recall Rating is 1

μ_B = a subject's Average Precision Rating when the subject's Recall Rating is 2

μ_C = a subject's Average Precision Rating when the subject's Recall Rating is 3

μ_D = a subject's Average Precision Rating when the subject's Recall Rating is 4

μ_E = a subject's Average Precision Rating when the subject's Recall Rating is 5

From Figure 17, it appears that when subjects give a recall rating equal to 1, they tend to give a precision rating greater than 3; when subjects give a recall rating equal to 2, they tend to give a precision rating also equal to 2. To test the statistical significance of this data, the following hypothesis test was performed:

H_0: $\mu_A = \mu_B + 1$

H_1: $\mu_A > \mu_B + 1$.

A one-sided t-test assuming unequal variances resulted in a t-value of 1.2755 (7 degrees of freedom) with a p-value equal to 0.1214. The p-value is greater than the critical value of 0.05; therefore the null hypothesis is not rejected. The failure to reject H_0 might be due to only 7 recall ratings being equal to 1, thus making the sample size too small to make the data statistically significant.

From Figure 17, it appears that $\mu_C = \mu_D = \mu_E$. To test the statistical significance of this data, the following hypothesis test was performed:

H_0: $\mu_C = \mu_D = \mu_E$

H_1: $\mu_C \neq \mu_D$

H_2: $\mu_C \neq \mu_E$

H_3: $\mu_D \neq \mu_E$.

Two-sided tests assuming unequal variances resulted in t-values of 1.6523 (205 degrees of freedom), 1.6528 (191 degrees of freedom), and 1.6536 (176 degrees of freedom) and p-values of 0.3011, 0.1198, and 0.4369 for H_1, H_2, and H_1, respectively. All three p-values are greater than the critical value of 0.05; therefore the null hypothesis is not rejected.

The final statistical test performed on the data in Figure 17 was to analyze whether μ_C, μ_D, and μ_E are greater than μ_B. Since H_0 was not rejected in the previous test, the data for recall ratings 3, 4, and 5 were pooled together. Letting μ_F be a subject's Average Precision Rating when the subject's Recall Rating is 3, 4, or 5, the following hypothesis test was performed:

H_0: $\mu_F = \mu_B$

H_1: $\mu_F > \mu_B$.

A one-sided t-test assuming unequal variances resulted in a t-value of 6.0421 (35 degrees of freedom) with a p-value equal to 3.4×10^{-7}. The p-value is less than the critical value of 0.05; therefore the null hypothesis is rejected.

In summary with regards to Figure 17, when subjects gave a low recall rating equal to 1, they demonstrated a clear intention to retrieve specific search results by giving the highest precision ratings. However, the high precision ratings were not statistically significant and further study is needed to confirm this relationship. When subjects gave recall ratings of 2 they also tended to give precision ratings of two, which might indicate low confidence on the part of the subjects. Subjects that gave recall ratings of 4 and 5 tended to give precision ratings of 3, possibly showing a clear intention to retrieve as many relevant documents as possible with less importance placed on precision. It's also possible that subjects were hesitant about either trying to achieve more precision or about making claims on their queries' precision.

Subjects showed different strategies for achieving more precision with their queries. Figure 18 and Figure 19 illustrate graphically the strategies applied between the

weather/economy task and the cell phone task. Statistical tests on the data in Figure 18 and Figure 19 follow each figure.

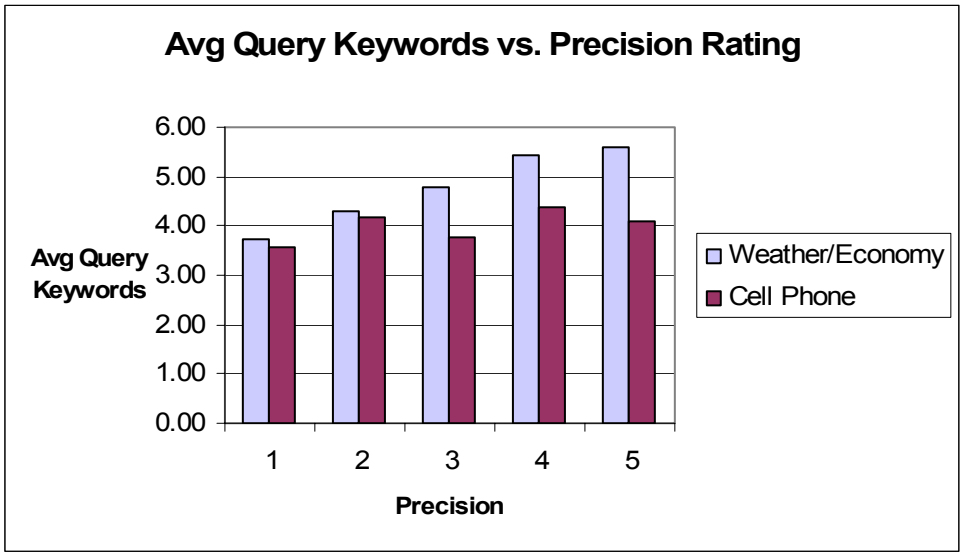


Figure 18. Average number of keywords per query vs. subjects' precision ratings

Figure 18 indicates that, with the weather/economy task, subjects tended to increase the number of keywords in their queries as they tried to achieve more precision. To analyze statistically whether subjects increased the number of keywords in their weather/economy queries as their precision ratings of those queries increased, regression analysis was performed on the following model.

$$y = \beta_o + \beta_1 x_1$$

In the model, y is the number of keywords in a weather/economy query and x_1 is the subject's precision rating of that query, where x_1 takes on values 1, 2, 3, 4, and 5. Regression analysis resulted in β_o and β_1 estimates of 3.29 and 0.50, respectively, indicating that the number of keywords in a weather/economy query increased by 0.50

with each increase in a query's precision rating. To test the statistical significance of the β_1 estimate, the following null and alternative hypotheses were tested.

H_0: $\beta_1 = 0$

H_1: $\beta_1 > 0$

The 95% confidence infterval for β_1 is (0.30, 0.71), which does not include 0. The t-value for the β_1 estimate was equal to 4.83 with an associated p-value of 2.97×10^{-6}. The p-value is less than the critical value of 0.05; therefore the null hypothesis is rejected. A similar test was performed on the cell phone data in Figure 18 and no statistically significant relationship was found between the number of keywords in a cell phone query and the subject's precision rating of that query.

The data in Figure 19 was statistically analyzed to detect any statistically significant relationships between the percentage of a query that appears in the search task description and the subject's precision rating of the query. A description of that analysis follows Figure 19.

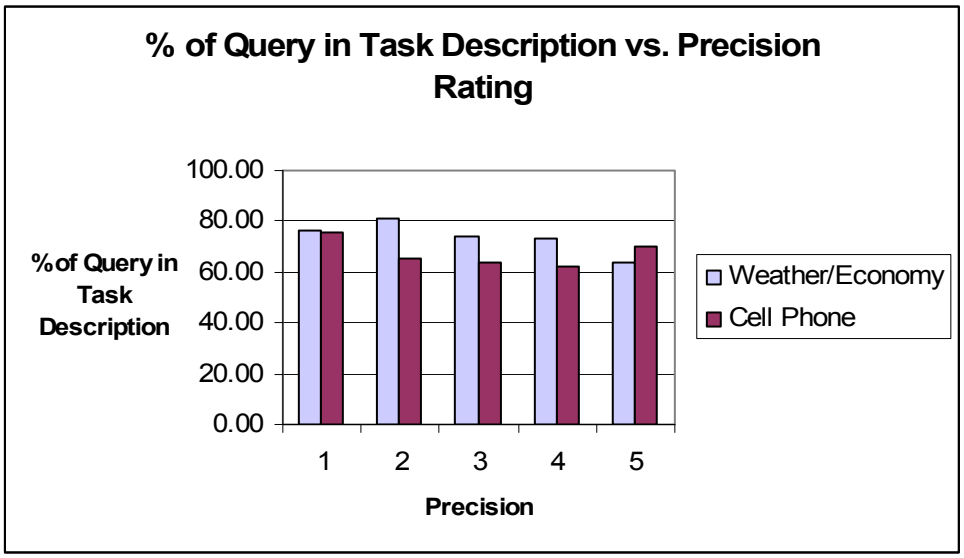


Figure 19. Percentage of query keywords in the search task description vs. subjects' precision ratings of their queries, by search task.

Figure 19 gives some indication that, with both tasks, subjects tended to decrease the percentage of their query keywords that appear in the search task description as they tried to achieve more precision with those queries. To analyze statistically whether, for the weather/economy task, subjects decreased the percentage of their query keywords that appeared in the search task description as they tried to achieve more precision with those queries, regression analysis was performed on the following model.

$$y = \beta_o + \beta_1 x_1$$

In the model, y is the percentage of a query's keywords that appear in the weather/economy search task description and x_1 is the subject's precision rating of that weather/economy query, where x_1 takes on values 1, 2, 3, 4, and 5. Regression analysis resulted in β_o and β_1 estimates of 85.01 and -3.46, respectively, indicating that the percentage of a query's keywords that appear in the weather/economy search task

description decreased by 3.46 with each increase in a query's precision rating. To test the statistical significance of the β_1 estimate, the following null and alternative hypotheses were tested.

H_0: $\beta_1 = 0$

H_1: $\beta_1 < 0$

The 95% confidence interval for β_1 is (-6.27, -0.65), which does not include 0. The t-value for the β_1 estimate was equal to -2.43 with an associated p-value of 0.0160. The p-value is less than the critical value of 0.05; therefore the null hypothesis is rejected. However, the estimated value of β_1, i.e. 3.46, is so small that it is arguably trivial and of little practical significance. A similar test was performed on the cell phone data in Figure 19 and no statistically significant relationship was found between the percentage of query keywords that appeared in the cell phone search task description and the precision ratings of those cell phone queries.

In summary with regards to Figure 18 and Figure 19, for the weather/economy task, subjects demonstrated a strategy to increase precision by increasing the number of keywords in their queries. For the cell phone task, subjects did not try to increase precision by increasing the number of keywords in their queries. Instead, for the cell phone task, subjects demonstrated a strategy to increase precision by substituting words. With the cell phone task, which was the task on which subjects had more background, a number of subjects clearly believed they could achieve more precision by substituting keywords without it being necessary to increase the number of keywords in the query.

Figure 20 indicates that the difference between subjects' recall and precision ratings for their first queries was almost twice as much as that for their subsequent queries. Statistical tests on the data represented in Figure 20 follow Figure 20.

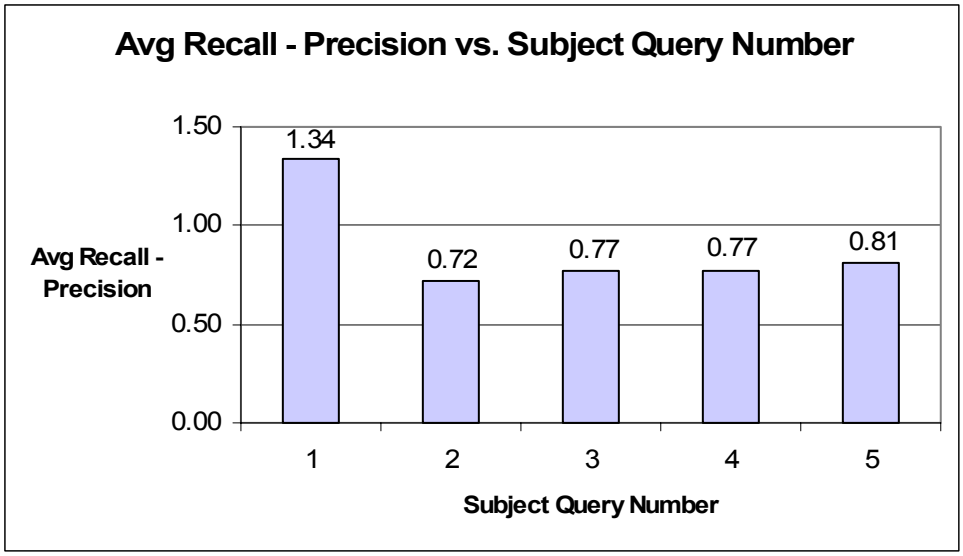


Figure 20. Subjects' average recall rating of their queries – their average precision rating of their queries vs. the subject's query number.

To test the significance of the data illustrated in Figure 20 across both search tasks, the following were defined. In the definitions, Recall – Precision is the difference between a query's recall and precision ratings as supplied by the subject who composed the query.

μ_A = the average value of Recall – Precision for all subjects' first queries

μ_B = the average value of Recall – Precision for all subjects' second queries

μ_C = the average value of Recall – Precision for all subjects' third queries

μ_D = the average value of Recall – Precision for all subjects' fourth queries

μ_E = the average value of Recall – Precision for all subjects' fifth queries

From Figure 20, it appears that $\mu_B = \mu_C = \mu_D = \mu_E$. To test the statistical significance of this data, the following hypothesis test was performed:

H_0: $\mu_B = \mu_C = \mu_D = \mu_E$

H_1: $\mu_B < \mu_C$

H_2: $\mu_B < \mu_D$

H_3: $\mu_B < \mu_E$.

H_4: $\mu_C < \mu_D$

H_5: $\mu_C < \mu_E$

H_6: $\mu_D < \mu_E$

For all six alternative hypotheses, one-sided tests assuming unequal variances resulted in p-values greater than the critical value of 0.05. For example, for H_3, a t-value of 0.4103 was found with a p-value equal to 0.3411. Therefore, the null hypothesis is not rejected.

The final statistical test performed on the data in Figure 20 was to analyze whether μ_B, μ_C, μ_D, and μ_E are less than μ_A. Since H_0 was not rejected in the previous test, the data for queries 1, 2, 3, 4, and 5 were pooled together. Letting μ_F be the average value of Recall – Precision for all subjects' second through fifth queries, the following hypothesis test was performed:

H_0: $\mu_F = \mu_A$

H_1: $\mu_F > \mu_A$.

A one-sided t-test assuming unequal variances resulted in a t-value of 2.7292 (98 degrees of freedom) with a p-value equal to 0.0038. The p-value is less than the critical value of 0.05; therefore the null hypothesis is rejected.

In conclusion, when composing their five queries for each search task, subjects tended to place more emphasis on recall than precision with their first query as compared to their subsequent queries.

6.6. QUERY DESCRIPTIONS

The subjects' descriptions of their queries did not reveal any insights into user beliefs or intentions. When writing descriptions about their queries, subjects did not expand very much on their queries, with the exception of a few subjects who wrote lengthier explanations about how one of their queries compared to another. To learn more about how a person composes queries, it would be necessary to know more about the person's domain knowledge with respect to the search task. For example, with regards to the weather/economy search task it would help to know if a subject is familiar with the magnitude of the effects that rising national fuel prices have on transportation costs. However, surveying a subject on various aspects of a search task would not be possible without clueing the subject on possible keywords to use in a query, thus inadvertently biasing the subject towards including a particular set of keywords in their queries. It is concluded that the more one attempts to restrict and define the information need, or desire, of a person the less one learns about how that information need affects

their query intentions. One possible explanation is that desire cannot be so easily controlled.

6.7. QUERY DISTANCE STATISTICS

Subjects returned their surveys after completing them and without being allowed to submit their queries to a search engine. Afterwards, data from the surveys were entered into a spreadsheet. Each query was submitted to the Google search engine and the first three pages of the search results were saved, i.e. the top 30 ranked web pages were saved. According to [92], an internet search engine user views an average of 1.39 search results screens per query, so there is justification in saving only the top 30 ranked web pages for each query. Using the distance measure, distances were calculated between all 165 unique queries from the cell phone task and all 169 queries from the weather/economy task. In the algorithm, the maximum number of chain links was set equal to five. If no connecting chain of five links or less existed between two queries, the distance between them was set equal to the total number of web pages retrieved by all queries. For the cell phone task, 4778 web pages were retrieved. For the weather/economy task, 4538 web pages were retrieved.

For each query q, all other queries were sorted according to their distance from q. All queries that were closest to query q, with the exception of q itself, received a distance rank equal to 1; the second closest queries received a distance rank equal to 2, etc. Average distances were taken across clusters of 10 consecutive distance rankings. For example, for each query, an average distance was calculated for all queries with distance

rankings 1 through 10, an average distance was calculated for all queries with distance rankings 11 through 20, etc., up to distance rankings 91 through 100. The average distance between query q and the queries in cluster k is denoted as $d(q_K)$, for K = 1 to 10. Those 10 clusters of queries and their distances are listed in Table 16 and plotted in Figure 21. The distances are plotted against the maximum ranking in each cluster. Note that if a reference query was not connected by a chain to any other query, then all other queries appeared in cluster 1 for that reference query with distances equal to 4538 or 4778 for the weather/economy and cell phone tasks, respectively. Such cases were not included when calculating $d(q_1)$ averages. There were 20 such cases for the weather/economy search task and 5 such cases for the cell phone search task.

Cluster	**$d(q_K)$ (Weather/Econ Task)**	**$d(q_K)$ (Cell Phone Task)**
1	50.72	47.05
2	65.48	56.06
3	72.41	61.03
4	78.85	89.14
5	84.56	70.41
6	89.30	74.40
7	94.41	139.63
8	297.45	224.93
9	1265.56	1108.23
10	3013.68	1618.34

Table 16. Average distances to ranked query distance clusters.

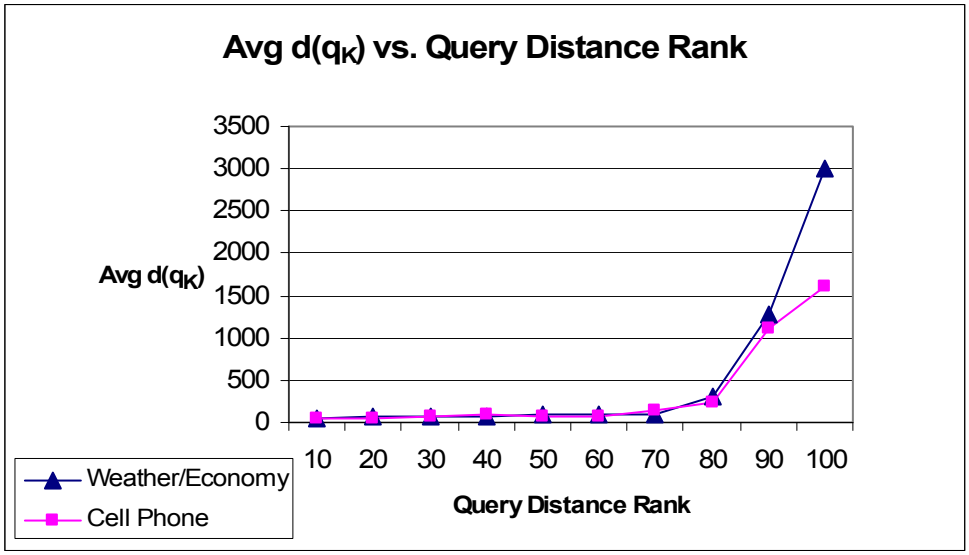


Figure 21. Average distance between a subject query and other queries vs. query distance rank.

Figure 22 indicates a linear trend in clusters 1 through 7, although no statistical tests were performed to test for linearity. The linear trend appears stronger for the weather/economy task than for the cell phone task. The reason behind this is that for the cell phone task there were queries in more distant clusters, e.g. cluster 4, that were theoretically infinitely distant from one or more reference queries. For the weather/economy task, no queries that were clustered in clusters 2 through 7 were infinitely distant from a reference query. In other words, with regards to weather/economy clusters 1 through 7, if a query was infinitely distant from a reference query, it was infinitely distant from all other queries in those clusters.

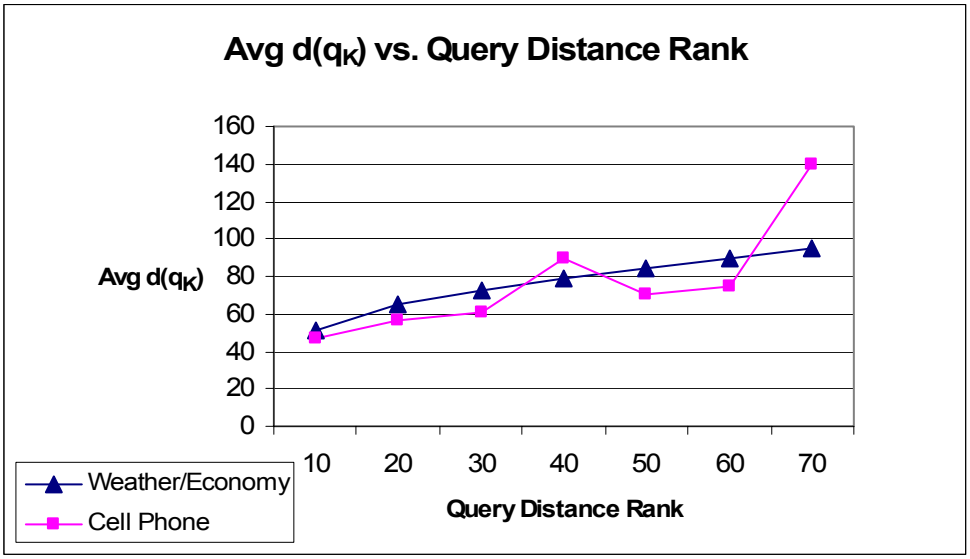


Figure 22. Average distance between a subject query and other queries vs. query distance rank for clusters 1 through 7.

For each query q in a subject's group of 5 queries, an average distance between query q and the subject's other queries was calculated. That distance is denoted as $d(q_S)$. The difference between $d(q_K)$ and $d(q_S)$ is equal to $d(q_K) - d(q_S)$. Averages for $d(q_K) - d(q_S)$ for each task are listed in Table 17 and plotted in Figure 23. In the figure, the points where graphs cross the horizontal axis are the points where, on average with respect to distance for each task, a subject's query is as similar to the subject's other queries as it is to queries in the cluster at that point on the horizontal axis.

Cluster	Avg d(q_K) − d(q_S) (Weather/Econ)	Avg d(q_K) − d(q_S) (Cell Phone)
1	-400.32	-174.70
2	-433.22	-170.37
3	-426.29	-165.40
4	-419.85	-137.28
5	-414.14	-156.94
6	-409.40	-152.95
7	-404.29	-87.72
8	-201.25	-6.18
9	782.44	881.08
10	2544.32	1450.65

Table 17. Cluster averages for d(q_K) − d(q_S) for each search task.

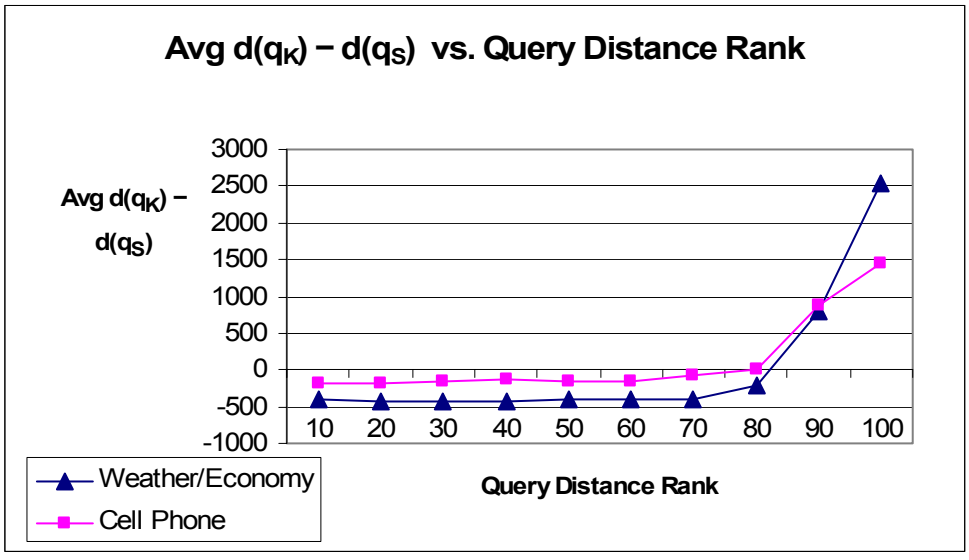


Figure 23. Average d(q_K) − d(q_S) vs. the distance ranking of other queries.

Of particular interest in Figure 23 is the difference between the weather/economy and cell phone graphs. Compared to the weather/economy task, the difference between a

subject's own queries and other subjects' queries is not as great for the cell phone task, i.e. subjects' own queries are clustered closer together for the cell phone task than for the weather/economy task. To test the statistical significance of the data for clusters 1 through 9, let μ_E be the mean $d(q_K) - d(q_S)$ for all weather/economy queries and let μ_C be the mean $d(q_K) - d(q_S)$ for all cell phone queries. The following hypothesis test was performed.

H_0: $\mu_C = \mu_E$

H_1: $\mu_C > \mu_E$

The estimated values of μ_E and μ_C were -267.19 and -27.35, respectively. A one-sided t-test assuming unequal variances resulted in a t-value of -7.3215 (2737 degrees of freedom) with a p-value equal to 1.6×10^{-13}. The p-value is less than the critical value of 0.05; therefore the null hypothesis is rejected.

To test further the statistical significance of the data independently for each cluster 1 through 9, let μ_{Ej} be the mean $d(q_j) - d(q_S)$ for all weather/economy queries and let μ_{Cj} be the mean $d(q_j) - d(q_S)$ for all cell phone queries, where $j = 1$ to 9. The following hypothesis test was performed.

H_0: $\mu_{Cj} = \mu_{Ej}$, for $j = 1$ to 9

H_1: $\mu_{Cj} > \mu_{Ej}$, for $j = 1$ to 9

One-sided t-tests assuming unequal variances resulted in p-values less than 0.05 for all j (> 253 degrees of freedom), except for $j = 9$. For $j = 9$, the resulting t-value was -0.5345 (300 degrees of freedom) with a p-value equal to 0.2967. Therefore, for $j = 1$ to 8, the null hypothesis is rejected, but for $j = 9$, the null hypothesis is not rejected.

The trend illustrated in Figure 23 might be related to different levels of expertise that subjects had on the two search tasks. If so, then a person that has less expertise on an area of interest might submit unintentionally queries that give more varied search results than if the person had more expertise. To look for more evidence of this, $d(q_K) - d(q_S)$ is plotted against each distance cluster by search task background. The data is listed in Table 18. Only data from the cell phone task was plotted, because only one subject claimed to have some background in the weather/economy task. The data is plotted in Figure 24 and the graph is consistent with Figure 23. A statistical test of the data follows Figure 24.

Cluster	**Avg $d(q_K) - d(q_S)$ (No Background)**	**Avg $d(q_K) - d(q_S)$ (Some Background)**
1	-228.02	-138.22
2	-225.00	-132.96
3	-220.36	-127.76
4	-215.17	-83.96
5	-210.60	-119.87
6	-206.89	-115.68
7	-202.81	-8.20
8	-103.69	63.73
9	1050.90	755.86
10	1679.34	1306.04

Table 18. Cluster averages for $d(q_K) - d(q_S)$ for the cell phone search task. Averages divide according to subject search task background.

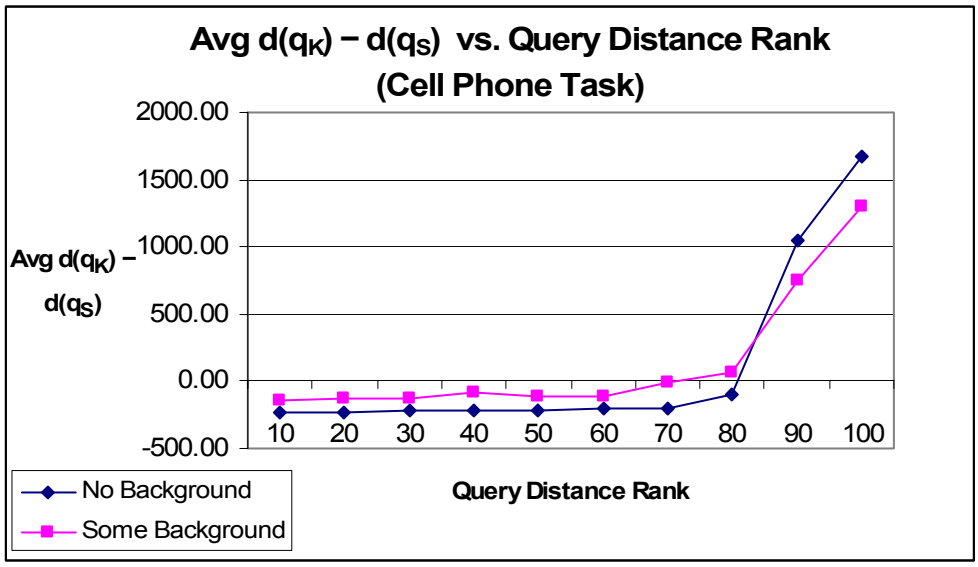


Figure 24. Average d(q_K) − d(q_S) vs. the distance ranking of other queries by background for the cell phone search task.

To test the statistical significance of the data in Figure 24 for clusters 1 through 9, let μ_N be the mean d(q_K) – d(q_S) for all cell phone queries composed by subjects with no task background and let μ_B be the mean d(q_K) – d(q_S) for all cell phone queries composed by subjects with some task background. The following hypothesis test was performed.

H_0: $\mu_B = \mu_N$

H_1: $\mu_B > \mu_N$

The estimated values of μ_N and μ_B were -49.9 and 0.77, respectively. A one-sided t-test assuming unequal variances resulted in a t-value of -1.2333 (1121 degrees of freedom) with a p-value equal to 0.1089. The p-value is greater than the critical value of 0.05; therefore the null hypothesis is not rejected.

To test further the statistical significance of the data in Figure 24 independently for each cluster 1 through 9, let μ_{Nj} be the mean d(q_j) – d(q_S) for all cell phone queries

composed by subjects with no task background and let μ_{Bj} be the mean $d(q_j) - d(q_S)$ for all cell phone queries composed by subjects with some task background, where $j = 1$ to 9. The following hypothesis test was performed.

H_0: $\mu_{Bj} = \mu_{Nj}$, for $j = 1$ to 9

H_1: $\mu_{Bj} > \mu_{Nj}$, for $j = 1$ to 9

One-sided t-tests assuming unequal variances resulted in p-values ranging from 0.01487 for k = 7 to 0.1008 for k = 5, with the p-value for k = 7 being the only p-value less than 0.05 (180 degrees of freedom). Therefore, the null hypothesis is not rejected for $j = 1$ to 6 and for $j = 8$ and 9. For $j = 7$, the null hypothesis is rejected. In conclusion, although Figure 24 does not provide statistically significant evidence that a person with less expertise on an area of interest might submit unintentionally queries that give more varied search results than if the person had more expertise, the evidence is very close to being statistically significant and further study is warranted.

In Figure 23 and Figure 24, both graphs cross the x-axis at approximately the 8th cluster. As pointed out earlier, where the graphs cross the x-axis is the point where, on average with respect to distance, a subject's query is as similar to the subject's other queries as it is to queries in the cluster at that point. Up to that point, Figure 23 and Figure 24 indicate that subjects generated variety in their own queries relative to other queries regardless of task or task background.

As further illustration of the variety in user queries, $d(q_S)$ was plotted for each subject for each task. The plot is in Figure 25. A query pair distance was not used in calculating $d(q_S)$ if the two queries did not have intersecting topics and were not

connected by a chain of topics. There were 68 such query pairs out of 366 total pairs in the weather/economy task and 23 such query pairs out of 377 total pairs in the cell phone task. Average $d(q_S)$ for the weather/economy task was 78.32. Average $d(q_S)$ for the cell phone task was 63.11.

It's unknown what caused the variety in subject queries. A larger study into subjects' desires is necessary to clarify the reasons behind the variety.

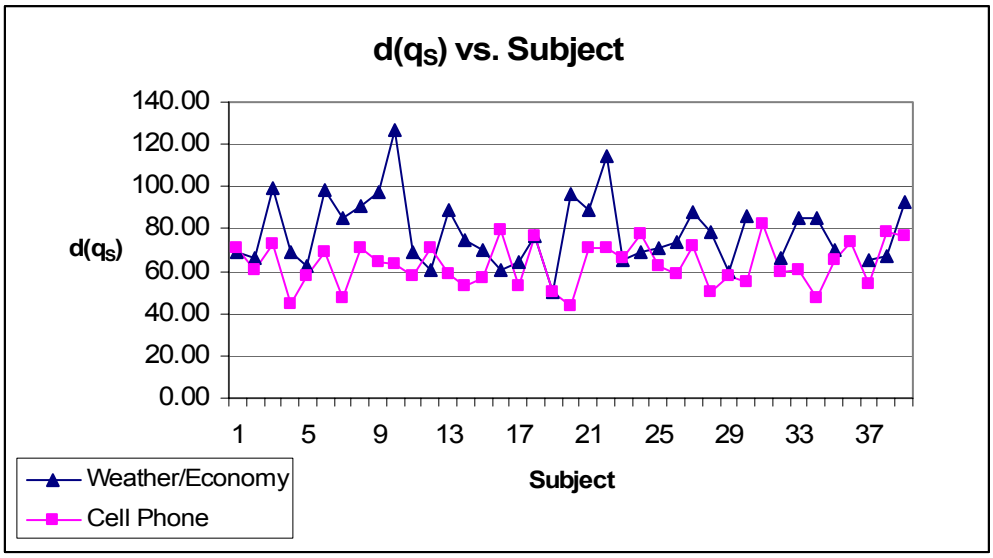


Figure 25. Average distance between a subject's own queries for all subjects. Averages do not include query pair distances for queries that are not connect by a chain of topics.

Shared keywords data was also plotted and the data is listed in Table 19. The average number of shared keywords between a query and each query in cluster k is denoted as $w(q_K)$. Table 19 also includes $d(q_K)$ for each cluster. Inspecting the data in Table 19 reveals an apparently linearly decreasing $w(q_K)$ coinciding with an apparently linearly increasing $d(q_K)$.

	Weather/Economy Task		Cell Phone Task	
Cluster	**w(q_K)**	**d(q_K)**	**w(q_K)**	**d(q_K)**
1	2.12	50.72	1.80	47.05
2	1.89	65.48	1.69	56.06
3	1.73	72.41	1.59	61.03
4	1.63	78.85	1.49	89.14
5	1.53	84.56	1.40	70.41
6	1.44	89.30	1.29	74.40
7	1.22	94.41	1.21	139.63
8	1.07	297.45	1.00	224.93
9	1.08	1265.56	0.95	1108.23
10	1.31	3013.68	0.98	1618.34

Table 19. Average number of shared keywords between a query and the queries in each cluster of ranked queries.

The data from Table 19 is plotted in Figure 26. The graph in Figure 26 shows that as the distance between queries increases, the number of shared keywords between those queries decreases. Towards the tail ends of the plots at around cluster 8, there is a slight increase in the number of shared keywords between queries. According to the distance plots in Figure 23, cluster 8 is about where some of a subject's own queries should appear. If a subject uses a set of keywords repeatedly in their queries, then that would cause the increase in shared keywords at the tail ends of the plots in Figure 26.

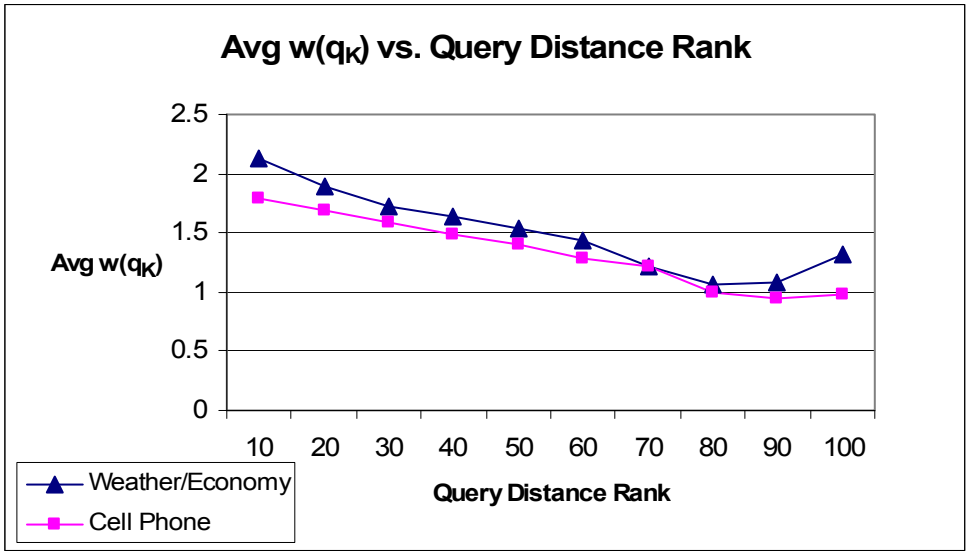


Figure 26. Average number of share keywords between a subject's query and other queries vs. the distance ranking of other queries.

The number of shared keywords between a query and all other queries was next compared to $w(q_K)$, the number of shared keywords between a query and each query in cluster k. The average number of shared keywords between a query and all other queries is denoted as $w(q_S)$. The difference between $w(q_K)$ and $w(q_S)$ is $w(q_K) - w(q_S)$. Data for $w(q_K) - w(q_S)$ is listed in Table 20. The data from Table 20 is plotted in Figure 27.

From the plot in Figure 27 it can be seen that, on average for both tasks, a subject's query shares the same number of keywords with queries in cluster 5 as it does with all other queries. Referring to Figure 26, it can be seen that a query shares an average of approximately 1.5 keywords with other queries in cluster 5.

Cluster	Avg w(q_K) − w(q_S) Weather/Econ Task	Avg w(q_K) − w(q_S) Cell Phone Task
1	0.58	0.38
2	0.35	0.26
3	0.19	0.16
4	0.09	0.06
5	-0.01	-0.03
6	-0.10	-0.15
7	-0.32	-0.22
8	-0.47	-0.43
9	-0.46	-0.49
10	-0.27	-0.47

Table 20. Cluster averages for w(q_K) − w(q_S) for each cluster.

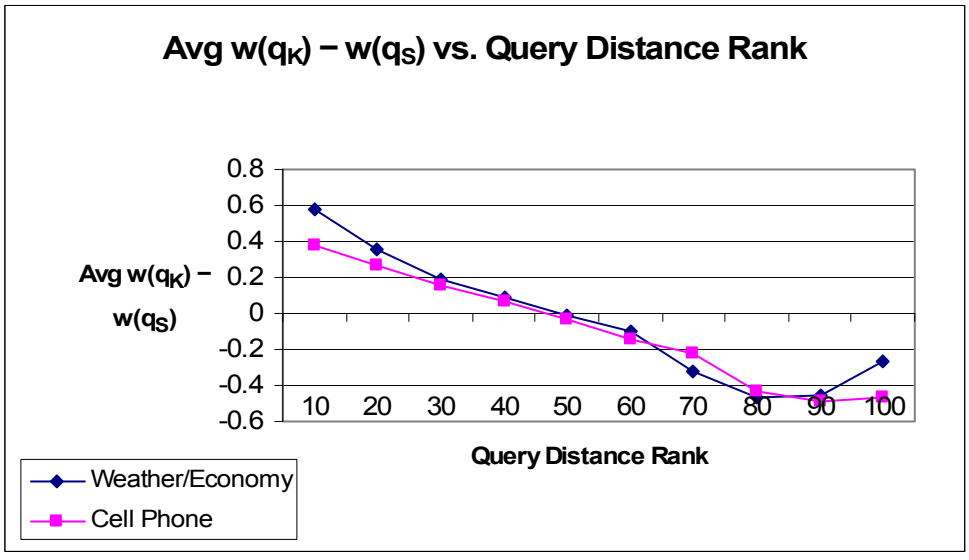


Figure 27. Average number of shared keywords between a subject's query and other cluster queries – the average number of shared keywords between a query and all other own queries vs. query distance rank.

6.8. SUMMARY

For this survey, subjects were assigned a search task, so they were not defining the information need themselves and their motivations did not come from personal interest. Self-defining an information need might have some influence on query formulation and on a subject's articulation of query motives and intentions. However, specifying an information need was necessary in order to obtain a consistent and comparable sample of queries from multiple subjects.

Judging from Figure 12, subjects showed a tendency to reduce the percentage of search task description words in their queries as they composed more queries. However, out of 380 total queries for the two search tasks, 370 queries contained at least one keyword from the search task description. Increasing the number of queries required from each subject, e.g. 10 instead of only 5, might result in subjects using more of their own words in their last few queries and possibly expose a subject's level of knowledge with regards to subtopics related to the search task. Subsequent studies could examine the relationships of a subject's query keywords to the task description as the demands to think of more keywords becomes greater, either after exhausting the use of task description words or after being forbidden from using keywords from the task description. On open question is whether use of terms with high co-occurrence varies relative to use of semantically related terms with topic and/or search engine experience.

From the data represented in Figures 6 through 27, it appears that subjects' beliefs about their information needs were reflected in their recall and precision ratings of their own queries. Beliefs about information needs also appear to have been reflected in the

strategies subjects employed to achieve higher precision for their queries when they did not have knowledge about the search results their queries would retrieve.

The distance measure shows potential for differentiating between search tasks whose relevant information is well known among a population of search engine users and search tasks whose relevant information is not well known among a population of search engine users. The data displayed in Figures 21 through 27 indicate that the distance characteristics and keyword characteristics of user queries, e.g. the number of shared keywords with other queries, may give some indication of the user's level of expertise relative to the general population. The survey results presented in this thesis were obtained with only two specific search tasks and one popular internet search engine. These results can be extended by applying the distance measure to other search tasks and possibly other search engines.

A summary of survey findings is listed in Table 21. For each item in Table 21, inferences are interpretations of the alternative hypothesis defined in the hypothesis test that was performed on the data in the item's figure. Each item's figure is listed in the "Figure" column. Included in some inference descriptions is a p-value, which is the probability of a Type I error, i.e. the probability of rejecting the null hypothesis when the null hypothesis is true. When the null hypothesis was not rejected, no p-value is given.

Item	Item Description	Inference	Figure
1	Avg % of Query in Task Description vs. Subject Query Number	Both tasks: Subjects use a smaller percentage of task words in their query with each successive query. **p-value:** 2.7×10^{-7}	Figure 12
2	Avg Query Keywords vs. Precision Rating	Weather/Econ: Subjects use more keywords to achieve higher precision **p-value:** 2.97×10^{-6} Cell Phone: Subjects do not change the number of query keywords to achieve higher precision	Figure 18
3	Avg % of Query in Task Description vs. Precision Rating	Weather/Econ: Subjects add their own words to task words to achieve higher precision, although those words may be accompanied by other task words. **p-value:** 0.0160 Cell Phone: Subjects substitute either their own words with their own words or task words with other task words to achieve higher precision.	Figure 19
4	Avg Recall – Precision vs. Subject Query Number	Both Tasks: Subjects place much more emphasis on recall than precision with their initial queries. Subjects place only slightly more emphasis on recall with successive queries. **p-value:** 0.0038	Figure 20
5	Avg $d(q_K) - d(q_S)$ vs. Query Distance Rank	Both Tasks: Subjects' queries are clustered closer together for the cell phone task than for the weather/econ task, suggesting an effect due to task expertise. **p-value:** 1.6×10^{-13}	Figure 23

Table 21. A summary of survey findings. Where p-values are given, they represent the probability of a Type I error. Where p-values are not given, the null hypothesis was not rejected.

CHAPTER 7:

CONCLUSIONS AND FUTURE WORK

In this thesis a measure was formulated and investigated for calculating distances between queries that users submit to internet search engines. The goal in developing the distance measure is to increase the efficiency of information searches by enhancing the dialogue nature of interactions between search engines and search engine users.

To better characterize what users bring to this dialogue, the distance measure was developed in the context of Belief, Desire, and Intention Theory. The distance measure is intuitive and relatively simple to understand, so that it might bring search engine users closer to understanding the search results that their queries obtain and improve users' understanding of the information space in which they are navigating.

The dialogue concept also motivated another goal of the distance measure: to help a search engine better understand its population of users and, consequently, better understand any particular user relative to the population. To test the distance measure, a survey was given to look for correlations between user beliefs, desires, intentions, user queries, and the search results of queries as reflected in the calculated distances between those search results. Key findings from the survey include those listed below.

1. Results of the survey indicate that people's beliefs about their information needs are reflected in the strategies they employ to achieve higher precision for their queries. In particular, subjects in this study showed a tendency to use more query keywords to achieve higher precision when they believed their information need to be greater.
2. The distance measure shows potential for differentiating between search tasks about which a population has differing beliefs with regards to their own knowledge of those tasks. To obtain search results for subjects' queries, the queries were submitted to the Google search engine. After applying the proposed distance measure to the search results, clustering queries based on calculated distances resulted in data plots that differed according to search task. The differing data plots are an indication that the distance measure is sensitive to user search task knowledge and to user beliefs about their knowledge.

Subsequent research should include more surveys, both similar and different to the survey conducted for this thesis. One simple extension would be to administer the survey to a new group of subjects and count how many new unique queries are collected. This could give some insight into how many unique queries related to the search tasks are in the total population. It might also provide evidence of query distributions across domains of user knowledge. Increasing the number of queries required from each subject, e.g. 10 instead of only 5, might result in subjects using more of their own words

in their last few queries and possibly expose a subject's level of knowledge with regards to subtopics related to the search task. Another line of inquiry is to examine the cognitive source of a subject's query keywords as the demand to think of more keywords is increased. The level of demand required to inspire greater variety of keywords may be dependent on user characteristics such as background knowledge and search behavior.

Future work should also include applying the distance measure to different search engines. The distance measure might serve as a tool for comparing one search engine's performance to others.

Another use of the distance measure is to create a metric topology on the space of queries and to use that topology to help users visualize and navigate the information space. If the area surrounding a search engine user's current position is defined as those queries that are within some distance from the user's current position, then the search engine could give clues as to how much of that area the user has covered or not covered. The distance measure provides the basis for an information visualization tool and a mathematical technique for finding or forming queries that locate a centroid to a user's queries.

Implementing such a query suggestion tool would require a database of historical user queries and an index of the search results they obtain. The desired fidelity of query distances would determine the number of search results pages for each query to index in the database. An alternative criterion to desired fidelity would be to index only those search results pages that have actually been viewed by past users. A query's search results pages could be automatically updated each time the query is submitted by a user.

Each query could also have an expiration date at which time the query record would be either automatically updated by the search engine or deleted from the database.

Calculating the distances between all queries from the weather/economy task took three days on a Dell Precision 360 personal computer with and Intel Pentium 4 processor. Calculating distances for the cell phone task took seven days. Due to the computational intensity involved with calculating query distances and in absence of a faster algorithm, implementing in real time a query suggestion tool based on the distance measure would require a pre-calculated database of query distances made readily accessible to the search engine. Independent and perhaps distributed computational resources could be dedicated to the task of calculating distances and query clusters.

When implementing a query suggestion tool in real time, a user's search tactic could be monitored and, as implied by survey results, analyzed in comparison with past users that have submitted similar queries to determine the user's probably expertise level and/or information need. The analysis could be performed either by the search engine or by a software agent on the user's own computer that communicates with the search engine to retrieve population statistics. The results of the analysis could narrow down the list of queries that the user might find useful.

One way for a query suggestion tool to be useful would be to guide the user to relevant queries in such a way that the user submits the same or less number of queries during their search session than they would otherwise submit. The second way for a query suggestion tool to be useful would be to guide the user to relevant queries in such a way that the user submits more queries during their search session than they would

otherwise submit, but the query suggestion tool would decrease the cognitive load associated with formulating queries and/or reduce the amount of time it takes to formulate and submit a query to the search engine. Full automation of query refinement is the ultimate end of this line of inquiry.

APPENDIX A:

QUERY SURVEY

This survey will ask you to formulate search engine queries that you would submit to an internet search engine, e.g. Google. The form of this survey is as follows. On each of the next two pages you will see a short description of a search task, for a total of two search tasks. For each task, after reading the description and answering some brief questions on your background, formulate five queries that you would submit to an internet search engine if you were to search the internet for web pages relevant to the search topic. To include a phrase in a query, surround the phrase with quotes ("…"). You are not limited to using keywords that appear in the search task description or in the number of keywords per query.

For example, a search task may be: Find out about possible links between allergic reactions and indoor carpets. Five possible queries are:

1. carpets allergies
2. "indoor carpeting" allergens
3. "hepa filters" pets
4. vacuum cleaners allergy
5. allergy prevention indoor carpets

The survey begins on the next page.

Here is your first **search task** description:

Find out what yearly weather trends do to the performance of the United States' economy.

Before supplying your queries, please answer the following four questions.

1. Have you ever performed an internet search on this topic or similar topics?

Yes No

2. Do you have any background on the topic described in the task description, e.g. personal experience, independent reading, or college coursework?

Yes No

If "Yes", what type of background?

3. On a scale of 1 to 5, with 1 being "very little additional information, I'm already an expert" and 5 being "all possible information, I don't know much", how much information on the given topic would you need to learn in order to become an expert on the topic?

1 2 3 4 5

4. Assume that a "search session" consists of one or more queries on the same topic. Approximately how many search sessions per week do you perform on the internet with a search engine?

0 to 5 6 to 10 Greater than 10

Please list five possible queries related to the search task:

1._____

2._____

3._____

4._____

5._____

Here is your second **search task** description:

Find information on selecting a cell phone plan based on expected cell phone usage.

Before supplying your queries, please answer the following four questions.

1. Have you ever performed an internet search on this topic or similar topics?

Yes No

2. Do you have any background on the topic described in the task description, e.g. personal experience, independent reading, or college coursework?

Yes No

If "Yes", what type of background?

3. On a scale of 1 to 5, with 1 being "very little additional information, I'm an expert" and 5 being "all possible information, I don't know much", how much information on the given topic would you need to learn in order to become an expert on the topic?

1 2 3 4 5

Please list five possible queries related to the search task:

1._____

2._____

3._____

4._____

5._____

As a follow up to listing your queries, for each of your queries please write one sentence, either a statement or a question, describing the meaning of each query. For instance, for the query "allergy prevention indoor carpets" that was given in the carpet/allergy example, one possible sentence description is "How do I eliminate allergens from my indoor carpets to help reduce my allergies?" You are not required to use any keywords in a particular query when writing a sentence description of that query.

Please write your query descriptions/meanings for the **weather/US economy** search topic:

1._____

_____

2._____

_____

3._____

_____

4._____

_____

5._____

_____

Please write your query descriptions/meanings for the **cell phone** search topic:

1._____

_____

2._____

_____

3._____

_____

4._____

_____

5._____

_____

On a scale of 1 to 5, with 1 being "weak" and 5 being "strong", for each of your queries rate the search results you intend to retrieve with your query with respect to
a) retrieving all relevant webpages and
b) retrieving relevant webpages while excluding irrelevant webpages

Please circle your ratings below for your five queries for the **weather/US economy** search topic.

Query 1 ratings:						Query 2 ratings:					
a)	1	2	3	4	5	a)	1	2	3	4	5
b)	1	2	3	4	5	b)	1	2	3	4	5

Query 3 ratings:						Query 4 ratings:					
a)	1	2	3	4	5	a)	1	2	3	4	5
b)	1	2	3	4	5	b)	1	2	3	4	5

Query 5 ratings:					
a)	1	2	3	4	5
b)	1	2	3	4	5

Please circle your ratings below for your five queries for the **cell phone** search topic.

Query 1 ratings:						Query 2 ratings:					
a)	1	2	3	4	5	a)	1	2	3	4	5
b)	1	2	3	4	5	b)	1	2	3	4	5

Query 3 ratings:						Query 4 ratings:					
a)	1	2	3	4	5	a)	1	2	3	4	5
b)	1	2	3	4	5	b)	1	2	3	4	5

Query 5 ratings:					
a)	1	2	3	4	5
b)	1	2	3	4	5

APPENDIX B:

SIMULATION

B.1. SIMULATION PURPOSE

The simulation's purpose is to act as a tool for comparing user search performance with and without query guidance by modeling a user submitting queries to an internet search engine. The simulation is not meant as a substitute for an actual human-search engine experiment or as a validation of the usefulness of these methods in an actual human-search engine environment. Instead, it is meant as a tool for investigating any effects on an internet search that the topology created by the distance measure proposed in this thesis might have.

B.2. SIMULATION DESCRIPTION

B.2.1. OVERVIEW

At the outset of each simulated user search, the user has an information goal. The simulation then generates sequences of queries selected from a database of queries generated by the simulation. Each query in the sequence is selected one at a time and the next query selected in the sequence depends on simulated user characteristics and, if a guided search is being simulated, the calculated distance from the previous submitted

query. If a guided search is not being simulated, each query in the sequence is randomly selected.

Each query in the sequence potentially generates information that will be available to the simulated user. If the search results retrieved by a query do not contain any documents relevant to the user's information need, then the query will not generate any information for the user. The amount of information gleaned from the search results by the simulated user will depend on simulated user characteristics. Before submitting another simulated query, the total amount of information accrued by the user over the sequence of all previous queries is evaluated. If the user's information goal has been achieved, then the user submits no more queries and the search ends. If the user's information goal has not been achieved, then another query is selected and added to the sequence.

B.2.2. MODELING INFORMATION AND LINGUISTIC ABILITIES

It is assumed that a topic can be divided into discrete sub-topics. Therefore, simulated information is bundled into discrete packets and represented as entities. Although this greatly simplifies the representation of information, for purposes here it is not an over-simplification when one considers that information in the real world is bundled into packets that take the form of words, paragraphs, chapters, web pages, etc. In an attempt to approximate associations between packets of information, statistical co-occurrences between information are randomly generated with each hypothetical database that is built during the simulation.

In modeling a user's search behavior on the internet, a user's linguistic abilities, e.g. vocabulary, are not modeled in the simulation. Instead, they are contained in the randomness of the simulation. Without any controlled linguistic abilities to guide the user in the simulation, the only query guidance available to the user is from the proposed methods. The guidance takes the form of suggested queries that result from the information structure supplied by mathematical constructs developed in this thesis.

B.2.3. GENERATING ARTIFICIAL QUERIES AND DOCUMENTS

Before simulating a user submitting queries to an internet search engine, there must exist artificial documents, queries, and query distances. Therefore, a model was built that creates the documents and queries as inputs into the user simulation. Both simulation models are built using the Arena simulation package.

Simulations typically model real world systems or potential real world systems, but generating artificial documents and queries is not really a model of a system. However, because Arena was chosen for modeling a user submitting queries to an internet search engine and because of the probability distributions and other features available in Arena, Arena was the software package of choice for generating artificial documents and queries.

The artificial queries are meant to represent all possible queries that a typical user might submit when researching the artificial user's subject of interest. A user would not necessarily submit all possible queries during a single search session, but would submit some of them until an information need is satisfied or until the user quits the search for

some other reason. Any number of artificial queries can be created in the simulation, although increasing the number of queries increases the computational expense. A total of 50 queries were created for the simulation. A series of array variables in the model represent the queries and each variable array holds information for all the queries in the database, e.g. the size of a query's search results list and the number of relevant documents in a query's search results list. Each query is assigned a unique index into the array variables.

In addition to setting the total number of artificial queries in the simulation, another step was to set the total number of artificial documents, or web pages, to create in the simulation. Each document in the simulation appears in at least one query's search results list. According to [92], an internet search engine user views an average of 1.39 search results screens per query. Since a user views an average of between one and two search results screens per query and search engines commonly display 10 documents per search results screen, a total of (50 queries)*(10 web pages/screen)*(2 screens/query) = 1000 unique artificial documents were created for the simulation. This creates search results lists that average more than 20 documents, because some documents are retrieved by more than one query in the simulation.

Each query was assigned a relevance level and a search results list. For each query, with a probability of 0.20 the query was assigned a relevance level between 0.001 and 0.050 according to a triangular distribution with a mean of 0.010. With a probability of (1 - 0.20) = 0.80, the query was assigned a relevance level equal to 0. Similarly, with a probability of 0.10 each document was assigned a relevance level between 0.001 and

0.050 according to a triangular distribution with a mean of 0.010. With a probability of (1 - 0.10) = 0.90, a document was assigned a relevance level equal to 0.

To create search results lists, each of the 1000 documents was initially assigned to the single query that had the most similar relevance level to the document's relevance level. Irrelevant documents initially were assigned randomly to irrelevant queries. After a document was initially assigned to a query, a document was assigned to another query with a probability based on the total number of queries and the difference between the relevance of the document and the relevance of the query. Therefore, it was possible for any artificial document to appear in the search results lists of two or more artificial queries.

The resultant set of artificial queries and documents was then assigned information. A document was assigned an amount of information dependent on the document's level of relevance. Documents in the same search results list usually, although not always, shared the same particular pieces of information. The whole process of creating artificial queries, documents, and search results lists generated random statistical co-occurrences of information among the documents. Upon completion of a set of artificial queries and their associated search results lists, distances between queries were calculated in the simulation using the distance measure developed in Method 2. A 2-dimensional variable array holds the distances between queries. For experimental purposes, 15 artificial databases were created as inputs to the user model.

B.2.4. THE USER MODEL

Each run of the simulation models 100 users submitting queries from each of the 15 artificial databases that were created as inputs into the user model, i.e. a total of 1500 search sessions, or users, are simulated. Running the model simulates a user that is submitting queries to a search engine. If an unguided search is being performed, then the simulated user selects queries at random. If a guided search is being performed, then the simulated user selects queries based on query distances. If the guided search criteria alone defined the user, then the user would be quite simple. However, at the risk of making the search behavior more complicated, the simulated user was made more interesting by adding some simulated human characteristics to the simulated user.

B.2.4.1. USER CHARACTERISTICS

Listed below are potential characteristics to be included in a user model. Each characteristic is described in terms of one or more key questions about the user.

Characteristics of a user model:

1. **Type of search**. Is the user performing a comprehensive search or are they looking for something specific? E.g. is the user trying to find one particular document or is the user trying to find at least one document from each set in a collection of sets of documents?
2. **Knowledge**. How much does the user know about their subject of interest?

The user's knowledge about their subject of interest could affect their search behavior. For example, a relevant document in a search results list could impact an expert's beliefs about the quality of their query.

3. **Stubbornness**. When the user finds a relevant document, will the user continue searching for other documents? I.e. how many times does the user need to see information before the user believes the information?

   A user might exhibit this characteristic as a way to validate the relevance and/or authenticity of a document by comparing it to other documents, thus convincing the user that the relevant document is indeed relevant and/or contains all needed information.

4. **Conservativeness**. How does the user respond to familiar documents in a search results list and/or to familiar performance, e.g. performance as measured by the relevance of a search results list? Will the user continue to make small changes to query terms or is the user willing to make larger changes when he begins to see familiar documents that were retrieved by previous queries and/or to see familiar performance?

   Conservativeness is similar to Stubbornness, but the difference between the two is that Conservativeness is in regards to actions and intentions whereas Stubbornness is in regards to beliefs.

5. **Thoroughness**. Will the user evaluate each and every document in a search results list? How many pages of a search results list will the user examine? How averse is the user to formulating multiple queries?

Thoroughness is reflected in the probability that a user will find a relevant document in a search results list conditioned on the ranking of the relevant document in the list. Thoroughness might appear to be affected by Knowledge. For example, a knowledgeable user more than a novice user might recognize a relevant document in a search results list from the short description of the document that is presented in the list.

6. **Memory**. How much of a search will the user remember? For example, if a user submits 10 queries over the course of a search, how many of the previous 10 search results will the user remember?

   A search results list might be made more memorable if it contains relevant documents.

7. **Patience**. How likely will the user give up on a search if their initial queries are not fruitful?

B.2.4.2. MODEL PARAMETERS

Parameters were defined in the simulation model to capture some of the above user characteristics. The parameters defined in the simulation are listed below.

1. **Detection Probability**. This parameter is assigned a value in the interval [0, 1] and it is the probability that the user identifies a document in a search results list as being relevant, given that the document is relevant. Detection Probability is intended to model the user's thoroughness and possibly knowledge and patience. For example, familiarity with orthopedics might increase the probability that a

user will recognize the word “orthopedics” in a search results page for the query “achilles tendonitis”. Given that orthopedic surgeons specialize in such injuries, an orthopedic website is likely to be relevant to the query.

2. **Information Fraction**. This parameter is assigned a value in the interval [0, 1] and it defines the amount of accrued information that will end the user’s search. For example, if Information Fraction is assigned the value 0.50, then the user’s search will end when the user accrues 50% or more of the available information. Information Fraction is a simplistic attempt to model the type of search and the user’s stubbornness.
3. **Patience Level**. This parameter is assigned a value greater than or equal to 1 and it equals the maximum number of queries that the user will submit in any given search session. Obviously, Patience Level is intended to model the user’s patience.
4. **Relevant Query Recognition Probability**. This parameter is assigned a value in the interval [0, 1] and it is the probability that the user will recognize a relevant query in a list of queries and submit the query to the search engine. Relevant Query Recognition Probability is intended to model the user’s thoroughness and knowledge.

B.2.4.3. USER SEARCH STEPS

Listed below are the search steps taken by an artificial user in the simulation.

1. For each user, the first query in the user's search is randomly chosen from among the 50 artificial queries. No query is submitted more than once by the same user.
2. When a query is submitted by a user in the model, the query's search results list is retrieved and reviewed by the user.
3. For each relevant document in the search results list, with probability "Detection Probability" the user views the document and accrues any new information contained in the document.
4. If the user's information goal has not been achieved and the user has not run out of patience, the user will submit another query. If the user is performing a random search, then the next query is selected at random. Otherwise, the next query is selected from a list of queries. The list of queries is based on query distances and a relevant query on the list is selected based on the "Relevant Query Recognition Probability".

B.2.4.4. QUERY SELECTION POLICIES

In User Search Step 4, if a guided search is being performed, then the user selects a query from a list of queries. The list of queries does not include all queries that have not been submitted up to that point in the search, but includes non-submitted queries that are closest to the last submitted query. The next query is selected from the list based on some query selection policy. Below are some example query selection policies.

1. Closest query
2. Farthest query

3. Distance of next selected query is conditioned on the relevance of previous search results
4. Random distance policy
5. Alternating distance
6. Periodic randomness

Statistical analysis could compare the different query selection policies based on different performance measures. For example, let's say that out of all the documents retrieved by the 50 queries, there are *K* relevant documents. Comparisons between query selection policies could be based on the following.

1. The average number of queries it takes to retrieve all *K* relevant documents.
2. The average number of queries it takes to retrieve any one of the *K* relevant documents.
3. The average number of queries it takes to retrieve some proportion of the *K* relevant documents.
4. The average number of queries it takes to retrieve either one "magic bullet" document or all documents in some subset of the *K* relevant documents.
5. Given *S* subsets of the *K* relevant documents, the average number of queries it takes to retrieve at least one document from each of the *S* subsets.
6. Assuming the amount of relevant information contained in any document is measured by some number, the average number of queries it takes to accumulate a level of information that is greater than or equal to some threshold value.

B.2.4.5. ALTERNATIVES TO RANDOM SEARCH

An alternative to using random query selection as a baseline for comparison is to use the "best" set of queries as a baseline for comparison. The best set of queries would be the set of queries that best achieves the goal of the search. Below is a list of possible criteria for judging the best set of queries for a search.

1. The minimum number of queries that retrieves all relevant documents.
2. For a fixed number *N*, the *N* queries that retrieve the most number of relevant documents.
3. For a fixed number *N*, the *N* queries that retrieve the set of documents with the most information relevant to the search.

B.2.5. OTHER CONTROL FACTORS

Other possible simulation control factors are listed below.

1. Computational power. The greater the computational power of the search engine, the greater the ability to calculate the true distances between queries.
2. The quality of the initial query.
3. The number of relevant documents, *K*.
4. The distribution of the *K* relevant documents throughout the query system, e.g. all relevant documents in one query, no more than some quantity *D* of relevant documents per query, etc.
5. The relationship between documents from different search results, e.g. degree of similarity, subcategories, supplements, complements, etc.

B.3. VALIDATION AND FUTURE USE

The difficulty with validating the simulation model lies in the difficulty in capturing cognitive processes within users and in capturing semantic meaning within and between bundles of information. People construct linguistic expressions, such as queries, based on their own knowledge of the information need and based on their own inferences. Different people have different levels of knowledge, they make different inferences, and they also exhibit different search behaviors on the internet, e.g. using multiple search engines, following links, using directories, etc. Differences in inference-making abilities between users are the most difficult characteristics to capture and validate, especially with regards to modeling an agent that generates linguistic expressions, e.g. queries, from the agent's own beliefs, desires, and intentions. To capture such abilities would require something akin to artificial intelligence. For example, given a user's linguistic abilities, subject knowledge, intellectual abilities, internet experience, etc., it would be necessary to predict the search engine queries that the user would generate and submit during a search session.

To capture via simulation any benefits from the proposed query distance measure, it would first need to be shown or assumed with justification that there are correlations between linguistic phenomena and the topology generated by the distance measure. In particular, it would need to be shown that the calculated distance between search results corresponds to the semantic distance between the queries that retrieved those search results. Perhaps more appropriately, the semantic distance between two queries could be

described as the semantic distance between the meanings or intentions behind those queries. An algorithm would then need to be developed based on the distance measure for suggesting queries to search engine users. Finally, it would be necessary to develop a query suggestion tool and test it on human subjects to study how people would use such a tool. The simulation could then be run to show the possible benefits of a guided search over an unguided search.

Despite the difficulties associated with modeling search engine users with simulation, using simulation to improve search engines is still worth pursuing. For instance, instead of modeling a user, a search engine developer could model query submission patterns, such as with the number of queries users submit during a search session, the number of search results pages that users view, or the degree of changes in queries that users make between successive queries in an attempt to find relevant information during their search session. Such patterns would vary from user to user due to not only individual search habits, but perhaps also to the type of search a person is doing. For example, studies might indicate that people in general demonstrate more or less significant changes between successive queries if they are searching for product prices, e.g. computer printer prices, as opposed to if they are searching for something like a journal article. Since internet search engines rely heavily on marketing fees for revenue, a search engine developer might wish to investigate such patterns and how those patterns might be affected by the implementation of a query suggestion tool before actually implementing the query suggestion tool.

APPENDIX C:

SURVEY ANALYSIS TASKS

A number of labor intensive tasks were required to obtain statistics from the survey data. For some tasks, computer programming code was written to perform the task. In the list of tasks below, the notation "-VB(*Procedure*)" indicates that a Visual Basic procedure by the name *Procedure* performs the task. The code for all Visual Basic procedures is given at the end of the task list.

Survey Analysis Task List:

1. Enter survey data into Excel

   An Excel workbook was created for each search task. Queries were entered by row into a worksheet and each query was entered exactly as it was written in the survey by its author.

2. Parse queries -VB(ParseQueries)

   Parsing a query only included splitting the query into its component keywords and extracting any phrases in quotation marks from the query. Although phrases in quotation marks were identified and extracted from queries, quotation marks were ignored when extracting all individual keywords and when counting the number of keywords in queries. For example, for the query *"cell phone" plans*, the phrase "cell phone" was extracted and output to its own column in the "Queries" spreadsheet. However, the three keywords "cell" "phone", and "plans" were each extracted from the query and each word was output to its own cell in the "Queries" spreadsheet.

3. Parse query descriptions -VB(ParseDescriptions)

   Parsing a query description only included splitting the description into its component keywords and extracting any phrases in quotation marks from the description.

4. Count query words in the task description -VB(QueryWordsInTask)

   For each keyword in each query, the search task description was searched to find the keyword. If a keyword was found in the task description, the query-keywords-in-the-task-description count was incremented by one. If a keyword appeared twice in the query and appeared only once in the task description, then the count was incremented by two.

5. Count query words in the query description -VB(QueryWordsInDesc)

   Similar to Task 4 above.

6. Count unique shared keywords between queries -VB(QuerySharedWords)

7. Calculate keyword data averages -VB(GetWordAverages)

   For each subject, averages were calculated on the data collected in Steps 2, 3, 4, 5, and 6.

8. Assign a unique number to each unique query -VB(AssignQueryNumbers)

   A total of 188 queries were collected for the weather/economy search task and 192 queries for the cell phone search task. 169 of the 188 weather/economy queries were unique and 165 of the 192 cell phone queries were unique. Each unique query was assigned a unique number from 1 to 169 and 1 to 165 for the weather/economy task and the cell phone task, respectively.

9. Submit queries to the Google search engine

   Each unique query was submitted manually to the Google search engine. The form of each submitted query was exactly as the subject had written it. Quotation marks were not ignored when counting the number of unique queries. Hence, a query was submitted with quotation marks to the search engine if the subject included quotation marks in their query.

   The source codes of the first three search results pages, i.e. the top 30 ranked web pages, were saved as text files. In cases where less than 21 web pages were retrieved, the pages of source code saved was only one or two, depending on the number of web pages retrieved.

10. Import search results into the Excel workbook -VB(GetSearchResults)

    Within each query's search results source code file or files, the URLs were found for each web page in the search results list and the URLs were entered into the Excel workbook.

11. Import search results into the Arena simulation and calculate distances

    For each search task, the search results for all queries were saved to a single text file and that text file was imported into the Arena simulation. The Arena simulation then calculated query distances using the distance measure and the imported search results.

12. Import query distances into the Excel workbook -VB(GetDistancesFromFile)

13. Find distance statistics for each subject's own queries -VB(SubjectDistanceStats)

    For each subject, the average distance between the subject's own queries was calculated. The minimum and maximum distances between any two of the subject's own queries were also found.

14. Sort queries by distance -VB(SortDistances)

    For each query, all other queries were sorted by distance in ascending order. Distances were those calculated by the distance measure.

15. Find average distances to subject's own queries -VB(SubjectAvgQueryDistances)

    For each query, the average distance between it and all the subject's other queries was calculated.

16. Find cluster statistics -VB(GetClosestQueriesStats)

    For each query, cluster averages were calculated for such properties as distance between the query and a query in the cluster, number of shared keywords between the query and a query in the cluster, etc.

17. Statistical formulas used for survey analysis.

    When performing regression analysis, the least squares fit method was used. In the least squares fit method, β_1 is estimated with the following [71]:

$$\hat{\beta}_1 = \frac{\operatorname{cov}(x, y)}{\sigma_x^2} = \frac{\mathrm{ss}_{xy}}{\mathrm{ss}_{xx}}$$

    where

$$\mathrm{ss}_{xx} = \sum_{i=1}^{n} (x_i - \bar{x})^2$$

$$\mathrm{ss}_{xy} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

    n = the number of data points

    $\bar{x}$ = the sample mean of the independent variable X

    $\bar{y}$ = the sample mean of the independent variable Y.

When performing two sample tests of means assuming unequal variances, the following formula was used to calculate t-values [4]:

$$t = \frac{\bar{x} - \bar{y} - \delta_0}{\sqrt{\frac{s_1^2}{m} + \frac{+ s_2^2}{n}}},$$

where

$\bar{x}$ = the mean of sample from population 1

$\bar{y}$ = mean of sample from population 2

δ_0 = hypothesized difference in population means

s_1^2 = variance of sample from population 1

s_2^2 = variance of sample from population 2

m = size of sample from population 1

n = size of sample from population 2

Code for each Visual Basic procedure is given next.

Task 2: ParseQueries Procedure

```
Private Sub OKButton_Click()

Dim sQuery As String
Dim sWord As String
Dim phraseChr As String
Dim wordStart As Integer
Dim wordEnd As Integer
Dim qIndex As Integer
Dim wIndex As Integer
Dim queryPosition As Integer
Dim endOfQuery As Boolean
Dim endOfPhrase As Boolean
Dim queryEndChr As String
Dim querySheet As String
Dim sPhrase As String
Dim sPhraseWord As String
Dim phrasePosition As Integer
Dim phraseWordStart As Integer
Dim phraseWordEnd As Integer
Const nHeaderRows As Integer = 3
Const phraseColumn As Integer = 7

queryEndChr = "@"
phraseChr = "%"

querySheet = "Queries"

For qIndex = 1 To Val(QNumber.Text)
    endOfQuery = False
    wIndex = 1
    queryPosition = 1
    sQuery = Worksheets(querySheet).Cells(qIndex + nHeaderRows, 1).Text
    Do While endOfQuery = False
        wordStart = InStr(queryPosition, sQuery, phraseChr)
        If wordStart = queryPosition Then
            wordStart = queryPosition + 1
            wordEnd = InStr(wordStart, sQuery, phraseChr)
            sWord = Mid(sQuery, wordStart, wordEnd - wordStart)
            If Application.WorksheetFunction.CountBlank(Cells(qIndex _
             + nHeaderRows, phraseColumn)) > 0 Then
                Worksheets(querySheet).Cells(qIndex + _
                nHeaderRows, phraseColumn).Value = sWord
            Else
                Worksheets(querySheet).Cells(qIndex + _
                nHeaderRows, phraseColumn).Value = _
                Worksheets(querySheet).Cells(qIndex + nHeaderRows, _
                phraseColumn).Value & "," & sWord
            End If
            endOfPhrase = False
            sPhrase = sWord
            phrasePosition = 1
            Do While endOfPhrase = False
```

```
                phraseWordStart = phrasePosition
                phraseWordEnd = InStr(phraseWordStart, sPhrase, " ")
                If phraseWordEnd = 0 Then
                    phraseWordEnd = Len(sPhrase) + 1
                    endOfPhrase = True
                End If
                sPhraseWord = Mid(sPhrase, phraseWordStart, _
                  phraseWordEnd - phraseWordStart)
                phrasePosition = phraseWordEnd + 1
                Worksheets(querySheet).Cells(qIndex + nHeaderRows, _
                  phraseColumn + wIndex).Value = sPhraseWord
                wIndex = wIndex + 1
            Loop

            queryPosition = InStr(wordEnd, sQuery, " ")
            If queryPosition = 0 Then
                endOfQuery = True
            Else
                queryPosition = queryPosition + 1
            End If

        Else
            wordStart = queryPosition
            wordEnd = InStr(wordStart, sQuery, " ")
            If wordEnd = 0 Then
                wordEnd = Len(sQuery) + 1
                endOfQuery = True
            End If

            sWord = Mid(sQuery, wordStart, wordEnd - wordStart)
            queryPosition = wordEnd + 1
            Worksheets(querySheet).Cells(qIndex + nHeaderRows, _
                     phraseColumn + wIndex).Value = sWord
            wIndex = wIndex + 1
        End If
    Loop
    If endOfQuery = True Then
        Worksheets(querySheet).Cells(qIndex + nHeaderRows, _
               phraseColumn + wIndex).Value = queryEndChr
    End If
Next qIndex

End Sub
```

Task 3:ParseDescriptions Procedure

```
Private Sub OKButton_Click()

Dim sDesc As String
Dim sWord As String
Dim phraseChr As String
Dim wordStart As Integer
Dim wordEnd As Integer
Dim dIndex As Integer
Dim wIndex As Integer
Dim descPosition As Integer
Dim endOfDesc As Boolean
Dim endOfPhrase As Boolean
Dim descSheet As String
Dim sPhrase As String
Dim sPhraseWord As String
Dim phrasePosition As Integer
Dim phraseWordStart As Integer
Dim phraseWordEnd As Integer
Dim descEndChr As String
Const nHeaderRows As Integer = 3
Const phraseColumn As Integer = 8

descEndChr = "@"
phraseChr = "%"
descSheet = "Descriptions"

For dIndex = 1 To Val(DNumber.Text)
    endOfDesc = False
    wIndex = 1
    descPosition = 1
    sDesc = Worksheets(descSheet).Cells(dIndex + nHeaderRows, 2).Text
    Do While endOfDesc = False
        wordStart = InStr(descPosition, sDesc, phraseChr)
        If wordStart = descPosition Then
            wordStart = descPosition + 1
            wordEnd = InStr(wordStart, sDesc, phraseChr)
            sWord = Mid(sDesc, wordStart, wordEnd - wordStart)
            If Application.WorksheetFunction.CountBlank(Cells(dIndex _
                   + nHeaderRows, phraseColumn)) > 0 Then
                Worksheets(descSheet).Cells(dIndex + nHeaderRows, _
                   phraseColumn).Value = sWord
            Else
                Worksheets(descSheet).Cells(dIndex + nHeaderRows, _
                   phraseColumn).Value = _
                        Worksheets(descSheet).Cells(dIndex + _
                             nHeaderRows, phraseColumn).Value & "," & _
                                  sWord
            End If
            endOfPhrase = False
            sPhrase = sWord
            phrasePosition = 1
```

```
            Do While endOfPhrase = False
                phraseWordStart = phrasePosition
                phraseWordEnd = InStr(phraseWordStart, sPhrase, " ")
                If phraseWordEnd = 0 Then
                    phraseWordEnd = Len(sPhrase) + 1
                    endOfPhrase = True
                End If
                sPhraseWord = Mid(sPhrase, phraseWordStart, _
                   phraseWordEnd - phraseWordStart)
                phrasePosition = phraseWordEnd + 1
                Worksheets(descSheet).Cells(dIndex + nHeaderRows, _
                   phraseColumn + wIndex).Value = sPhraseWord
                wIndex = wIndex + 1
            Loop

            descPosition = InStr(wordEnd, sDesc, " ")
            If descPosition = 0 Then
                endOfDesc = True
            Else
                descPosition = descPosition + 1
            End If

        Else
            wordStart = descPosition
            wordEnd = InStr(wordStart, sDesc, " ")
            If wordEnd = 0 Then
                wordEnd = Len(sDesc) + 1
                endOfDesc = True
            End If

            sWord = Mid(sDesc, wordStart, wordEnd - wordStart)
            descPosition = wordEnd + 1
            Worksheets(descSheet).Cells(dIndex + nHeaderRows, _
              phraseColumn + wIndex).Value = sWord
            wIndex = wIndex + 1
        End If
    Loop

    If endOfDesc = True Then
        Worksheets(descSheet).Cells(dIndex + nHeaderRows, _
              phraseColumn + wIndex).Value = descEndChr
    End If

Next dIndex

End Sub
```

Task 4:QueryWordsInTask Procedure

```
Private Sub OKButton_Click()

Dim queryIndex As Integer
Dim queryWordIndex As Integer
Dim weightCheckIndex As Integer
Dim taskWordIndex As Integer
Dim queryTaskWords As Integer
Dim queryWordCount As Integer
Dim totalWordCount As Integer
Dim queryWord As String
Dim taskWord As String
Dim weightCheckWord As String
Dim wordWeight As Integer
Dim querySheet As String
Dim wordSheet As String
Dim keywordColumns As Integer
Dim outputCol As Integer
Dim queryEndChr As String
Dim taskEndChr As String
Dim doubleWeight As Boolean
Dim endWeightCheck As Boolean
Dim wordMatchFound As Boolean
Dim endOfQuery As Boolean
Dim endOfTask As Boolean
Dim totalTaskWords As Integer
Const queryHeaderRows As Integer = 3
Const wordHeaderRows As Integer = 1
Const taskRow As Integer = 1
Const taskCol As Integer = 1
Const qSubjectCol As Integer = 3
Const qQueryCol As Integer = 4
Const phraseCol As Integer = 7
Const weightCheckRow = 22
Const wordWordCol As Integer = 9
Const wordInTaskCol As Integer = 10
Const wSubjectCol As Integer = 11
Const wQueryCol As Integer = 12

totalWordCount = 0
queryEndChr = "@"
taskEndChr = "@"

querySheet = "Queries"
wordSheet = "WordData"

If CellOptionButton.Value = True Then
    keywordColumns = 10
    totalTaskWords = 14
Else
    keywordColumns = 9
    totalTaskWords = 15
```

```
End If

outputCol = phraseCol + keywordColumns + 3

For queryIndex = 1 To Val(QueriesText.Text)
    queryWordIndex = 1
    queryWord = Worksheets(querySheet).Cells(queryHeaderRows + _
         queryIndex, phraseCol + queryWordIndex).Text
    queryWordCount = 0
    totalWordCount = totalWordCount + 1
    queryTaskWords = 0
    endOfQuery = False

    Do While endOfQuery = False
        weightCheckIndex = weightCheckRow
        endWeightCheck = False
        weightCheckWord = Worksheets(wordSheet). _
              Cells(weightCheckIndex, 1).Value
        doubleWeight = False
        If EconOptionButton.Value = True Then
            Do While endWeightCheck = False
                If StrComp(queryWord, weightCheckWord, vbTextCompare) _
                        = 0 Then
                    doubleWeight = True
                    endWeightCheck = True
                Else
                    weightCheckIndex = weightCheckIndex + 1
                    weightCheckWord = _
                        Worksheets(wordSheet).Cells(weightCheckIndex, _
                             1).Value
                    If StrComp(weightCheckWord, taskEndChr, _
                    vbTextCompare) = 0 Then
                        endWeightCheck = True
                    End If
                End If
            Loop
        End If
        If doubleWeight = True Then
            wordWeight = 2
        Else
            wordWeight = 1
        End If
        queryWordCount = queryWordCount + wordWeight
        Worksheets(wordSheet).Cells(wordHeaderRows + totalWordCount, _
              wordWordCol).Value = queryWord
        taskWordIndex = 1
        wordMatchFound = False
        endOfTask = False
        Do While endOfTask = False
            taskWord = Worksheets(wordSheet).Cells(wordHeaderRows + _
                   taskWordIndex, 1).Text
            If StrComp(taskWord, queryWord, vbTextCompare) = 0 Then
                wordMatchFound = True
                endOfTask = True
```

```
            Else
                taskWordIndex = taskWordIndex + 1
                taskWord = Worksheets(wordSheet).Cells(wordHeaderRows _
                   + taskWordIndex, 1).Text
                If StrComp(taskWord, taskEndChr, vbTextCompare) _
                   = 0 Then
                    endOfTask = True
                End If
            End If
        Loop
        If wordMatchFound = True Then
            queryTaskWords = queryTaskWords + wordWeight
            Worksheets(wordSheet).Cells(wordHeaderRows + _
                   totalWordCount, wordInTaskCol).Value = 1
        Else
            Worksheets(wordSheet).Cells(wordHeaderRows + _
                   totalWordCount, wordInTaskCol).Value = 0
        End If
        Worksheets(wordSheet).Cells(wordHeaderRows + _
              totalWordCount, wSubjectCol).Value = _
              Worksheets(querySheet).Cells(queryHeaderRows + _
                   queryIndex, qSubjectCol).Value
        Worksheets(wordSheet).Cells(wordHeaderRows + totalWordCount, _
              wQueryCol).Value = _
              Worksheets(querySheet).Cells(queryHeaderRows + _
                        queryIndex, qQueryCol).Value
        queryWordIndex = queryWordIndex + 1
        queryWord = Worksheets(querySheet).Cells(queryHeaderRows + _
              queryIndex, phraseCol + queryWordIndex).Text
        If StrComp(queryWord, queryEndChr, vbTextCompare) = 0 Then
            endOfQuery = True
        Else
            totalWordCount = totalWordCount + 1
        End If
    Loop
    Worksheets(querySheet).Cells(queryHeaderRows + queryIndex, _
          outputCol - 1).Value = queryWordCount
    Worksheets(querySheet).Cells(queryHeaderRows + queryIndex, _
          outputCol).Value = queryTaskWords
    Worksheets(querySheet).Cells(queryHeaderRows + queryIndex, _
          outputCol + 1).Value = queryTaskWords / queryWordCount * 100
    Worksheets(querySheet).Cells(queryHeaderRows + queryIndex, _
          outputCol + 2).Value = queryTaskWords / totalTaskWords * 100
Next queryIndex

End Sub
```

Task 5: QueryWordsInDesc Procedure

```
Private Sub OKButton_Click()

Dim kIndex As Integer
Dim qIndex As Integer
Dim wCount As Integer
Dim totalWordCount As Integer
Dim dIndex As Integer
Dim kPrevIndex As Integer
Dim queryWord As String
Dim descWord As String
Dim prevWord As String
Dim querySheet As String
Dim descSheet As String
Dim wordSheet As String
Dim outputCol As Integer
Dim wordWeight As Integer
Dim wordSubject As Integer
Dim wordQuery As Integer
Dim queryAppearances As Integer
Dim descAppearances As Integer
Dim percentOfDesc As Single
Dim percentOfQuery As Single
Dim thisDescWords As Integer
Dim thisQueryWords As Integer
Dim nextCell As String
Dim phraseChr As String
Dim queryEndChr As String
Dim descEndChr As String
Dim wordInDesc As Boolean
Dim endOfQuery As Boolean
Dim endOfQueryScan As Boolean
Dim endOfDesc As Boolean
Dim uniqueCheck As Boolean
Dim uniqueWord As Boolean
Dim descWordsCol As Integer
Dim queryWordsCol As Integer
Const queryPhraseCol As Integer = 7
Const descHeaderRows As Integer = 3
Const descQueryCol As Integer = 1
Const descSubjectCol As Integer = 3
Const descUserQueryCol As Integer = 4
Const descPhraseCol As Integer = 8
Const wordHeaderRows As Integer = 1
Const wordWordCol As Integer = 13
Const wordInQueryCol As Integer = 14
Const wordSubjectCol As Integer = 15
Const wordQueryCol As Integer = 16

phraseChr = "%"
queryEndChr = "@"
descEndChr = "@"
```

```
totalWordCount = wordHeaderRows

querySheet = "Queries"
descSheet = "Descriptions"
wordSheet = "WordData"

If CellOptionButton.Value = True Then
    queryWordsCol = 19
    descWordsCol = 38
    outputCol = 39
Else
    queryWordsCol = 18
    descWordsCol = 50
    outputCol = 51
End If

For qIndex = 1 To Val(QueriesText.Text)
    wCount = 0
    descAppearances = 0
    kIndex = 1
    queryWord = Worksheets(querySheet).Cells(descHeaderRows + qIndex, _
         queryPhraseCol + kIndex).Value
    queryAppearances = 0
    endOfQuery = False
    Do While endOfQuery = False
        uniqueWord = True
        If kIndex >= 2 Then
            kPrevIndex = 1
            uniqueCheck = False
            Do While uniqueCheck = False
                prevWord = Worksheets(querySheet). _
                   Cells(descHeaderRows + qIndex, queryPhraseCol + _
                        kPrevIndex).Value
                If StrComp(queryWord, prevWord, vbTextCompare) = 0 Then
                    uniqueCheck = True
                    uniqueWord = False
                ElseIf kPrevIndex < kIndex - 1 Then
                    kPrevIndex = kPrevIndex + 1
                Else
                    uniqueCheck = True
                End If
            Loop
        End If
        If uniqueWord = True Then
            totalWordCount = totalWordCount + 1
            dIndex = 1
            endOfDesc = False
            wordInDesc = False
            descWord = Worksheets(descSheet).Cells(descHeaderRows + _
                   qIndex, descPhraseCol + dIndex).Value
            Do While endOfDesc = False
                If StrComp(queryWord, descWord, vbTextCompare) = 0 Then
                    descAppearances = descAppearances + 1
                    wordInDesc = True
```

```
            End If
            dIndex = dIndex + 1
            descWord = Worksheets(descSheet).Cells(descHeaderRows _
                    + qIndex, descPhraseCol + dIndex).Value
            If StrComp(descWord, descEndChr, vbTextCompare) _
            = 0 Then
                endOfDesc = True
            End If
        Loop
        If wordInDesc = True Then
            wCount = wCount + 1
            Worksheets(wordSheet).Cells(totalWordCount, _
               wordInQueryCol).Value = 1
            dIndex = 1
            endOfQueryScan = False
            descWord = Worksheets(querySheet). _
               Cells(descHeaderRows + qIndex, queryPhraseCol + _
               dIndex).Value
            Do While endOfQueryScan = False
                If StrComp(queryWord, descWord, vbTextCompare) _
                = 0 Then
                    queryAppearances = queryAppearances + 1
                End If
                dIndex = dIndex + 1
                descWord = _
                    Worksheets(querySheet).Cells(descHeaderRows + _
                         qIndex, queryPhraseCol + dIndex).Value
                If StrComp(descWord, descEndChr, vbTextCompare) _
                = 0 Then
                    endOfQueryScan = True
                End If
            Loop
        Else
            Worksheets(wordSheet).Cells(totalWordCount, _
               wordInQueryCol).Value = 0
        End If
        Worksheets(wordSheet).Cells(totalWordCount, _
               wordWordCol).Value = queryWord
        wordSubject = Worksheets(descSheet).Cells(descHeaderRows _
               + qIndex, descSubjectCol).Value
        Worksheets(wordSheet).Cells(totalWordCount, _
               wordSubjectCol).Value = wordSubject
        wordQuery = Worksheets(descSheet).Cells(descHeaderRows + _
               qIndex, descUserQueryCol).Value
        Worksheets(wordSheet).Cells(totalWordCount, _
               wordQueryCol).Value = wordQuery
    End If
    kIndex = kIndex + 1
    queryWord = Worksheets(querySheet).Cells(descHeaderRows + _
          qIndex, queryPhraseCol + kIndex).Value
    If StrComp(queryWord, queryEndChr, vbTextCompare) = 0 Then
        endOfQuery = True
    End If
Loop
```

```
    Worksheets(descSheet).Cells(descHeaderRows + qIndex, _
         outputCol).Value = wCount
    thisDescWords = Worksheets(descSheet).Cells(descHeaderRows + _
         qIndex, descWordsCol).Value
    percentOfDesc = descAppearances / thisDescWords * 100
    Worksheets(descSheet).Cells(descHeaderRows + qIndex, outputCol + _
         1).Value = percentOfDesc
    thisQueryWords = Worksheets(querySheet).Cells(descHeaderRows + _
         qIndex, queryWordsCol).Value
    percentOfQuery = queryAppearances / thisQueryWords * 100
    Worksheets(descSheet).Cells(descHeaderRows + qIndex, outputCol + _
         2).Value = percentOfQuery
Next qIndex

End Sub
```

Task 6: QuerySharedWords Procedure

```
Private Sub OKButton_Click()

Dim numberOfQueries As Integer
Dim queryIndex1 As Integer
Dim queryIndex2 As Integer
Dim wordIndex1 As Integer
Dim wordIndex2 As Integer
Dim queryWord1 As String
Dim queryWord2 As String
Dim queryEndChr As String
Dim sharedWordsCount As Integer

Dim wordMatchFound As Boolean
Dim endOfQuery1 As Boolean
Dim endOfQuery2 As Boolean
Dim lastQuery As Boolean
Dim lastQueryPair As Boolean
Dim nextWord1 As Boolean
Dim uniqueCheck As Boolean

Dim querySheet As String
Dim sharedWordsSheet As String

Dim rowIndex As Integer
Dim columnIndex As Integer

Const queryHeaderRows As Integer = 3
Const sharedHeaderRows As Integer = 3
Const sharedLeaderCols As Integer = 3
Const phraseCol As Integer = 7

numberOfQueries = Val(QueriesText.Text)
sharedWordCount = 0
queryEndChr = "@"
taskEndChr = "@"

sharedWordsSheet = "QuerySharedWords"
querySheet = "Queries"

queryIndex1 = 1
lastQuery = False
Do While lastQuery = False
    queryIndex2 = queryIndex1 + 1
    lastQueryPair = False
    Do While lastQueryPair = False
        sharedWordsCount = 0
        wordIndex1 = 1
        rowIndex = queryHeaderRows + queryIndex1
        columnIndex = phraseCol + wordIndex1
        queryWord1 = Worksheets(querySheet).Cells(rowIndex, _
              columnIndex).Text
```

```
endOfQuery1 = False
Do While endOfQuery1 = False
    wordIndex2 = 1
    rowIndex = queryHeaderRows + queryIndex2
    columnIndex = phraseCol + wordIndex2
    queryWord2 = Worksheets(querySheet).Cells(rowIndex, _
           columnIndex).Text
    endOfQuery2 = False
    Do While endOfQuery2 = False
        If StrComp(queryWord1, queryWord2) = 0 Then
            sharedWordsCount = sharedWordsCount + 1
            endOfQuery2 = True
        Else
            wordIndex2 = wordIndex2 + 1
            columnIndex = phraseCol + wordIndex2
            queryWord2 = Worksheets(querySheet). _
                Cells(rowIndex, columnIndex).Text
            If StrComp(queryWord2, queryEndChr) = 0 Then
                endOfQuery2 = True
            End If
        End If
    Loop
    rowIndex = queryHeaderRows + queryIndex1
    wordIndex1 = wordIndex1 + 1
    columnIndex = phraseCol + wordIndex1
    queryWord1 = Worksheets(querySheet).Cells(rowIndex, _
           columnIndex).Text
    nextWord1 = False
    Do While nextWord1 = False
        If StrComp(queryWord1, queryEndChr) = 0 Then
            rowIndex = sharedHeaderRows + queryIndex1
            columnIndex = sharedLeaderCols + queryIndex2
            Worksheets(sharedWordsSheet).Cells(rowIndex, _
                columnIndex).Value = sharedWordsCount
            Worksheets(sharedWordsSheet).Cells(columnIndex, _
                rowIndex).Value = sharedWordsCount
            nextWord1 = True
            endOfQuery1 = True
        Else
            wordIndex2 = 1
            columnIndex = phraseCol + wordIndex2
            queryWord2 = Worksheets(querySheet). _
                Cells(rowIndex, columnIndex).Text
            uniqueCheck = False
            Do While uniqueCheck = False
                If StrComp(queryWord1, queryWord2) = 0 Then
                    uniqueCheck = True
                Else
                    wordIndex2 = wordIndex2 + 1
                    columnIndex = phraseCol + wordIndex2
                    queryWord2 = _
                        Worksheets(querySheet).Cells(rowIndex, _
                          columnIndex).Text
                End If
```

```
                    Loop
                    If wordIndex1 = wordIndex2 Then
                        nextWord1 = True
                    Else
                        wordIndex1 = wordIndex1 + 1
                        columnIndex = phraseCol + wordIndex1
                        queryWord1 = Worksheets(querySheet). _
                            Cells(rowIndex, columnIndex).Text
                    End If
                End If
            Loop
        Loop
        queryIndex2 = queryIndex2 + 1
        If queryIndex2 > numberOfQueries Then
            lastQueryPair = True
        End If
    Loop
    rowIndex = sharedHeaderRows + queryIndex1
    columnIndex = sharedLeaderCols + queryIndex1
    Worksheets(sharedWordsSheet).Cells(rowIndex, columnIndex).Value _
         = wordIndex1 - 1
    queryIndex1 = queryIndex1 + 1
    If queryIndex1 = numberOfQueries Then
        lastQuery = True
    End If
Loop

End Sub
```

Task 7: GetWordAverages Procedure

```
Private Sub OKButton_Click()

Dim keywordColQS As Integer
Dim queryWordsInTaskColQS As Integer
Dim percentofQueryColQS As Integer
Dim percentofTaskColQS As Integer
Dim descWordColDS As Integer
Dim queryWordsInDescColDS As Integer
Dim percentofQueryColDS As Integer
Dim percentofDescColDS As Integer
Dim inputDataCols(9) As Integer
Dim avgDataCols(9) As Integer

Dim distanceStatsSheet As String
Dim inputDataSheet As String
Dim inputDataSheets(2) As String
Dim totalSubjects As Integer
Dim subjectIndex As Integer
Dim subjectFirstRow As Integer
Dim queryIndex As Integer
Dim statIndex As Integer
Dim dataIndex As Integer
Dim dataValue As Single
Dim dataSum As Single
Dim dataAvg As Single
Dim nextQuery As Integer
Dim lastSubjectQuery As Boolean
Dim lastSubject As Boolean
Dim lastAvg As Boolean

Const numberOfAvgs As Integer = 10
Const avgsFromQuerySheet = 6
Const headerRowsQS As Integer = 3
Const headerRowsDDS As Integer = 1
Const subjectColQS As Integer = 3
Const queryNoColQS As Integer = 4
Const recallColQS As Integer = 5
Const precisionColQS As Integer = 6
Const avgRecallColDDS As Integer = 13
Const avgPrecisionColDDS As Integer = 14
Const avgQueryWordsColDDS As Integer = 15
Const avgQueryWordsInTaskColDDS As Integer = 16
Const avgPercentQWordsInTaskColDDS As Integer = 17
Const avgQWordsPercentofTaskColDDS As Integer = 18
Const avgDescWordsColDDS As Integer = 19
Const avgQWordsInDescColDDS As Integer = 20
Const avgQWordsPercentofDescColDDS As Integer = 21
Const avgPercentQWordsInDescColDDS As Integer = 22

inputDataCols(0) = recallColQS
inputDataCols(1) = precisionColQS
```

```
avgDataCols(0) = avgRecallColDDS
avgDataCols(1) = avgPrecisionColDDS
avgDataCols(2) = avgQueryWordsColDDS
avgDataCols(3) = avgQueryWordsInTaskColDDS
avgDataCols(4) = avgPercentQWordsInTaskColDDS
avgDataCols(5) = avgQWordsPercentofTaskColDDS
avgDataCols(6) = avgDescWordsColDDS
avgDataCols(7) = avgQWordsInDescColDDS
avgDataCols(8) = avgQWordsPercentofDescColDDS
avgDataCols(9) = avgPercentQWordsInDescColDDS

inputDataSheets(0) = "Queries"
inputDataSheets(1) = "Descriptions"
distanceStatsSheet = "DistanceStats"

If CellOptionButton.Value = True Then
    keywordColQS = 19
    queryWordsInTaskColQS = 20
    percentofQueryColQS = 21
    percentofTaskColQS = 22
    descWordColDS = 38
    queryWordsInDescColDS = 39
    percentofDescColDS = 40
    percentofQueryColDS = 41
    inputDataCols(2) = keywordColQS
    inputDataCols(3) = queryWordsInTaskColQS
    inputDataCols(4) = percentofQueryColQS
    inputDataCols(5) = percentofTaskColQS
    inputDataCols(6) = descWordColDS
    inputDataCols(7) = queryWordsInDescColDS
    inputDataCols(8) = percentofDescColDS
    inputDataCols(9) = percentofQueryColDS
Else
    keywordColQS = 18
    queryWordsInTaskColQS = 19
    percentofQueryColQS = 20
    percentofTaskColQS = 21
    descWordColDS = 50
    queryWordsInDescColDS = 51
    percentofDescColDS = 52
    percentofQueryColDS = 53
    inputDataCols(2) = keywordColQS
    inputDataCols(3) = queryWordsInTaskColQS
    inputDataCols(4) = percentofQueryColQS
    inputDataCols(5) = percentofTaskColQS
    inputDataCols(6) = descWordColDS
    inputDataCols(7) = queryWordsInDescColDS
    inputDataCols(8) = percentofDescColDS
    inputDataCols(9) = percentofQueryColDS
End If

totalSubjects = Val(SNumber.Text)
subjectIndex = 1
subjectFirstRow = headerRowsQS + 1
```

```
lastSubject = False
Do While lastSubject = False
    queryIndex = 1
    lastSubjectQuery = False
    Do While lastSubjectQuery = False
        nextQuery = Worksheets(inputDataSheets(0)). _
             Cells(subjectFirstRow + queryIndex, queryNoColQS).Value
        If nextQuery > queryIndex Then
            queryIndex = queryIndex + 1
        Else
            lastSubjectQuery = True
        End If
    Loop
    statIndex = 1
    lastAvg = False
    Do While lastAvg = False
        If statIndex <= avgsFromQuerySheet Then
            inputDataSheet = inputDataSheets(0)
        Else
            inputDataSheet = inputDataSheets(1)
        End If
        dataSum = 0
        dataIndex = 1
        Do While dataIndex <= queryIndex
            dataValue = Worksheets(inputDataSheet). _
              Cells(subjectFirstRow + dataIndex - 1, _
                   inputDataCols(statIndex - 1)).Value
            dataSum = dataSum + dataValue
            dataIndex = dataIndex + 1
        Loop
        dataAvg = dataSum / queryIndex
        Worksheets(distanceStatsSheet).Cells(headerRowsDDS + _
              subjectIndex, avgDataCols(statIndex - 1)).Value = dataAvg
        statIndex = statIndex + 1
        If statIndex > numberOfAvgs Then
            lastAvg = True
        End If
    Loop
    If subjectIndex < totalSubjects Then
        subjectIndex = subjectIndex + 1
        subjectFirstRow = subjectFirstRow + queryIndex
    Else
        lastSubject = True
    End If
Loop

End Sub
```

Task 8: AssignQueryNumbers Procedure

```
Private Sub OKButton_Click()

Dim querySheet As String
Dim uniqueQuerySheet As String
Dim distanceStatsSheet As String
Dim thisQuery As String
Dim thatQuery As String
Dim thisQueryIndex As Integer
Dim thatQueryIndex As Integer
Dim totalQueries As Integer
Dim queryMatch As Boolean
Dim lastQuery As Boolean
Const headerRowsQS = 3
Const queryRowUQS = 3
Const leaderColsUQS = 1
Const headerRowsDDS = 1
Const leaderColsDDS = 1
Const queryColDDS = 2

querySheet = "Queries"
uniqueQuerySheet = "SearchResults1"
distanceStatsSheet = "DistanceStats"

totalQueries = Val(QNumber.Text)
thisQueryIndex = 1
lastQuery = False
Do While lastQuery = False
    thisQuery = Worksheets(querySheet).Cells(headerRowsQS + _
         thisQueryIndex, 1).Value
    thatQueryIndex = 1
    queryMatch = False
    Do While queryMatch = False
        thatQuery = Worksheets(uniqueQuerySheet).Cells(queryRowUQS, _
              leaderColsUQS + thatQueryIndex).Value
        If StrComp(thisQuery, thatQuery, vbTextCompare) = 0 Then
            Worksheets(distanceStatsSheet).Cells(headerRowsDDS + _
                   thisQueryIndex, queryColDDS).Value = thatQueryIndex
            queryMatch = True
        Else
            thatQueryIndex = thatQueryIndex + 1
        End If
    Loop
    thisQueryIndex = thisQueryIndex + 1
    If thisQueryIndex > totalQueries Then
        lastQuery = True
    End If
Loop

End Sub
```

Task 10: GetSearchResults Procedure

```
Private Sub OKButton_Click()

Dim sTemp As String
Dim sLength As String
Dim sRight As String
Dim url As String
Dim htmlCode As String
Dim startFound As Integer
Dim startFound1 As Integer
Dim startFound2 As Integer
Dim httpFound As Integer
Dim urlLength As Integer
Dim nextFound As Integer
Dim rankCount As Integer
Dim nextSiteFound As Integer
Dim rankMaxStart As Integer
Dim rankMaxEnd As Integer
Dim rankMaxText As String
Dim rankMax As Integer
Dim urlRank As Integer
Dim qIndex As Integer
Dim htmlSheetRow As Integer
Const nHeaderRows As Integer = 3
Const nLeaderColumns As Integer = 1

htmlSheetRow = nHeaderRows + 1

For qIndex = 1 To Val(QNumber.Text)

    If CellOptionButton.Value = True Then
        Open "C:\Documents and Settings\craig.LB233G\My _
          Documents\GradSchool\Dissertation\SearchResultsData\ _
          CellPhoneQueries\SR" & qIndex & "1.txt" For Input As #1
    Else
        Open "C:\Documents and Settings\craig.LB233G\My _
          Documents\GradSchool\Dissertation\SearchResultsData\ _
          EconomyQueries\SR" & qIndex & "1.txt" For Input As #1
    End If

    htmlCode = Input(LOF(1), 1)
     'Read source code for 1st search results page.

    rankMaxEnd = InStr(htmlCode, "* of about")
    If rankMaxEnd = 0 Then
        Close #1
        GoTo Line100
        'Search Results list empty; No websites returned with query
    End If
    rankMaxStart = InStr(rankMaxEnd - 4, htmlCode, "*") + 1
    rankMaxText = Mid(htmlCode, rankMaxStart, rankMaxEnd _
          - rankMaxStart)
    rankMax = Val(rankMaxText)
```

```
startFound1 = InStrRev(htmlCode, "</url?sa")
'Find last sponsored link.
startFound2 = InStrRev(htmlCode, "</pagead")
'Find last sponsored link.

If startFound1 < startFound2 Then
    startFound = startFound2
Else
    startFound = startFound1
End If

If startFound = 0 Then
    startFound = InStr(htmlCode, "seconds)")
End If

Seek #1, startFound

Do While rankCount < rankMax
    nextSiteFound = 1
    Do While nextSiteFound > 0
        Line Input #1, sTemp
        nextSiteFound = Len(sTemp)
    Loop
    Do While httpFound = 0
        Line Input #1, sTemp
        httpFound = InStr(sTemp, "<http")
    Loop
    urlLength = InStr(httpFound, sTemp, ">") - httpFound
    url = Mid(sTemp, httpFound + 1, urlLength - 1)
    rankCount = rankCount + 1
    urlRank = urlRank + 1

    Worksheets("SRNew1").Cells(htmlSheetRow, 1).Value = url
    Worksheets("SRNew1").Cells(htmlSheetRow, qIndex + _
          nLeaderColumns).Value = urlRank
    Worksheets("SRNew2").Cells(htmlSheetRow, 1).Value = url
    Worksheets("SRNew2").Cells(htmlSheetRow, qIndex + _
          nLeaderColumns).Value = 1
    Worksheets("TopicData").Cells(htmlSheetRow - nHeaderRows, _
          1).Value = url
    Worksheets("TopicData").Cells(htmlSheetRow - nHeaderRows, _
          2).Value = qIndex
    Worksheets("TopicData").Cells(htmlSheetRow - nHeaderRows, _
          3).Value = urlRank

    htmlSheetRow = htmlSheetRow + 1
    httpFound = 0

Loop

Close #1

nextFound = InStrRev(htmlCode, "20&sa=N>")
```

```
     'Check for more search results
If nextFound = 0 Then GoTo Line100

startFound = 0

If CellOptionButton.Value = True Then
    Open "C:\Documents and Settings\craig.LB233G\My _
          Documents\GradSchool\Dissertation\SearchResultsData\ _
          CellPhoneQueries\SR" & qIndex & "2.txt" For Input As #2
Else
    Open "C:\Documents and Settings\craig.LB233G\My _
          Documents\GradSchool\Dissertation\SearchResultsData\ _
          EconomyQueries\SR" & qIndex & "2.txt" For Input As #2
End If

htmlCode = Input(LOF(2), 2)
'Read source code for 2nd search results page.

rankMaxEnd = InStr(htmlCode, "* of about")
rankMaxStart = InStr(rankMaxEnd - 4, htmlCode, "*") + 1
rankMaxText = Mid(htmlCode, rankMaxStart, rankMaxEnd _
              - rankMaxStart)
rankMax = Val(rankMaxText)

startFound1 = InStrRev(htmlCode, "</url?sa")
startFound2 = InStrRev(htmlCode, "</pagead")
     'Find last sponsored link.

If startFound1 < startFound2 Then
    startFound = startFound2
Else
    startFound = startFound1
End If

If startFound = 0 Then
    startFound = InStr(htmlCode, "seconds)")
End If

Seek #2, startFound

Do While rankCount < rankMax
    nextSiteFound = 1
    Do While nextSiteFound > 0
        Line Input #2, sTemp
        nextSiteFound = Len(sTemp)
    Loop
    Do While httpFound = 0
        Line Input #2, sTemp
        httpFound = InStr(sTemp, "<http")
    Loop
    urlLength = InStr(httpFound, sTemp, ">") - httpFound
    url = Mid(sTemp, httpFound + 1, urlLength - 1)
    rankCount = rankCount + 1
    urlRank = urlRank + 1
```

```
    Worksheets("SRNew1").Cells(htmlSheetRow, 1).Value = url
    Worksheets("SRNew1").Cells(htmlSheetRow, qIndex + _
          nLeaderColumns).Value = urlRank
    Worksheets("SRNew2").Cells(htmlSheetRow, 1).Value = url
    Worksheets("SRNew2").Cells(htmlSheetRow, qIndex + _
          nLeaderColumns).Value = 1
    Worksheets("TopicData").Cells(htmlSheetRow - nHeaderRows, _
          1).Value = url
    Worksheets("TopicData").Cells(htmlSheetRow - nHeaderRows, _
          2).Value = qIndex
    Worksheets("TopicData").Cells(htmlSheetRow - nHeaderRows, _
          3).Value = urlRank

    htmlSheetRow = htmlSheetRow + 1
    httpFound = 0
Loop

Close #2

nextFound = InStrRev(htmlCode, "30&sa=N>")
     'Check for more search results
If nextFound = 0 Then GoTo Line100

startFound = 0

If CellOptionButton.Value = True Then
    Open "C:\Documents and Settings\craig.LB233G\My _
     Documents\GradSchool\Dissertation\SearchResultsData\ _
     CellPhoneQueries\SR" & qIndex & "3.txt" For Input As #3
Else
    Open "C:\Documents and Settings\craig.LB233G\My _
     Documents\GradSchool\Dissertation\SearchResultsData\ _
     EconomyQueries\SR" & qIndex & "3.txt" For Input As #3
End If

htmlCode = Input(LOF(3), 3)

rankMaxEnd = InStr(htmlCode, "* of about")
rankMaxStart = InStr(rankMaxEnd - 4, htmlCode, "*") + 1
rankMaxText = Mid(htmlCode, rankMaxStart, rankMaxEnd _
              - rankMaxStart)
rankMax = Val(rankMaxText)

startFound1 = InStrRev(htmlCode, "</url?sa")
startFound2 = InStrRev(htmlCode, "</pagead")
 'Find last sponsored link.

If startFound1 < startFound2 Then
    startFound = startFound2
Else
    startFound = startFound1
End If
```

```
    If startFound = 0 Then
        startFound = InStr(htmlCode, "seconds)")
    End If

    Seek #3, startFound

    Do While rankCount < rankMax
        nextSiteFound = 1
        Do While nextSiteFound > 0
            Line Input #3, sTemp
            nextSiteFound = Len(sTemp)
        Loop
        Do While httpFound = 0
            Line Input #3, sTemp
            httpFound = InStr(sTemp, "<http")
        Loop
        urlLength = InStr(httpFound, sTemp, ">") - httpFound
        url = Mid(sTemp, httpFound + 1, urlLength - 1)
        rankCount = rankCount + 1
        urlRank = urlRank + 1

        Worksheets("SRNew1").Cells(htmlSheetRow, 1).Value = url
        Worksheets("SRNew1").Cells(htmlSheetRow, qIndex + _
              nLeaderColumns).Value = urlRank
        Worksheets("SRNew2").Cells(htmlSheetRow, 1).Value = url
        Worksheets("SRNew2").Cells(htmlSheetRow, qIndex + _
              nLeaderColumns).Value = 1
        Worksheets("TopicData").Cells(htmlSheetRow - nHeaderRows, _
              1).Value = url
        Worksheets("TopicData").Cells(htmlSheetRow - nHeaderRows, _
              2).Value = qIndex
        Worksheets("TopicData").Cells(htmlSheetRow - nHeaderRows, _
              3).Value = urlRank

        htmlSheetRow = htmlSheetRow + 1
        httpFound = 0

    Loop

    Close #3

Line100:
    startFound = 0
    rankCount = 0
    urlRank = 0

Next qIndex

End Sub
```

Task 12: GetDistancesFromFile Procedure

```
Private Sub OKButton_Click()

Dim qRow As Integer
Dim qColumn As Integer
Dim queryDistance As Integer
Dim queryDistancesSum As Double
Dim finiteDistancesSum As Double
Dim queryPairs As Integer
Dim finiteDistanceQueryPairs As Integer
Dim avgDistances As Double
Dim avgFiniteDistances As Double
Dim sTemp As String
Const nLeaderColumns As Integer = 3
Const nHeaderRows As Integer = 3

If CellOptionButton.Value = True Then
    Open "C:\Documents and Settings\craig.LB233G\My _
     Documents\GradSchool\Dissertation\Simulation\CellSearchResults\ _
     calculated distances " & QNumber.Text & ".txt" For Input As #1
Else
    Open "C:\Documents and Settings\craig.LB233G\My _
     Documents\GradSchool\Dissertation\Simulation\EconSearchResults\ _
     calculated distances " & QNumber.Text & ".txt" For Input As #1
End If

For qRow = 1 To Val(QNumber.Text)
    For qColumn = 1 To Val(QNumber.Text)
        Line Input #1, sTemp
        queryDistance = Val(sTemp)
        Worksheets("Distances").Cells(qRow + nHeaderRows, qColumn + _
              nLeaderColumns).Value = queryDistance
        If qColumn < qRow Then
            queryDistancesSum = queryDistancesSum + queryDistance
            queryPairs = queryPairs + 1
            If queryDistance < Val(DNumber.Text) Then
                finiteDistancesSum = finiteDistancesSum + queryDistance
                finiteDistanceQueryPairs = finiteDistanceQueryPairs + 1
            End If
        End If
    Next qColumn
Next qRow

Close #1
avgDistances = queryDistancesSum / queryPairs
avgFiniteDistances = finiteDistancesSum / finiteDistanceQueryPairs
Worksheets("Distances").Cells(nHeaderRows + 1, nLeaderColumns + _
     Val(QNumber.Text) + 1).Value = avgDistances
Worksheets("Distances").Cells(nHeaderRows + 1, nLeaderColumns + _
     Val(QNumber.Text) + 2).Value = avgFiniteDistances

End Sub
```

Task 13: SubjectDistanceStats Procedure

```
Private Sub OKButton_Click()

Dim totalSubjects As Integer
Dim subjectIndex As Integer
Dim nextSubject As Integer
Dim subjectFirstRow As Integer
Dim thisQueryIndex As Integer
Dim thatQueryIndex As Integer
Dim firstQueryRow As Integer
Dim firstQueryIndex As Integer
Dim secondQueryIndex As Integer
Dim queryDistance As Integer
Dim minDistance As Integer
Dim maxDistance As Integer
Dim distanceSum As Long
Dim totalDistances As Integer
Dim avgDistance As Single
Dim totalQueries As Integer
Dim lastDistanceThisQuery As Boolean
Dim lastQuery As Boolean
Dim lastSubject As Boolean
Const headerRows As Integer = 1
Const subjectCol As Integer = 1
Const queryCol As Integer = 2
Const headerRowsDS As Integer = 3
Const leaderColsDS As Integer = 3
Const minDistanceCol As Integer = 8
Const maxDistanceCol As Integer = 9
Const avgDistanceCol As Integer = 10

distanceSheet = "Distances"
distanceStatsSheet = "DistanceStats"
totalSubjects = Val(SNumber.Text)
subjectIndex = 1
subjectFirstRow = headerRows + 1
lastSubject = False
Do While lastSubject = False
    minDistance = 10000
    maxDistance = 0
    distanceSum = 0
    avgDistance = 0
    nextSubject = Worksheets(distanceStatsSheet). _
          Cells(subjectFirstRow + 1, subjectCol).Value
    If nextSubject > subjectIndex Then
        Worksheets(distanceStatsSheet).Cells(subjectFirstRow, _
              queryCol + 1).Value = 0
        lastQuery = True
    Else
        firstQueryRow = subjectFirstRow
        firstQueryIndex = 1
        totalDistances = 0
        lastQuery = False
```

```
    End If
    Do While lastQuery = False
        thisQueryIndex = Worksheets(distanceStatsSheet). _
              Cells(firstQueryRow, queryCol).Value
        secondQueryIndex = 1
        lastDistanceThisQuery = False
        Do While lastDistanceThisQuery = False
            thatQueryIndex = Worksheets(distanceStatsSheet). _
              Cells(firstQueryRow + secondQueryIndex, queryCol).Value
            queryDistance = Worksheets(distanceSheet). _
              Cells(headerRowsDS + thisQueryIndex, leaderColsDS + _
              thatQueryIndex).Value
            Worksheets(distanceStatsSheet).Cells(firstQueryRow, _
              queryCol + firstQueryIndex + secondQueryIndex - 1). _
                   Value = queryDistance
            distanceSum = distanceSum + queryDistance
            totalDistances = totalDistances + 1
            If queryDistance < minDistance Then
                minDistance = queryDistance
            End If
            If queryDistance > maxDistance Then
                maxDistance = queryDistance
            End If
            secondQueryIndex = secondQueryIndex + 1
            nextSubject = Worksheets(distanceStatsSheet). _
              Cells(firstQueryRow + secondQueryIndex, subjectCol).Value
            If nextSubject > subjectIndex Then
                lastDistanceThisQuery = True
            End If
        Loop
        If secondQueryIndex = 2 Then
            avgDistance = distanceSum / totalDistances
            lastQuery = True
        Else
            firstQueryIndex = firstQueryIndex + 1
            firstQueryRow = firstQueryRow + 1
        End If
    Loop
    Worksheets(distanceStatsSheet).Cells(headerRows + subjectIndex, _
         minDistanceCol).Value = minDistance
    Worksheets(distanceStatsSheet).Cells(headerRows + subjectIndex, _
         maxDistanceCol).Value = maxDistance
    Worksheets(distanceStatsSheet).Cells(headerRows + subjectIndex, _
         avgDistanceCol).Value = avgDistance
    subjectIndex = subjectIndex + 1
    If subjectIndex <= totalSubjects Then
        subjectFirstRow = firstQueryRow + 2
    Else
        lastSubject = True
    End If
Loop

End Sub
```

Task 14: SortDistances Procedure

```
Private Sub OKButton_Click()

Dim queryDistances() As Integer
Dim queryNumbers() As Integer
Dim querySubjects() As Integer
Dim queryWords() As Integer
Dim querySharedWords() As Integer
Dim distanceTemp As Integer
Dim queryTemp As Integer
Dim subjectTemp As Integer
Dim wordsTemp As Integer
Dim sharedWordsTemp As Integer
Dim placeHolder As Integer
Dim queryIndex1 As Integer
Dim queryIndex2 As Integer
Dim distanceIndex As Integer
Dim queryIndex As Integer
Dim columnIndex As Integer
Dim rowIndex As Integer
Dim numberOfQueries

Dim lastQuery As Boolean
Dim firstQuery As Boolean
Dim sortComplete As Boolean
Dim lastQuerySorted As Boolean
Dim smallestQuery As Boolean

Dim distanceSheet As String
Dim sortedSheet As String
Dim querySheet As String
Dim sharedWordsSheet As String

Dim queryHeaderRowQSD As Integer
Dim subjectHeaderRowQSD As Integer
Dim wordsHeaderRowQSD As Integer
Dim sharedWordsHeaderRowQSD As Integer

Const headerRowsCDA As Integer = 3
Const headerRowsQSD As Integer = 4
Const headerRowsQS As Integer = 3
Const headerRowsQSW As Integer = 3
Const leaderColsCDA As Integer = 3
Const leaderColsQSD As Integer = 1
Const wordColumnQS As Integer = 19
Const leaderColsQSW As Integer = 3
Const queryColumnCDA As Integer = 1
Const subjectColumnCDA As Integer = 2

numberOfQueries = Val(QNumber.Text)

sortedSheet = "QueriesSortedByDistance"
sharedWordsSheet = "QuerySharedWords"
```

```
queryHeaderRowQSD = headerRowsQSD + numberOfQueries + 5
subjectHeaderRowQSD = queryHeaderRowQSD + numberOfQueries + 5
wordsHeaderRowQSD = subjectHeaderRowQSD + numberOfQueries + 5
sharedWordsHeaderRowQSD = wordsHeaderRowQSD + numberOfQueries + 5

distanceSheet = "DistancesAll"
querySheet = "Queries"

ReDim queryDistances(1 To numberOfQueries)
ReDim queryNumbers(1 To numberOfQueries)
ReDim querySubjects(1 To numberOfQueries)
ReDim queryWords(1 To numberOfQueries)
ReDim querySharedWords(1 To numberOfQueries)

queryIndex = 1
lastQuerySorted = False
Do While lastQuerySorted = False
    distanceIndex = 1
    lastQuery = False
    Do While lastQuery = False
        rowIndex = headerRowsCDA + distanceIndex
        columnIndex = leaderColsCDA + queryIndex
        queryDistances(distanceIndex) = Worksheets(distanceSheet). _
              Cells(rowIndex, columnIndex).Value
        columnIndex = queryColumnCDA
        queryNumbers(distanceIndex) = Worksheets(distanceSheet). _
              Cells(rowIndex, columnIndex).Value
        columnIndex = subjectColumnCDA
        querySubjects(distanceIndex) = Worksheets(distanceSheet). _
              Cells(rowIndex, columnIndex).Value
        rowIndex = headerRowsQS + distanceIndex
        columnIndex = wordColumnQS
        queryWords(distanceIndex) = Worksheets(querySheet). _
              Cells(rowIndex, columnIndex).Value
        rowIndex = headerRowsQSW + distanceIndex
        columnIndex = leaderColsQSW + queryIndex
        querySharedWords(distanceIndex) = _
              Worksheets(sharedWordsSheet).Cells(rowIndex, _
              columnIndex).Value
        If distanceIndex < numberOfQueries Then
            distanceIndex = distanceIndex + 1
        Else
            lastQuery = True
        End If
    Loop

    queryIndex1 = 1
    queryIndex2 = 2
    placeHolder = queryIndex2
    sortComplete = False
    Do While sortComplete = False
        If queryDistances(queryIndex1) _
              > queryDistances(queryIndex2) Then
```

```
placeHolder = queryIndex2
distanceTemp = queryDistances(queryIndex1)
queryTemp = queryNumbers(queryIndex1)
subjectTemp = querySubjects(queryIndex1)
wordsTemp = queryWords(queryIndex1)
sharedWordsTemp = querySharedWords(queryIndex1)
queryDistances(queryIndex1) = queryDistances(queryIndex2)
queryDistances(queryIndex2) = distanceTemp
queryNumbers(queryIndex1) = queryNumbers(queryIndex2)
queryNumbers(queryIndex2) = queryTemp
querySubjects(queryIndex1) = querySubjects(queryIndex2)
querySubjects(queryIndex2) = subjectTemp
queryWords(queryIndex1) = queryWords(queryIndex2)
queryWords(queryIndex2) = wordsTemp
querySharedWords(queryIndex1) = _
       querySharedWords(queryIndex2)
querySharedWords(queryIndex2) = sharedWordsTemp
If queryIndex1 > 1 Then
    queryIndex2 = queryIndex1
    queryIndex1 = queryIndex2 - 1
    firstQuery = False
Else
    firstQuery = True
End If
Do While firstQuery = False
    If queryDistances(queryIndex1) _
       > queryDistances(queryIndex2) Then
        distanceTemp = queryDistances(queryIndex1)
        queryTemp = queryNumbers(queryIndex1)
        subjectTemp = querySubjects(queryIndex1)
        wordsTemp = queryWords(queryIndex1)
        sharedWordsTemp = querySharedWords(queryIndex1)
        queryDistances(queryIndex1) = _
            queryDistances(queryIndex2)
        queryDistances(queryIndex2) = distanceTemp
        queryNumbers(queryIndex1) = _
            queryNumbers(queryIndex2)
        queryNumbers(queryIndex2) = queryTemp
        querySubjects(queryIndex1) = _
            querySubjects(queryIndex2)
        querySubjects(queryIndex2) = subjectTemp
        queryWords(queryIndex1) = queryWords(queryIndex2)
        queryWords(queryIndex2) = wordsTemp
        querySharedWords(queryIndex1) = _
            querySharedWords(queryIndex2)
        querySharedWords(queryIndex2) = sharedWordsTemp
        If queryIndex1 > 1 Then
            queryIndex2 = queryIndex1
            queryIndex1 = queryIndex2 - 1
        Else
            firstQuery = True
        End If
    ElseIf queryDistances(queryIndex1) = _
       queryDistances(queryIndex2) Then
```

```
                smallestQuery = False
                Do While smallestQuery = False
                    If (queryNumbers(queryIndex1) > _
                         queryNumbers(queryIndex2) And _
                         queryDistances(queryIndex1) = _
                              queryDistances(queryIndex2)) Then
                        queryTemp = queryNumbers(queryIndex1)
                        subjectTemp = querySubjects(queryIndex1)
                        wordsTemp = queryWords(queryIndex1)
                        sharedWordsTemp = _
                              querySharedWords(queryIndex1)
                        queryNumbers(queryIndex1) = _
                              queryNumbers(queryIndex2)
                        queryNumbers(queryIndex2) = queryTemp
                        querySubjects(queryIndex1) = _
                              querySubjects(queryIndex2)
                        querySubjects(queryIndex2) = subjectTemp
                        queryWords(queryIndex1) = _
                              queryWords(queryIndex2)
                        queryWords(queryIndex2) = wordsTemp
                        querySharedWords(queryIndex1) = _
                              querySharedWords(queryIndex2)
                        querySharedWords(queryIndex2) = _
                              sharedWordsTemp
                        If queryIndex1 > 1 Then
                            queryIndex2 = queryIndex1
                            queryIndex1 = queryIndex2 - 1
                        Else
                            smallestQuery = True
                            firstQuery = True
                        End If
                    Else
                        smallestQuery = True
                        firstQuery = True
                    End If
                Loop
            Else
                firstQuery = True
            End If
        Loop
    ElseIf queryDistances(queryIndex1) = _
      queryDistances(queryIndex2) Then
        smallestQuery = False
        Do While smallestQuery = False
            If (queryNumbers(queryIndex1) > _
               queryNumbers(queryIndex2) And _
               queryDistances(queryIndex1) = _
               queryDistances(queryIndex2)) Then
                queryTemp = queryNumbers(queryIndex1)
                subjectTemp = querySubjects(queryIndex1)
                wordsTemp = queryWords(queryIndex1)
                sharedWordsTemp = querySharedWords(queryIndex1)
                queryNumbers(queryIndex1) = _
                    queryNumbers(queryIndex2)
```

```
                queryNumbers(queryIndex2) = queryTemp
                querySubjects(queryIndex1) = _
                    querySubjects(queryIndex2)
                querySubjects(queryIndex2) = subjectTemp
                queryWords(queryIndex1) = queryWords(queryIndex2)
                queryWords(queryIndex2) = wordsTemp
                querySharedWords(queryIndex1) = _
                    querySharedWords(queryIndex2)
                querySharedWords(queryIndex2) = sharedWordsTemp
                If queryIndex1 > 1 Then
                    queryIndex2 = queryIndex1
                    queryIndex1 = queryIndex2 - 1
                Else
                    smallestQuery = True
                End If
            Else
                smallestQuery = True
            End If
        Loop
    End If
    If placeHolder < numberOfQueries Then
        queryIndex1 = placeHolder
        queryIndex2 = queryIndex1 + 1
        placeHolder = queryIndex2
    Else
        sortComplete = True
    End If
Loop

distanceIndex = 1
lastQuery = False
Do While lastQuery = False
    rowIndex = headerRowsQSD + distanceIndex
    columnIndex = leaderColsQSD + queryIndex
    Worksheets(sortedSheet).Cells(rowIndex, columnIndex).Value _
          = queryDistances(distanceIndex)
    rowIndex = queryHeaderRowQSD + distanceIndex
    Worksheets(sortedSheet).Cells(rowIndex, columnIndex).Value _
          = queryNumbers(distanceIndex)
    rowIndex = subjectHeaderRowQSD + distanceIndex
    Worksheets(sortedSheet).Cells(rowIndex, columnIndex).Value _
          = querySubjects(distanceIndex)
    rowIndex = wordsHeaderRowQSD + distanceIndex
    Worksheets(sortedSheet).Cells(rowIndex, columnIndex).Value _
          = queryWords(distanceIndex)
    rowIndex = sharedWordsHeaderRowQSD + distanceIndex
    Worksheets(sortedSheet).Cells(rowIndex, columnIndex).Value _
          = querySharedWords(distanceIndex)
    If distanceIndex < numberOfQueries Then
        distanceIndex = distanceIndex + 1
    Else
        lastQuery = True
    End If
Loop
```

```
    If queryIndex < numberOfQueries Then
        queryIndex = queryIndex + 1
    Else
        lastQuerySorted = True
    End If
Loop

End Sub
```

Task 15: SubjectAvgQueryDistances Procedure

```
Private Sub OKButton_Click()

Dim numberOfQueries As Integer
Dim queryIndex As Integer
Dim subjectQueryIndex As Integer
Dim rowIndex As Integer
Dim columnIndex As Integer

Dim subjectNumber As Integer
Dim subjectRow As Integer
Dim prevSubject As Integer
Dim thatSubjectNumber As Integer
Dim queryNumber As Integer
Dim thatQueryNumber As Integer
Dim nextQuery As Integer

Dim queryDistance As Integer
Dim distancesSum As Integer
Dim distancesAvg As Single

Dim distanceStatsSheet As String
Dim allDistancesSheet As String

Dim lastQuery As Boolean
Dim lastSubjectQuery As Boolean

Dim outputHeaderRowDDS As Integer

Const headerRowsADS As Integer = 3
Const leaderColsADS As Integer = 3
Const queryRowADS As Integer = 1
Const subjectRowADS As Integer = 2
Const queryColumnADS As Integer = 1
Const subjectColumnADS As Integer = 2
Const x3ColumnDDS As Integer = 4

distanceStatsSheet = "DistanceStats"
allDistancesSheet = "DistancesAll"

outputHeaderRowDDS = Val(OutputHeaderRow.Text)

numberOfQueries = Val(QNumber.Text)
queryIndex = 1
prevSubject = 0
lastQuery = False
Do While lastQuery = False
    columnIndex = leaderColsADS + queryIndex
    queryNumber = Worksheets(allDistancesSheet).Cells(queryRowADS, _
          columnIndex).Value
    subjectNumber = Worksheets(allDistancesSheet). _
          Cells(subjectRowADS, columnIndex).Value
    If prevSubject < subjectNumber Then
```

```
        subjectRow = headerRowsADS + queryIndex
        prevSubject = subjectNumber
    End If
    subjectQueryIndex = 0
    thatQueryNumber = Worksheets(allDistancesSheet).Cells(subjectRow, _
         queryColumnADS).Value
    rowIndex = subjectRow
    distancesSum = 0
    lastSubjectQuery = False
    Do While lastSubjectQuery = False
        If queryNumber = thatQueryNumber Then
            rowIndex = rowIndex + 1
        Else
            subjectQueryIndex = subjectQueryIndex + 1
            queryDistance = Worksheets(allDistancesSheet). _
                   Cells(rowIndex, columnIndex).Value
            distancesSum = distancesSum + queryDistance
            rowIndex = rowIndex + 1
        End If
        thatSubjectNumber = Worksheets(allDistancesSheet). _
              Cells(rowIndex, subjectColumnADS).Value
        If thatSubjectNumber <> subjectNumber Then
            lastSubjectQuery = True
        Else
            thatQueryNumber = Worksheets(allDistancesSheet). _
                   Cells(rowIndex, queryColumnADS).Value
        End If
    Loop
    If subjectQueryIndex = 0 Then
        distancesAvg = 0
    Else
        distancesAvg = distancesSum / subjectQueryIndex
        rowIndex = outputHeaderRowDDS + queryIndex
        Worksheets(distanceStatsSheet).Cells(rowIndex, _
              x3ColumnDDS).Value = distancesAvg
    End If
    If queryIndex = numberOfQueries Then
        lastQuery = True
    Else
        queryIndex = queryIndex + 1
    End If
Loop

End Sub
```

Task Error! Reference source not found.**:** Error! Reference source not found. **Procedure**

```
Private Sub OKButton_Click()

Dim sharedWordsSheet As String
Dim allDistancesSheet As String
Dim distanceStatsSheet As String

Dim numberOfQueries As Integer
Dim numberOfSubjects As Integer

Dim nearQueriesList() As Integer
Dim nearDistancesList() As Integer
Dim nearSubjectsList() As Integer
Dim subjectLists() As Integer
Dim subjectListQueries() As Integer
Dim nearWordsList() As Integer
Dim nearSharedWordsList() As Integer
Dim listItems As Integer
Dim listSet As Integer
Dim queryIndex As Integer
Dim listIndex As Integer
Dim sortedListIndex As Integer
Dim rowIndex As Integer
Dim columnIndex As Integer
Dim outputColumnIndex As Integer

Dim thisQuery As Integer
Dim queryWords As Integer
Dim thatQuery As Integer
Dim queryNumber As Integer
Dim nextQueryNumber As Integer
Dim thisSubject As Integer
Dim subjectNumber As Integer
Dim thisDistance As Integer
Dim thatDistance As Integer

Dim queriesCount As Integer
Dim distancesSum As Long
Dim wordsSum As Integer
Dim sharedWordsSum As Integer
Dim avgDistance As Single
Dim x6Deviation As Single
Dim avgWords As Single
Dim x8Deviation As Single
Dim avgSharedWords As Single
Dim x10Deviation As Single
Dim avgSharedWordsAll As Single

Dim subjectsCount As Integer
Dim minSubjectQueries As Integer
Dim maxSubjectQueries As Integer
Dim subjectQueriesSum As Integer
Dim avgSubjectQueries As Single
```

```
Dim firstQueryFound As Boolean
Dim listFilled As Boolean
Dim queryWordsFound As Boolean
Dim lastQuery As Boolean
Dim lastSubject As Boolean
Dim lastListSet As Boolean
Dim taskEnd As Boolean

Const sortedDistancesRowSDS As Integer = 4
Dim sortedNumbersRowSDS As Integer
Dim sortedSubjectsRowSDS As Integer
Dim sortedWordsRowSDS As Integer
Dim sortedSharedWordsRowSDS As Integer
Dim outputRowDDS As Integer

Const listSize As Integer = 10
Const maxLists As Integer = 10
Const x1Column As Integer = 2   'number of Query keywords column
Const x2Column As Integer = 3
Const x3Column As Integer = 4
Const x4Column As Integer = 5
Const x5Column As Integer = 6
Const x6Column As Integer = 7
Const x7Column As Integer = 8
Const x8Column As Integer = 9
Const x9Column As Integer = 10
Const x10Column As Integer = 11
Const x11Column As Integer = 12
Const x12Column As Integer = 13
Const x13Column As Integer = 14
Const x14Column As Integer = 15
Const x15Column As Integer = 16

Const allDistancesSubjectCol As Integer = 2
Const sortedDistancesSheet As String = "QueriesSortedByDistance"
Const leaderColsSDS As Integer = 1
Const subjectRowSDS As Integer = 2
Const queryColumnSWS As Integer = 3
Const queryRowSWS As Integer = 3

numberOfSubjects = Val(SNumber.Text)
numberOfQueries = Val(QNumber.Text)

ReDim nearQueriesList(1 To numberOfQueries)
ReDim nearDistancesList(1 To numberOfQueries)
ReDim nearSubjectsList(1 To numberOfSubjects)
ReDim subjectLists(1 To numberOfSubjects)
ReDim subjectListQueries(1 To numberOfSubjects)
ReDim nearWordsList(1 To numberOfQueries)
ReDim nearSharedWordsList(1 To numberOfQueries)

sharedWordsSheet = "QuerySharedWords"
distanceStatsSheet = "DistanceStats"
```

```
allDistancesSheet = "DistancesAll"

If CellOptionButton.Value = True Then
    sortedNumbersRowSDS = 201
    sortedSubjectsRowSDS = 398
    sortedWordsRowSDS = 595
    sortedSharedWordsRowSDS = 792
    outputRowDDS = 214
Else
    sortedNumbersRowSDS = 197
    sortedSubjectsRowSDS = 390
    sortedWordsRowSDS = 583
    sortedSharedWordsRowSDS = 776
    outputRowDDS = 210
End If

queryIndex = 1
taskEnd = False
Do While taskEnd = False
    listSet = 1
    listIndex = 1
    sortedListIndex = 1
    columnIndex = leaderColsSDS + queryIndex
    thisSubject = Worksheets(sortedDistancesSheet). _
         Cells(subjectRowSDS, columnIndex).Value
    thisQuery = Worksheets(sortedDistancesSheet). _
         Cells(sortedNumbersRowSDS, columnIndex).Value
    queryNumber = thisQuery
    lastListSet = False
    Do While lastListSet = False
        rowIndex = sortedNumbersRowSDS + sortedListIndex
        nextQueryNumber = Worksheets(sortedDistancesSheet). _
              Cells(rowIndex, columnIndex).Value
        listFilled = False
        Do While listFilled = False
            If nextQueryNumber = queryNumber Then
                rowIndex = sortedSubjectsRowSDS + sortedListIndex
                subjectNumber = Worksheets(sortedDistancesSheet). _
                        Cells(rowIndex, columnIndex).Value
                If subjectNumber <> thisSubject Then
                    nearSubjectsList(subjectNumber) = _
                        nearSubjectsList(subjectNumber) + 1
                End If
                If sortedListIndex < numberOfQueries Then
                    sortedListIndex = sortedListIndex + 1
                    rowIndex = sortedNumbersRowSDS + sortedListIndex
                    nextQueryNumber = _
                        Worksheets(sortedDistancesSheet).Cells _
                             (rowIndex, columnIndex).Value
                Else
                    listFilled = True
                    lastListSet = True
                End If
            Else
```

```
nearQueriesList(listIndex) = nextQueryNumber
rowIndex = sortedDistancesRowSDS + sortedListIndex
nearDistancesList(listIndex) = _
   Worksheets(sortedDistancesSheet).Cells(rowIndex, _
        columnIndex).Value
rowIndex = sortedSubjectsRowSDS + sortedListIndex
subjectNumber = Worksheets(sortedDistancesSheet). _
        Cells(rowIndex, columnIndex).Value
nearSubjectsList(subjectNumber) = _
        nearSubjectsList(subjectNumber) + 1
rowIndex = sortedWordsRowSDS + sortedListIndex
nearWordsList(listIndex) = _
   Worksheets(sortedDistancesSheet).Cells(rowIndex, _
        columnIndex).Value
rowIndex = sortedSharedWordsRowSDS + sortedListIndex
nearSharedWordsList(listIndex) = _
   Worksheets(sortedDistancesSheet).Cells(rowIndex, _
        columnIndex).Value
If (listIndex = listSize) And _
   (sortedListIndex < numberOfQueries) Then
    thisDistance = nearDistancesList(listIndex)
    sortedListIndex = sortedListIndex + 1
    rowIndex = sortedDistancesRowSDS + sortedListIndex
    thatDistance = Worksheets(sortedDistancesSheet). _
       Cells(rowIndex, columnIndex).Value
    Do While listFilled = False
        If thisDistance = thatDistance Then
            subjectNumber = _
                 Worksheets(sortedDistancesSheet). _
                 Cells(sortedSubjectsRowSDS + _
                 sortedListIndex, columnIndex).Value
            nearSubjectsList(subjectNumber) = _
                 nearSubjectsList(subjectNumber) + 1
            queryNumber = nextQueryNumber
            rowIndex = sortedNumbersRowSDS + _
                          sortedListIndex
            nextQueryNumber = _
                 Worksheets(sortedDistancesSheet). _
                 Cells(rowIndex, columnIndex).Value
            If nextQueryNumber <> queryNumber Then
                listIndex = listIndex + 1
                nearQueriesList(listIndex) = _
                      nextQueryNumber
                rowIndex = sortedDistancesRowSDS + _
                              sortedListIndex
                nearDistancesList(listIndex) = _
                      Worksheets(sortedDistancesSheet) _
                           .Cells(rowIndex, _
                           columnIndex).Value
                rowIndex = sortedWordsRowSDS + _
                           sortedListIndex
                nearWordsList(listIndex) = _
                      Worksheets(sortedDistancesSheet) _
                      .Cells(rowIndex, _
```

```
                                columnIndex).Value
                            rowIndex = sortedSharedWordsRowSDS + _
                                    sortedListIndex
                            nearSharedWordsList(listIndex) = _
                                Worksheets(sortedDistancesSheet) _
                                .Cells(rowIndex, _
                                columnIndex).Value
                        End If
                        If sortedListIndex < numberOfQueries Then
                            sortedListIndex = sortedListIndex + 1
                            rowIndex = sortedDistancesRowSDS + _
                                    sortedListIndex
                            thatDistance = _
                                Worksheets(sortedDistancesSheet) _
                                .Cells(rowIndex, _
                                columnIndex).Value
                        Else
                            listFilled = True
                            lastListSet = True
                        End If
                    Else
                        listFilled = True
                    End If
                Loop
            Else
                If sortedListIndex < numberOfQueries Then
                    listIndex = listIndex + 1
                    queryNumber = nextQueryNumber
                    sortedListIndex = sortedListIndex + 1
                    rowIndex = sortedNumbersRowSDS + _
                                sortedListIndex
                    nextQueryNumber = _
                         Worksheets(sortedDistancesSheet). _
                              Cells(rowIndex, columnIndex).Value
                Else
                    listFilled = True
                    lastListSet = True
                End If
            End If
        End If
    Loop

    listIndex = 1
    queriesCount = 0
    distancesSum = 0
    wordsSum = 0
    sharedWordsSum = 0
    lastQuery = False
    Do While lastQuery = False
        queriesCount = queriesCount + 1
        distancesSum = distancesSum + nearDistancesList(listIndex)
        wordsSum = wordsSum + nearWordsList(listIndex)
        sharedWordsSum = sharedWordsSum + _
                              nearSharedWordsList(listIndex)
```

```
    nearQueriesList(listIndex) = 0
    nearDistancesList(listIndex) = 0
    nearWordsList(listIndex) = 0
    nearSharedWordsList(listIndex) = 0
    If nearQueriesList(listIndex + 1) = 0 Then
        lastQuery = True
    Else
        listIndex = listIndex + 1
    End If
Loop

rowIndex = outputRowDDS + queryIndex
avgDistance = distancesSum / listIndex
avgWords = wordsSum / listIndex
avgSharedWords = sharedWordsSum / listIndex
avgSharedWordsAll = _
      Worksheets(distanceStatsSheet).Cells(rowIndex, _
           x5Column).Value

x6Deviation = avgDistance - _
               Worksheets(distanceStatsSheet).Cells(rowIndex, _
               x3Column).Value
x8Deviation = avgWords - Worksheets(distanceStatsSheet). _
               Cells(rowIndex, x4Column).Value
x10Deviation = avgSharedWords - avgSharedWordsAll

outputColumnIndex = x6Column + (listSet - 1) * 10
Worksheets(distanceStatsSheet).Cells(rowIndex, _
      outputColumnIndex).Value = avgDistance

outputColumnIndex = x7Column + (listSet - 1) * 10
Worksheets(distanceStatsSheet).Cells(rowIndex, _
      outputColumnIndex).Value = x6Deviation

queryWords = Worksheets(distanceStatsSheet).Cells(rowIndex, _
               x1Column).Value
outputColumnIndex = x8Column + (listSet - 1) * 10
Worksheets(distanceStatsSheet).Cells(rowIndex, _
           outputColumnIndex).Value = queryWords - avgWords

outputColumnIndex = x9Column + (listSet - 1) * 10
Worksheets(distanceStatsSheet).Cells(rowIndex, _
           outputColumnIndex).Value = x8Deviation

outputColumnIndex = x10Column + (listSet - 1) * 10
Worksheets(distanceStatsSheet).Cells(rowIndex, _
           outputColumnIndex).Value = avgSharedWords

outputColumnIndex = x11Column + (listSet - 1) * 10
Worksheets(distanceStatsSheet).Cells(rowIndex, _
           outputColumnIndex).Value = x10Deviation

subjectsCount = 0
minSubjectQueries = 5
```

```
maxSubjectQueries = 1
subjectQueriesSum = 0
subjectNumber = 1
lastSubject = False
Do While lastSubject = False
    If nearSubjectsList(subjectNumber) > 0 Then
        subjectsCount = subjectsCount + 1
        If nearSubjectsList(subjectNumber) _
           < minSubjectQueries Then
            minSubjectQueries = nearSubjectsList(subjectNumber)
        End If
        If nearSubjectsList(subjectNumber) _
           > maxSubjectQueries Then
            maxSubjectQueries = nearSubjectsList(subjectNumber)
        End If
        subjectQueriesSum = subjectQueriesSum + _
                nearSubjectsList(subjectNumber)
        subjectLists(subjectNumber) = _
                subjectLists(subjectNumber) + 1
        subjectListQueries(subjectNumber) = _
            subjectListQueries(subjectNumber) + _
                    nearSubjectsList(subjectNumber)
    End If
    nearSubjectsList(subjectNumber) = 0
    If subjectNumber < numberOfSubjects Then
        subjectNumber = subjectNumber + 1
    Else
        lastSubject = True
    End If
Loop

avgSubjectQueries = subjectQueriesSum / subjectsCount
outputColumnIndex = x12Column + (listSet - 1) * 10
Worksheets(distanceStatsSheet).Cells(rowIndex, _
      outputColumnIndex).Value = subjectsCount
outputColumnIndex = x13Column + (listSet - 1) * 10
Worksheets(distanceStatsSheet).Cells(rowIndex, _
      outputColumnIndex).Value = minSubjectQueries
outputColumnIndex = x14Column + (listSet - 1) * 10
Worksheets(distanceStatsSheet).Cells(rowIndex, _
      outputColumnIndex).Value = maxSubjectQueries
outputColumnIndex = x15Column + (listSet - 1) * 10
Worksheets(distanceStatsSheet).Cells(rowIndex, _
      outputColumnIndex).Value = avgSubjectQueries
If lastListSet = False Then
    If listSet < maxLists Then
        listSet = listSet + 1
        listIndex = 1
        rowIndex = sortedNumbersRowSDS + sortedListIndex
        queryNumber = Worksheets(sortedDistancesSheet). _
           Cells(rowIndex, columnIndex).Value
        sortedListIndex = sortedListIndex + 1
    Else
        lastListSet = True
```

```
            End If
        End If
    Loop
    If queryIndex < numberOfQueries Then
        queryIndex = queryIndex + 1
    Else
        taskEnd = True
    End If
Loop

End Sub
```

REFERENCES

[1] Amo, P., Ferraras, F.L., Cruz, F., and Rosa, M., 2000. "Smoothing Functions for Automatic Relevance Feedback in Information Retrieval", *Proceedings of the 11th International Workshop on Database and Expert Systems Applications*, pp. 115-119.

[2] Anick, P.G. and Tipirneni, S., 1999. "The Paraphrase Search Assistant: Terminological Feedback for Iterative Information Seeking", *Proceedings of the 22nd Annual International ACM SIGIR Conference*, pp. 153-159.

[3] Appelt, D. E., 1985. *Planning English Sentences*. Cambridge University Press.

[4] Armitage, P.and Berry, G., 1994. *Statistical Methods in Medical Research - 3rd Edition*. Blackwell.

[5] Armstrong, R., Freitag, D., Joachims, T., and Mitchell, T., 1995. "WebWatcher: A Learning Apprentice for the World Wide Web", *Proceedings of AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Resources*, pp. 6-12.

[6] Aumann, R., 1976. "Agreeing to Disagree", *Annals of Statistics*, Vol. 4, pp. 1236-1239.

[7] Austin, J.L., 1962. *How to do Things with Words*. Oxford U. Press, New York.

[8] Baeza-Yates, R. and Ribeiro-Neto, B., 1999. *Modern Information Retrieval*. ACM Press, Addison-Wesley, 1999.

[9] Bahle, D., Williams, H.E., and Zobel, J., 2001. "Optimized Phrase Querying and Browsing in Text Databases", *Proceedings of the Australasian Computer Science Conference*, pp. 11-19.

[10] Balabanovic, M. and Shoham, Y., 1995. "Learning Information Retrieval Agents: Experiments with Automated Web Browsing", *Proceedings of AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Resources*, pp. 13-18.

[11] Bates, M.J., 1989. "The Design of Browsing and Berrypicking Techniques for the Online Search Interface", *Online Review*, Vol. 13, pp. 407-424.

[12] Belkin, N.J. and Croft, W.B. 1992. "Information Filtering and Information Retrieval: Two Sides of the Same Coin?", *Communications of the ACM*, 35(12), pp. 29-38.

[13] Berry, M.W. and Browne, M., 1999. "Understanding Search Engines: Mathematical Modeling and Text Retrieval", *Society for Industrial and Applied Mathematics*. Philadelphia, PA.

[14] Bilange, E., 1991. "A Task Independent Oral Dialogue Model", *Proceedings of the Fifth Conference of the European Chapter of the Association for Computational Linguistics*, pp. 83-88.

[15] Bratman, M.E., 1987. *Intentions, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA.

[16] Bruza, P.D. and Dennis, S., 1997. "Query re-formulation on the Internet: Empirical Data and the Hyperindex Search Engine". *Proceedings of the RIAO97 Conference - Computer-Assisted Information Searching on Internet*, Centre de Hautes Etudes Internationales d'Informatique Documentaires, pp. 488-499.

[17] Bruza, P.D., McArthur, R., and Dennis, S., 2000. "Interactive Internet Search: Keyword, Directory and Query Reformulation Mechanisms Compared", *Proceedings of the 23 rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[18] Byrne, C.C. and McCracken, S.A., 1999. "An Adaptive Thesaurus Employing Semantic distance, Relational Inheritance and Nominal Compound Interpretation for Linguistic Support of Information Retrieval", *Journal of Information Science*, 25(2), pp. 113-131.

[19] Cai, D. and van Rijsbergen, C.J., 2002. "Automatic Query Expansion Based on Direct Divergence". *Proceedings of the International Conference on Information Technology: Coding and Computing*, pp. 8-15.

[20] Carmel, E., Crawford, S., and Chen, H., 1992. "Browsing in Hypertext: A Cognitive Study", *IEEE Transactions on Systems, Man, and Cybernetics*, 22(5), pp. 865-883.

[21] Catledge, L.D. and Pitkow, J.E., 1995. "Characterizing Browsing Strategies in the World-Wide Web", *Computer Networks and ISDN Systems*, 27(6), pp. 1065-1073.

[22] Chalmers, M. and Chitson, P, 1992. "Bead: Exploration in Information Visualization", *Proceedings of the 15th Annual International ACM/SIGIR Conference*, pp. 330-337.

[23] Chen, L. and Sycara, K., 1998. "WebMate: A Personal Agent for Browsing and Searching", *Autonomous Agents '98 Conference*, pp. 132-139.

[24] Clark, H.H. and Wilkes-Gibbs, D., 1986. "Referring as a Collaborative Process", *Cognition*, 22(1), pp. 1-39.

[25] Cockburn, A. and Jones, S., 1996. "Which Way Now? Analyzing and Easing Inadequacies in WWW Navigation", *International Journal of Human-Computer Studies*, Vol. 45, pp. 105-129.

[26] Courtney, A., Janssen, W., Severson, D., Spreitzer, M., and Wymor, F., 1994. *Inter-Language Unification, release 1.5*, Xerox PARC.

[27] Cove, J.F. and B.C. Walsh, 1988. "Online Text Retrieval via Browsing", *Information Processing and Management*, 24(1), pp. 31-37.

[28] Croft, W.B., 1980. "A Model of Cluster Searching Based on Classification", *Information Systems*, Vol. 5, pp. 189-195.

[29] Cutting, D.R., Karger, D.R., and Pedersen, J.O., 1993. "Constant Interaction-Time Scatter/Gather Browsing of Very Large Document Collections", *Proceedings of the 16th Annual International ACM/SIGIR Conference*, pp. 126-135.

[30] Cutting, D.R., Karger, D.R., Pedersen, J.O., and Tukey, J.W., 1992. "Scatter/Gather: A Cluster-Based Approach to Browsing Large Document Collections", *Proceedings of the 15th Annual International ACM/SIGIR Conference on R&D in IR*, pp. 318-329.

[31] Dennis, S., McArthur, R., and Bruza, P.D., 1998. "Searching the World Wide Web Made Easy? The Cognitive Load Imposed by Query Refinement Mechanisms", *Proceedings of the Third Australian Document Computing Symposium*, Department of Computer Science, University of Sydney, TR-518, pp. 65-71.

[32] Egghe, L. and Rousseau, R., 1998. "A Theoretical Study of Recall and Precision Using a Topological Approach to Information Retrieval", *Information Processing & Management*, 34(2), pp. 191-218.

[33] Ellis, D., 1989. "A Behavioural Approach to Information Retrieval System Design", *Journal of Documentation*, 45(3), pp. 171-212.

[34] Fitzpatrick, L. and Dent, M., 1997. "Automatic Feedback Using Past Queries: Social Searching?", *Proceedings of the 20th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 306-313.

[35] Fowler, R.H., Fowler, W.A.L., and Wilson, B.A., 1991. "Integrating Query, Thesaurus, and Documents through a Common Visual Representation", *Proceedings of the 14th Annual International ACM/SIGIR Conference*, pp. 142-151.

[36] Fragoudis, D. and Likothanassis, S.D., 1999. "Retriever: A Self-Training Agent for Intelligent Information Discovery", *Proceedings of the 1999 International Conference on Information Intelligence and Systems*, pp. 594-599.

[37] Garnham, A. 1982. “Testing Psychological Theories about Inference Making”, *Memory and Cognition*, 10(4), pp. 341-349.

[38] Gauch, S., Wang, G., and Gomez, M., 1996. “ProFusion: Intelligent Fusion from Multiple, Distributed Search Engines”, *Journal of Universal Computer Science*, 2(9), pp. 637-649.

[39] Glover, E.J., Flake, G.W., Lawrence, S., Birmingham, W.P., Kruger, A., Giles, C.L., and Pennock, D.M., 2001. “Improving Category Specific Web Search by Learning Query Modifications”, *Symposium on Applications and the Internet*, pp. 23-31.

[40] Glover, E.J., Lawrence, S., Birmingham, W.P., and Giles, C.L., 1999. “Architect of a Metasearch Engine that Supports User Information Needs”, *Eighth International Conference on Information and Knowledge Management*, pp. 210-216.

[41] Grice, H. 1975. “Logic and conversation”, in *Speech Acts*, edited by P. Cole, & Morgan, J., Academic Press, New York.

[42] Gross, J.L. and Yellen, J. (Editors), 2003. *Handbook of Graph Theory*, Columbia University, New York, NY.

[43] Han, S., Boley, D., Gini, M., Gross, R., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., and Moore, J., 1998. “WebACE: A Web Agent for Document Categorization and Exploration”, *Autonomous Agents ’98 Conference*, pp. 408-415.

[44] Harman, D., 1988. “Towards Interactive Query Expansion”, *Proceedings of the Eleventh Annual International ACMSIGIR Conference on Research & Development in Information Retrieval*, pp. 321-331.

[45] Hearst, M.A., 1995. “Tilebars: Visualization and Term Distribution Information in Full Text Information Access”, *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*.

[46] Hearst, M.A., Karger, D., and Pedersen, J.O., 1995. “Scatter/Gather as a Tool for the Navigation of Retrieval Results”, *Working Notes of the AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval*, Robert Burke, editor.

[47] Hill, W.C., Hollan, J.D., Wroblewski, D., and McCandless, T. 1992. “Read Wear and Edit Wear”, *Proceedings of ACM Conference on Human Factors in Computing Systems*, *CHI ’92*, pp. 3-9.

[48] Hogeweg-de-Haart, H.P., 1984. “Characteristics of Social Science Information: A Selective Review of the Literature. Part II”, *Social Science Information Studies*, 4(1), pp. 15-30.

[49] Horacek, H. 1997. "A Model for Adapting Explanations to the User's Likely Inferences", *User Modeling and User-Adapted Interaction*, 7(1), pp. 1-55.

[50] Howe, A.E. and Dreilinger, D., 1997. "SavvySearch: A Meta-Search Engine that Learns which Search Engines to Query", *AI Magazine*, 18(2), pp. 19-25.

[51] http://web.realnames.com/Virtual.asp?page=Eng_Corporate_PressRelease_053100

[52] http://www.clienthelpdesk.com/statistics_research/web_statistics.html

[53] http://www.searchengines.com

[54] http://www.webtop.com/search/vanilla/press210800_3.htm

[55] Jaccard, P., 1901. "Etude Comparative de la distribution florale dans une portion des Alpes et du Jura", *Bulletin de la Societe vaudoise des Sciences Naturelles*, Vol. 37, pp. 547-579.

[56] Jaczynski, M. and Trousse, B., 1997. "Broadway: A World Wide Web Advisor Reusing Past Navigations from a Group of Users", *Proceedings of 3rd UK Workshop of Case-Based Reasoning (UKCBR '97)*.

[57] Jansen, B.J., Spink, A., and Saracevic, T., 1998. "Failure Analysis in Query Construction: Data and Analysis from a Large Sample of Web Queries", *Proceedings of the Third ACM Conference on Digital Libraries*, pp. 289-290.

[58] Jardine, N. and van Rijsbergen, C.J., 1971. "The Use of Hierarchical Clustering in Information Retrieval", *Information Storage and Retrieval*, Vol. 7, pp. 217-240.

[59] Kahn, I. and Card, H., 1998. "Adaptive Information Agents Using Competitive Learning", *Journal of Network and Computer Applications*, Vol. 21: 69-89.

[60] Kintsch, W., Keenan, J. and McKoon, G. 1974. "Memory for Information Inferred During Reading", In *The Representation of Meaning in Memory*, W. Kintsch (ed.), Earlbaum, Hillsdale, New Jersey.

[61] Korfhage, R.R., 1991. "To See or Not to See – Is that the Query?", *Proceedings of the 14th Annual International ACM/SIGIR Conference*, pp. 134-141.

[62] Larson, R.R., 1992. "Experiments in Automatic Library of Congress Classification", *Journal of the American Society for Information Science*, 43(2), pp. 130-148.

[63] Lau, T. and Horvitz, E., 1999. "Patterns of Search: Analyzing and Modeling Web Query Refinement", *Proceedings of the 7th International Conference on User Modeling*, pp. 119-128.

[64] Lewis, D., 1969. *Convention: A Philosophical Study*, Harvard University Press, Cambridge, Massachusetts.

[65] Lieberman, H., 1995. "Letizia: An Agent That Assists Web Browsing", *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pp. 924-929.

[66] Line, M.B., 1974. "Information Requirements in the Social Sciences", *Proceedings of the Conference on Access to Knowledge and Information in the Social Sciences and Humanities*, pp. 146-158.

[67] Marchionini, G., 1989. "Information-seeking strategies of novices using a full-text electronic encyclopedia", *Journal of the American Society for Information Science*, 40(1), pp. 54-66.

[68] Menczer, F. and Belew, R.K., 1998. "Adaptive Information Agents in Distributed Textual Environments", *Autonomous Agents '98 Conference*, pp. 157-164.

[69] Miller, D.R.H., Leek, T., and Schwartz, R.M., 1999. "BBN at TREC-7: Using Hidden Markov Models for Information Retrieval", *Proceedings of the Seventh Text Retrieval Conference, TREC-7*. NIST Special Publications.

[70] Moldovan, D.I. and Mihalcea, R., 2000. "Using WordNet and Lexical Operators to Improve Internet Searches", *IEEE Internet Computing*, 4(1): 34-43.

[71] Montgomery, D.C., Peck, E.A., and Vining, G.G., 2001. *Introduction to Linear Regression Analysis – 3rd Edition*. John Wiley & Sons, Inc.

[72] Morita, M. and Shinoda, Y. 1994. "Information Filtering Based on User Behavior Analysis and Best Match Text Retrieval", *Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pp. 272-281.

[73] Moukas, A., 1997. "Amalthaea: Information Discovery and Filtering Using a Multiagent Evolving Ecosystem", *Applied Artificial Intelligence: An International Journal*, 11(5): 437-457.

[74] Nevill-Manning, C.G. and Witten, I.H., 1997. "Compression and Explanation Using Hierarchical Grammars", *Computer Journal*, 40(2), pp. 103-116.

[75] Pirolli, P., Schank, P., Hearst, M.A., and Diehl, C., 1996. "Scatter/Gather Browsing Communicates the Topic Structure of a Very Large Text Collection", *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pp. 213-220.

[76] Polanyi, R. and Scha, R., 1984. “A Syntactic Approach to Discourse Semantics”, *Proceedings of the 10th International Conference on Computational Linguistics*, pp. 413-419.

[77] Raghavan, V.V. and Sever, H., 1995. “On the Reuse of Past Optimal Queries”, *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 344-350.

[78] Rich, E.A. 1979. “User Modeling via Stereotypes”, *Cognitive Science*, 3, pp. 329-354.

[79] Robertson, S.E. and Jones, K.S., 1976. “Relevance Weighting of Search Terms”, *Journal of the American Society for Information Science*, January, pp. 129-146.

[80] Robertson, S.E., Walker, S., and Beaulieu, M., 1999. “Okapi at TREC-7: Automatic ad hoc, Filtering, VLC and Interactive Track”, *Proceedings of the TREC-7, NIST Special Publication 500-242*.

[81] Rocchio, J., 1971. “Relevance Feedback Information Retrieval”, *The Smart Retrieval System: Experiments in Automatic Document Processing*, Gerard Salton, editor. Prentice-Hall, Englewood Cliffs, NJ.

[82] Rousseau, R., 1998. “Jaccard Similarity Leads to the Marczewski-Steinhaus Topology for Information Retrieval”, *Information Processing & Management*, 34(1), pp. 87-94.

[83] Salton, G., 1968. *Automatic Information Organization and Retrieval*. McGraw-Hill, NY.

[84] Salton, G., 1971. *The Smart Retrieval System – Experiments in Automatic Document Retrieval*. Prentice Hall Inc. Englewood Cliffs, NJ.

[85] Salton, G., 1988. *Automatic Text Processing*. Addison-Wesley.

[86] Salton, G. and Buckley, C., 1988. “Term-weighting Approaches in Automatic Text Retrieval”, *Information Processing & Management*, 24(5), pp. 513–523.

[87] Salton, G. and Buckley, C., 1990. “Improving Retrieval Performance by Relevance Feedback", *Journal of the American Society for Information Science*, 41(4), pp. 288-297.

[88] Schiffer, S., 1972. *Meaning*, Oxford University Press, Oxford.

[89] Schindler, E., 2004. “Search Party”, *Networker*, 8(4), pp. 28-32.

[90] Schvaneveldt, R. W. (Editor), 1990. *Pathfinder Associative Networks: Studies in Knowledge Organization*, Ablex, Norwood, NJ.

[91] Searle, J.R., 1975. "Indirect Speech Acts", *Syntax and Semantics3: Speech Acts*, P. Cole and J. Morgan, editors, Academic Press, New York, pp.59-82.

[92] Silverstein, C., Henzinger, M., Marais, H., and Moricz, M. 1998. "Analysis of a Very Large Alta Vista Query Log", RC Technical Note 1998-014, http://www.research.digital.com/SRC/.

[93] Spoerri, A., 1993. "InfoCrystal: A Visual Tool for Information Retrieval and Management", *Proceedings of Information Knowledge and Management*, Washington, D.C.

[94] Stevens, C. 1992b. "Knowledge-Based Assistance for Accessing Large, Poorly Structured Information Spaces", Ph.D. thesis, University of Colorado, Department of Computer Science, Boulder. http://www.holodeck.com/curt/mypapers.html.

[95] Stoan, S.K., 1984. "Research and Library Skills: An Analysis and Interpretation", *College & Research Libraries*, 45(2), pp. 99-109.

[96] Stone, S., 1982. "Humanities Scholars: Information Needs and Uses", *Journal of Documentation*, 38(4), pp. 292-312.

[97] Tauscher, L. and Greenberg, S., 1997. "How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems", *International Journal of Human-Computer Studies*, Vol. 47, pp. 97-137.

[98] Taylor, R.S. 1962. "The Process of Asking Questions", *American Documentation*, 13(4), pp. 391-396.

[99] Thompson, R.H. and Croft, B.W., 1989. "Support for Browsing in an Intelligent Text Retrieval System", *International Journal of Man [sic] -Machine Studies*, 30(6), pp. 639-668.

[100] Thuring, M. and Wender, K. 1985. "Über kausale Inferenzen beim Lesen", *Sprache und Kognition*, 2, pp. 76-86.

[101] van Rijsbergen, C.J., 1980. *Information Retrieval*. Butterworths, London.

[102] Velez, B., Weiss, R., Sheldon, M., and Gifford D., 1997. "Fast and Effective Query Refinement", *Proceedings of the 20th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pp. 6-15.

[103] Voorhees, E.M., Gupta, N.K., and Johnson-Laird, B., 1995. "The Collection Fusion Problem", *Proceedings of the Third Text Retrieval Conference TREC-3*, pp. 95-104.

[104] Wildemuth, B.M., 2004. "The Effects of Domain Knowledge on Search Tactic Formulation", *Journal of the American Society for Information Science and Technology*, 55(3), pp. 246-258.

[105] Willett, P., 1988. "Recent Trends in Hierarchical Document Clustering: A Critical Review", *Information Processing and Management*, 24(5), pp. 577-597.

[106] Williams, H., Zobel, J., and Anderson, P., 1999. "What's next? Index Structures for Efficient Phrase Querying", *Proceedings of the Australasian Database Conference*, pp. 141-152.

[107] Worona, S., 1971. "Query Clustering in a Large Document Space", *The SMART Retrieval System*, G. Salton, editor, pp. 298-310.

[108] Xu, J. and Croft, W. 1996. "Query Expansion Using Local and Global Document Analysis", *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 4-11.

VITA

Craig Alan Nowack

Education

M.S., Industrial Engineering with concentration in Operations Research
Penn State University
University Park, PA
December 1996

B.S., Mathematics
SUNY College at Brockport
Brockport, NY
December 1993

Experience

Graduate Research Assistant
Penn State University
University Park, PA
1994-1996, 2001-2004
Course Development, Simulation Modeling

Graduate Teaching Assistant
Penn State University
University Park, PA
1998–2000

Logistical Planner
Praxair, Inc.
Tonawanda, NY
1996–1997

Penn State Industrial Engineering Graduate Association
Secretary

Penn State International Futbol Club
President