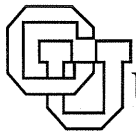
Spoken-Language Help for High-Functionality Applications

Michael Paul Jones

CU-CS-841-97



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

SPOKEN-LANGUAGE HELP FOR HIGH-FUNCTIONALITY APPLICATIONS

by

MICHAEL PAUL JONES

B.S., Brigham Young University, 1990

M.S., University of Colorado, 1992

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

1997

This thesis entitled:

Spoken-Language Help for High-Functionality Applications

written by Michael Paul Jones

has been approved for the Department of Computer Science

_____

James H. Martin

_____

Thomas K. Landauer

Date _____

The final copy of this thesis has been examined by the signators, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Jones, Michael Paul (Ph.D., Computer Science)
Spoken-Language Help for High-Functionality Applications
Thesis directed by Associate Professor James H. Martin

Modern, high-functionality computer applications are difficult to learn and use. Users of these applications frequently need help to complete their work. However, traditional keyword-based help systems are awkward to use and they frequently fail to satisfy their users' information needs. Allowing users to ask for help in their own words offers a more natural interaction than provided by keyword systems. Analysis of the language used in the questions facilitates the successful retrieval of answers at a far superior rate than traditional systems.

This dissertation describes SAGE, a help system developed by combining techniques gleaned from the information retrieval (IR) and artificial intelligence (AI) communities. SAGE responds to users' questions by retrieving answers from an existing collection of help text. The kernel of SAGE is based on an advanced IR technique called Latent Semantic Indexing (LSI), a variant of the vector space model. This kernel is then enhanced by adding robust natural language processing and machine learning techniques.

Both the development and evaluation of SAGE have been empirical efforts. Live user studies with a simulated system were used to gather data crucial to the design of the system. Evaluation of the system was based on live user studies and on relevance judgments for a set of naturally occurring queries.

The results of these evaluations indicate that the SAGE approach can be quite effective when deployed in realistic settings. In particular, the SAGE results have shown that given unconstrained natural language questions, answers to users' information needs can be retrieved from existing help text.

DEDICATION

To my very understanding and supportive wife, Heather.

ACKNOWLEDGEMENTS

I want to begin by thanking my parents for their advice, support, and encouragement throughout my academic career. They have always tried to establish a love for learning in their children and they always urged us to do our best.

Next I want to thank all my friends and colleagues at the University of Colorado in Boulder. Primarily, I am grateful to my advisor, Jim Martin, for helping me during my tenure at CU. Sometimes I felt that he was just the crutch I needed to lean on when I was frustrated. At other times in my research, he seemed to toss me in the proverbial deep end where I had to learn to swim by myself. I appreciate his willingness to let me pursue my research interests even when they did not mesh well with his own. He always concerned himself with my needs and interests. Again, thanks for everything Jim.

I appreciate the insights and time given by the other members of my committee. Thanks go to Tom Landauer, Clayton Lewis, Liz Jessup, and Dan Jurafsky for sharing their wisdom, knowledge, and experiences with me.

I have also enjoyed associations with some wonderful, fellow graduate students. I apologize if I have forgotten someone in the following list. I appreciate Keith Vander Linden, Nick Wilde, and Cathleen Wharton for imparting tidbits of wisdom before departing. Casey Boyd and I entered the program together and now we are leaving at the same time. He helped me immeasurably through the hurdles, red-tape, classes, prelims, job search, and good and bad times all associated with my graduate school experience. I appreciate Laura Mather and Carlos Maltzahn for giving me a taste of the synergy that can be generated within a well-functioning research group. Also, I need to thank Laura for enduring my many repetitive (and I am sure, tiring) questions about linear algebra and the singular value decomposition of matrices. She never seemed to mind answering the same questions over and over.

My research included two user studies. I am grateful to the many paid and unpaid volunteers who acted as subjects and assistants during the experiments. I especially appreciate Taimi Metzlers and Karl Griepentrog for the many unpaid hours they volunteered.

I am thankful for the financial support I was given. The Computer Science department supplied me with opportunities to work as a teaching and research assistant. Contracts from US West and DARPA permitted me to work on projects related to my dissertation research. I also want to acknowledge Marc Rochkind for employing me and for allowing me to work an odd and varied schedule to meet my school commitments.

Finally and most importantly, I never would have completed my degree without the love, encouragement, and support of my family. I appreciate my daughters, Amber, Ashlee, and Emily for understanding when their daddy did not have as much time to spend with them as they wished. They also lifted my spirits with their smiles and hugs each evening when I returned home. My dear wife, Heather,

never doubted my ability to finish and she would not let me doubt myself either. I want to thank her for her involvement and interest in my work throughout my studies. My frustrations became her frustrations and my triumphs became her triumphs. Heather, I appreciate your faith in me and the constant source of strength and support you always are to me.

CONTENTS

FIGURES

TABLES

CHAPTER 1

INTRODUCTION

In spite of their designers' best efforts, existing high-functionality, commercial applications (e.g., word processors, spreadsheets, and databases) are not always intuitive to their users. Frequently, users must find some way to get help with an application to complete their work. New versions of the software usually increase functionality without increasing the users' ability to get help on using the functionality available.

Competition among vendors of these applications is fierce. To gain a competitive edge, each new release of an application includes new and often more complex features. The addition of features also complicates the user interface as new options and functionalities are crammed into already crowded menus, dialog boxes, and tool button bars. Typically, such a plethora of features abounds that no one could hope to understand how to use all of them. Users of a particular application have certain expectations about what tasks can be accomplished with the application, but often they lack the knowledge of how to complete a desired task. What then are the users to do when faced with an unfamiliar task?

Usually, users can select an appropriate application for solving their computing tasks; however, they must seek help occasionally to complete some of their tasks. Existing help systems are often inadequate because users cannot consistently use them effectively. They become frustrated with the help system and skeptical about its ability to provide answers to their questions. This often leads them to conclude that the help system is useless and consequently, they cease using it.

The research presented in this dissertation is concerned with easing the use of application features by improving access to information about those features. The focus of study is on users trying to complete specific tasks and the questions they ask while doing so. A help system, SAGE (**S**earch for **A**nswers, **G**rant **E**nlightenment), is developed to answer questions through a combination of techniques gleaned from the information retrieval and artificial intelligence communities. Finally, the implementation of SAGE is verified by real users attempting to solve real tasks.

1.1 Task-based Problem Solving

In his study on learning strategies and exploratory behavior of computer users, Rieman (1994) found that the dominant strategies for resolving problems or "figuring out how to do it" were, in order of preference:

(1) try things out (e.g., exploration of the interface),
(2) look in a printed manual,
(3) ask for help (usually from colleagues),
(4) work around the problem, and
(5) use the on-line help facility.

The interesting result from this study is that the preferred strategy for problem resolution was not a help-seeking strategy. Instead, the preferred strategy was to explore the various components available in the interface to discover a solution.

Note that while strategy number four, working around a problem, is acceptable and possible for some tasks, it is really a problem avoidance strategy rather than a problem resolution strategy. Consequently, it is outside the scope of the subsequent discussion and is not included.

1.1.1 Exploratory Learning Postexperiment interviews conducted during Rieman's study revealed two primary reasons for subjects' choice of exploratory learning for solving their task specific problems. The first was that some subjects achieved a sense of satisfaction and independence by discovering things for themselves. They enjoyed hunting around within the interface for new features even when the strategy might prove more time-consuming than an alternative.

The second reason for choosing to explore was that it was perceived as the only viable option available. Subjects commented that they often misplaced manuals, thereby reducing their effectiveness. Some subjects did not have access to a knowledgeable person who could help, either because they did not know anyone familiar with the application or the local expert was unavailable. Many had tried using on-line help in the past, but frequently, the interaction was unsuccessful. Consequently, they did not have confidence that they could find the needed help there.

1.1.2 Look in a Manual The second most preferred strategy was searching for help in a printed manual. Using a manual has its advantages and disadvantages. If a good match exists between users' vocabulary and the index, the desired information can usually be found quite quickly. However, finding the correct information often takes just as much searching within the manual as the exploratory learning strategy does within the interface. Experienced users become more familiar with the language of the application and are more successful with a manual, but novice users have a particularly difficult time locating information they need in the manual (Franzke and Rieman, 1993).

An interesting caveat to this learning strategy surfaced during interviews with the subjects. For many of them, the manufacturer's manual that accompanies the software was not the manual of choice. Instead, they turned to a popular "How-to-use-Program-X" type of manual published by a third party. Often, the third party guides are written with a focus on solving problems and accomplishing tasks rather than providing an overview of the application.

The fact that manuals are not part of the software can also be either an advantage or disadvantage. A typical high-functionality application uses most of the display real estate, leaving little room for on-line help text to be displayed. A manual provides the benefit of not disturbing or occluding the current contents of the display. However, looking up information in a manual becomes problematic when it is misplaced.

1.1.3 Ask for Help The options available for this strategy include calling a technical support phone number, asking a colleague, or posting a question to a relevant Internet newsgroup. Asking for help is an appealing problem solving

strategy because the user can ask questions in a natural way and a knowledgeable person is most likely to be able to help solve the problem.

A technical support line is staffed by trained personnel who are experienced at answering questions about an application. Consequently, the likelihood of receiving a satisfying answer from a call to a technical support line is high. However, this help-seeking strategy is rarely used. The probable reason is the time involved in finding the phone number, perhaps navigating through touch-tone menus to direct the call, waiting on hold, and sometimes waiting for a return call. Many vendors also charge for this service, thus further discouraging its use.

Asking a colleague is a promising strategy if a knowledgeable colleague who does not mind answering questions is available. Unfortunately, a local expert is frequently unavailable to users. Some subjects noted that they are usually the most experienced users of the software and they could only get help from the phone-support lines. Other biases against asking for help include not wanting to be embarrassed by asking a novice question or being uncomfortable interrupting someone from their work.

Occasionally, Internet news can be a successful medium for receiving help. The drawback to this approach is that obtaining an answer usually takes more time than users are willing to wait. Also, the question is not directed to anyone in particular and thus, there is a distinct possibility that no one will respond.

1.1.4 On-line Help As with the other problem resolution mechanisms, on-line help has its advantages and disadvantages. From a purely economic stand point, on-line help is cheaper to produce and ship than a printed manual. Of course, this benefit is directly realized by the software producers, but it may also be reflected in the software's purchase price. Also, on-line help provides certain benefits that manuals do not. It cannot be misplaced like the manual might. The widespread use and availability of the Internet enable on-line help systems to be updated or corrected electronically as well, thus benefiting the user through accuracy and completeness of the contents. For occasional users of word processing systems, on-line help has been shown to improve their editing performance (Santhanam and Wiedenbeck, 1991).

Just as a manual might be lost and thus become unavailable as a learning strategy, asking for help is not always an option because an expert may not be available. Conversely, an on-line help system does not go out of town or get sick and therefore, it is always available to answer questions. Furthermore, some colleagues might be bothered by interruptions, but the system never will be.

In spite of the many benefits, on-line help systems were not identified as a preferred method for obtaining help. The participants in Rieman's study described frustrations trying to use on-line help effectively. They stated that often they did not understand the help messages or else the response clearly did not answer their questions. Many described on-line help as "useless."

Mack, Lewis, and Carroll (1983) reported similar problems in their studies of on-line help. Some problems resulted from the help text itself. It was either hard for users to understand (e.g., the use of jargon) or too long (in which case users

balked at searching through it for the specific information of interest). Other problems described include a problem description that did not match any of the system's help categories (the vocabulary problem (Furnas et al., 1987)) and the inability of the system to infer enough context to provide useful suggestions. Additional examinations of the messages in on-line help systems are provided by (Lang, Graesser, and Hemphill, 1990), (Graesser, Person, and Huber, 1992), (Fach, Bannert, and Kunkel, 1993), and (Yetim, 1993).

1.2 Why On-line Help?

The previously mentioned research has exposed some weaknesses of existing on-line help. It is notoriously "user-unfriendly." Thus, the question **"Why would we expect a speech-based, natural language help system to be useful?"** becomes important.

There are two primary criticisms of on-line help.

(1) Finding answers within an on-line help system is difficult.

(2) On-line help messages are difficult to understand.

These problems may partially contribute to each other. It is probably true that even when the "right" piece of information is found, users will not recognize that it is the information they sought if they have a hard time understanding it. They then come to the conclusion (and justifiably so) that using the system is not successful enough to warrant the effort.

The primary benefit of on-line help as an inexpensive, constant, and tireless source of information is so compelling that it is worth investigating methods for overcoming its deficiencies. The focus of the research presented in this dissertation is on the first criticism, finding answers to users' questions. However, the research also provides some suggestions for improving the help text itself. These suggestions come from an analysis of the questions asked by users and the information they were seeking.

One cause of difficulty in finding information within traditional keyword-based help systems, is simply an issue of interaction style. Navigating a keyword help system can be cumbersome. In contrast, human language is a more natural communication medium for people. Software users who seek help from a colleague can request help in an easy, natural way and they can receive a correct, focused answer (assuming the colleague knows the answer and is willing to help!) A spoken language help system provides a similar communication medium as the helpful colleague, thus easing the interaction for the users. It is also designed to accept questions in the users' own language rather than forcing them to use specific keywords. The system described in this dissertation also gives more focused answers than traditional help systems by analyzing parts of the question and using that information to find an appropriate answer.

1.3 A User's Plight

To gain insight into the problems encountered when using the help system that comes with a popular commercial word processor, I obtained permission from a user to observe as she produced a short document. For the sake of reference, she is

named Amy[1] in the description of her experience. I believe Amy's experience with the help system was typical of many users.

As part of the document, Amy wanted to create a bulleted list. When she came to this task, she turned to the help system. She began scanning the keyword list provided with the system. Her interaction with me and the help system went something like the following.

Amy "I don't know the **right** name for these [bullets], so I don't think I'll be able to find the answer." (emphasis added).

me "How would you ask me for help if I couldn't see what you are working on?"

Amy "I would ask you, 'How do I add dots to indicate items in a list?' "

me "Try using **list** as a keyword."

Amy followed the suggestion and the system presented a selection of help page titles related to lists. She scanned them and the conversation proceeded.

Amy "I don't think any of these are what I want." (The title of the page she needed was among those presented).

me "The dots are called bullets. Try using **bullet** as a keyword."

She entered the recommended keyword and the system then presented five different options related to bullets.

Amy (becoming frustrated) "How am I supposed to know which one to choose? Are you sure you don't want to tell me how to create bullets?"

I believe that similar encounters are common between users and existing help systems. Amy's prior experience with other word processors and help systems lead her to believe that if she could not think of the **right** term, she was not going to receive much help. Even after I told her the correct keyword, she was frustrated by the system's inability to accept and respond to her real question. After several encounters like this one, users often decide that using the help system is not usually worth the effort. If help systems are to fulfill their intended role, they must accept and respond appropriately to questions like the one Amy posed to me.

1.4 Research Goals

The primary goal of this research is to construct an intelligent, helpful, and useful help system. The work investigates the following questions.

- **What are the properties of the questions users want to ask?**
- **What system functionality do users expect?**
- **What do they expect the interaction with the system to be like?**
- **Once a question has been submitted, how accurately can it be answered?**

The development of the help system, SAGE, is described as it progressed through the stages of data collection, implementation, and analysis. The data collection effort solicited spoken help requests and the analysis of the users' questions and expectations of the system were based on this data. However, SAGE works only with questions in text format. Consequently, most of SAGE's circuitry is applicable to written natural language questions as well. A goal during the development

[1] Of course, Amy is not her real name.

of SAGE was to combine technologies from the information retrieval, natural language processing, and machine learning communities to process and answer users' questions.

The system is not expected to handle all questions correctly. The goal is, minimally, to answer simple and common questions correctly. Simple questions are unambiguous, context independent, and directed at discovering how to accomplish a specific task within the application. Simple questions are not required to use the same vocabulary as the application for referring to a particular application feature or task. Some examples of common questions are those that one might find in a Frequently Asked Questions (FAQ) file. An FAQ file is a collection of the questions (and answers) asked most often within a particular community of computer users, hobbyists, or people with similar interests. Fortunately, most of the questions collected during this research fall into this category.

As alluded to in the research questions given above, two nongoals of the work are speech recognition research and automatic text generation. Consequently, the speech recognition component of SAGE is currently provided by a human experimenter. The answers provided by SAGE are canned (static) text (Fox, 1988). In this work, the existing help text database is used as source material.

1.5 Research Overview

The work presented here is based on an empirical research methodology. The development of the system and theories presented here is guided by data collected from real users with real tasks. That data is used to evaluate the system at each stage of its construction and to suggest further improvements.

The data consists of a corpus of questions collected from users trying to perform various word processing tasks. The questions arose from the users' need for help in completing the assigned tasks. Subjects outfitted with a microphone were permitted to ask for help from "the system." Subjects believed they were speaking to the computer, but an unseen person listened to and answered their questions. Each question was collected to form the corpus on which the development of SAGE was based.

The corpus was analyzed to learn about the types of questions asked. On average, subjects asked very short questions, some of which were not complete sentences. While a few users needed definitions of terms or wanted to know why something had occurred, most of the questions asked about how to complete a task. A small percentage of the questions were compound; they asked about two tasks in the same request. An important result was the users' willingness to make context-free requests. That is, they did not expect the system to keep any history of or make inferences based on the questions they had already asked.

These questions guided the development of SAGE by forming a test corpus on which each iteration of the system was evaluated. At the core of SAGE is an information retrieval engine based on Latent Semantic Indexing (LSI) (Deerwester et al., 1990; Dumais et al., 1988). It indexes the help text documents and returns them ranked in order of how similar they are to a submitted question. Performance of the system is based on how frequently a top-ranking document answers the question.

Standard information retrieval techniques for processing queries (questions) and the document collection (help texts) were tested. Many of these failed to improve the system's performance because the goal of information retrieval is different from that of answering questions (perhaps subtly so). Information retrieval is typically concerned with breadth of coverage. It seeks to find all the documents that discuss a topic. In question answering, it is only necessary to find enough information to answer the question. In the help collection used, this was usually possible with a single document.

Improved performance was gained by adding simple natural language processing and machine learning to the system. SAGE identifies the verbs and nouns in the question and uses them to rearrange the order of the top-ranked documents produced by LSI. For example, a question about creating tables may result in an ordering of the collection such that most of the top-ranked documents describe various operations on tables. However, due to the statistical nature of LSI, a document that uses the word **table** in comparatively large quantities may be judged as the most likely to answer the question. This can happen even though it describes an operation other than creating a table. Identification of the verb in the question permits SAGE to reorder the table documents by judging how well they discuss a creation action.

Further gains were made by eliminating from the ordered set those documents whose purpose did not match the goal of the question. Documents from the glossary that define terms are obviously inappropriate for answering a "How do I ..." question, but they are most likely to answer a "What is a ..." question. Finally, dramatic improvement was obtained by allowing SAGE to learn vocabulary from users. The help collection was augmented with some of the users' questions and the results tested for the remaining questions.

A validation of the techniques embodied in SAGE was made with a new set of users and their questions. Results from this experiment showed that SAGE's performance was comparable to that obtained with the first corpus of users' questions.

1.6 Speech vs. Typed Input

Although speech recognition is not part of the current research plan, it is appropriate to give some reasons why I believe this research should be pursued with speech input rather than typed input. Certainly, there are potential problems and drawbacks to a speech interface. Recognition errors frequently occur and speech interfaces may be disruptive in a crowded office setting. However, I believe speech to be an easier interface to use compared with the keyboard, particularly for novice users (Martin, 1989).

The subjects in the data collection study described in chapter 3 found speech to be an agreeable mode of communication. Speech is also more efficient and sometimes, more accurate than typed input. In one study comparing speech and typed input, Martin (1989) found speech to be a faster, more efficient channel. Speech avoids potential spelling mistakes that can make parsing and interpretation more difficult. Even if speech recognition is not currently up to the task, technological advances in speech recognition are being made so quickly that our systems must be

prepared to handle speech input when it becomes available.

A final, important benefit of speech is the opportunity for hands-busy information access (Cohen and Oviatt, 1995). Though word processing is a task and environment that easily permit a user to use the keyboard, other tasks have information needs that must be met while the user's hands are busy. These tasks may be as common as driving or as demanding as surgery. Virtual environments with head-mounted displays hinder the normal hand-eye coordination and provide a compelling motivation for speech-based information access. Additionally, public kiosks are becoming a common source for information and a speech interface may be more desirable than exposing a keyboard in such a public interface. While word processing may not fully realize the hands-busy benefits of speech, the research provides insight into spoken information requests that is applicable to other domains.

1.7 Overview of the Dissertation

Chapter 2 summarizes some related work in the various research fields that this work incorporates. The initial data collection experiment is described in Chapter 3. This chapter also investigates the subjects' reactions to a prototype of SAGE. The corpus of questions collected is analyzed in Chapter 4. An introduction to the information retrieval (IR) component is given in Chapter 5. This chapter also describes the primary evaluation metric applied throughout the development of the system. Chapter 6 extends the core IR capabilities by testing additional techniques commonly used within the IR community. Further enhancements to SAGE are made in Chapter 7 where ideas from the natural language processing and machine learning communities are used to build lightweight, plug-in components that massage the input and output of the IR engine. Chapter 8 evaluates SAGE in a live user test. General observations and lessons learned from the experiment are also given. Finally, Chapter 9 summarizes the contributions of the work and offers some suggestions for further research.

CHAPTER 2

RELATED WORK

The purpose of this chapter is to provide a framework for understanding how this research fits within various communities and to provide some background necessary for understanding the research. This is a daunting task due to the number of communities related to this work. Consider the events that occur when our hypothetical user, Amy, needs help with her work. She is working along in her word processor until she needs to accomplish a certain task that is unfamiliar to her. She asks her intelligent help system how to do the task. To answer her question, the system must perform the following actions.

(1) Convert the signal of her spoken question to text.
(2) Analyze the content of the question.
(3) Find an answer to the question and present it to her.

The process involves speech recognition, natural language processing, and information retrieval. Machine learning may be included in the construction of the intelligent help system. The system is built around the belief that asking for help from a friend is easier than using a traditional help system. Consequently, research on the interactions between users with task-specific problems and the experts who help them is also germane. Research from the user interface community also provides insight into the expectations and behaviors of users seeking help.

The information provided in this chapter is not meant to be a comprehensive coverage of any one of these research areas and should not be viewed as such. Its intention is to introduce some related research and provide references for finding more information. A brief introduction to various information retrieval principles and models is presented in Section 2.1. Question answering from the perspectives of information retrieval, natural language processing, and as a user interface is reviewed in Section 2.2. Finally, Section 2.3 gives an overview of some speech-based interfaces used in information access applications.

2.1 Information Retrieval Principles and Models

The information retrieval research related to this work starts with the premise that there is a collection or corpus of documents available in free-text form. For example, these documents could be journal articles, abstracts, books in a library, etc. These documents are to be indexed in some way such that people can submit queries and find documents relevant to their queries. Documents are indexed by words, commonly called terms. The indexer's goal is to choose terms that reflect the content and meaning of each document.

Early indexing systems required a set of keywords that were manually assigned to each document. This was a painstaking and time-consuming effort. Maintaining consistency in the index was also difficult if many people were involved in

assigning keywords to documents.

The tediousness of manual indexing soon gave way to automation. A major research effort at Cornell University was begun in the early 1960's that produced the SMART retrieval system (Salton, 1971; Salton and McGill, 1983a; Salton, 1989; Buckley et al., 1995). As of 1996, this system was still an active research project and it is widely used by researchers and practitioners around the world. Early work on the automatic selection and assignment of keywords was provided by (Sparck Jones, 1971).

2.1.1 Retrieval Models Three primary information retrieval models are of interest:

- the exact-match model,
- the vector space model, and
- the probabilistic model.

A more thorough treatment of these models can be had in any of (Salton and McGill, 1983a; Salton, 1989; Turtle and Croft, 1992; van Rijsbergen, 1979). However, the following summaries suffice as background for this research.

The Exact-Match Model The exact-match model has its roots in the early keyword systems. It is a simple boolean model that requires a document to satisfy the query parameters completely before being returned. The operators AND, OR, and NOT are used to specify the set of keywords that must or must not be present in the document. However, the use of boolean operators does not imply an exact-match model; only the treatment of the boolean operators as a boolean logic indicates an exact-match model. For example, the boolean query **natural AND language AND processing** could be handled by a less restrictive model that returns documents with fewer than all three terms present. A document indexed by two of the terms might be scored lower than one with all three terms, but it could still be returned. However, in the exact-match model, a document must be indexed by all three terms to be returned. Additionally, the documents returned are not ranked by this model.

While many commercial systems exist that are based on this model, it is not used in active research. The other two models both rank documents based on how "close" they match the query. As such, both can accept unrestricted queries and they do not require all query terms to be present in a document for it to be returned.

The Vector Space Model The vector space model represents terms and documents in a high-dimensional space. The high-dimensional space is constructed as a term-document matrix. Each index term occupies a row in the matrix and each document in the collection is represented by a column. The value of each matrix cell is computed as a function of the relationship between the term and document that correspond to the row and column of the cell. Many functions can be used (Salton and Buckley, 1988), but for simplicity's sake, assume the value is just a count of the number of times the term appears in a document.[1]

[1] A more complex (and realistic) function is described in Chapter 5.

Each document can now be thought of as a vector in an **n**-dimensional space, where **n** is the number of index terms. The vectors project along each dimension according the corresponding term's frequency within the document. Queries can also be projected into the space. Each query term has a corresponding vector in the space. Adding the appropriate term vectors together gives a new vector that represents the query. With the query vector projected into the document space, similarity comparisons can be made between the query vector and each document vector to compute an expected relevance to the query for each document. The comparison most commonly used is the cosine between vectors. This gives a real value from -1.0 to 1.0 by which each document in the collection is ranked.

The Probabilistic Model The probabilistic model attempts to compute the probability of relevance for each document in the collection. The probabilities are usually learned from a training set of queries and the documents relevant to them. A complete description of the probabilistic model is beyond the scope of this overview, but see (Fuhr, 1992) for a recent survey.

The INQUERY system (Turtle and Croft, 1990) is a unique type of probabilistic model, called an inference network model. It is based on Bayesian inference networks. An inference network is a directed, acyclic dependency graph in which nodes represent propositions and edges represent the dependence relations between those propositions. The network consists of a sub-network for the documents and another for the user's information need. The formalism permits multiple queries to be treated as different evidence of a single information need that can be remembered and combined to produce each return set. The formal power of the model is such that it subsumes the other two models (Turtle and Croft, 1992); however, the model has not yet proven superior in empirical tests (Sparck Jones, 1995).

2.1.2 Retrieval Evaluation There are many ways to evaluate the effectiveness of a particular retrieval system (Salton and McGill, 1983b; Frakes and Baeza-Yates, 1992). Some methods are interactive. These include user perceptions (Su, 1992; Smithson, 1994) and operational evaluations (Robertson and Hancock-Beaulieu, 1992). The drawback to using these methods is the requirement of involving subjects or complex tasks in the evaluation. Consequently, most systems use automated performance measures.

The automatic evaluation of a retrieval system is based on a set of documents determined to be relevant to a particular query. A collection of queries with their associated lists of relevant documents makes up a test set. Due to their simplicity and interpretability, the most commonly used evaluation measures are precision and recall (Salton, 1992). Recall is the percentage of all relevant documents returned. Precision is the percentage of returned documents that are relevant. Since these two measures are dependent on one another, most systems report their precision level at various levels of recall to give a measure of performance. Robertson (1969a; 1969b) provides a compelling theoretical argument for using fallout rather than precision. Fallout is the ratio of nonrelevant documents retrieved to the total number of nonrelevant documents. However, interpreting the results using fallout is difficult and so precision is typically used.

In 1992, the National Institute of Standards and Technology (NIST) instituted TREC (Text REtrieval Conference), a forum for the evaluation of leading research and commercial retrieval systems. The participants compete with the same collection of texts and queries to evaluate which models and techniques work well and which do not. Descriptions of the various systems and the results of this competition are published each year in conference proceedings (for example, see (Harman, 1995a; Harman, 1995b)). Usually, the newest and best information retrieval techniques are reported at this conference. A few of these are described and tested as part of the development of SAGE in Chapter 6.

2.1.3 Combining with Natural Language Processing Efforts to integrate natural language processing (NLP) techniques with information retrieval (IR) are becoming increasingly common. The efforts have been made for both query processing and index creation. Natural language processing in information retrieval systems is being done at the lexical, syntactic, semantic, and discourse levels (see (Smeaton, 1992) for an excellent summary). Recent work includes disambiguating different word senses in the index (Sanderson, 1994; Voorhees, 1993), finding synonyms automatically to augment queries (Chen, 1994; Grefenstette, 1994; Haas and Jr., 1994; Broglio et al., 1995; Lu and Keefer, 1995), and trying to identify the topic sentence in a paragraph (Oddy et al., 1992; Fukumoto, Suzuki, and Fukumoto, 1997) or document (Lin and Hovy, 1997) automatically. Topic sentences are then weighted more heavily during the construction of the index. Syntactic parsing of the collection to find head-modifier phrases for use as index "terms" (Jing and Croft, 1994; Strzalkowski, Carballo, and Marinescu, 1995; Smeaton, O'Donnell, and Kelledy, 1995; Strzalkowski, 1994; Strzalkowski et al., 1997; Zhai, 1997) is also popular.

The results from applying these NLP techniques to the collection index are mixed and not all the improvements have been as significant as expected. For example, (Strzalkowski et al., 1997) found that using phrases as indexing terms produces only modest improvements. However, using phrases to identify similar sentences within an initial document return set and then expanding the query with those sentences significantly improved retrieval performance. In contrast, (Zhai, 1997) reports an increase of 18% in precision by indexing with noun phrases. Korycinski and Newell raise some issues in applying NLP techniques to book indexing (Korycinski and Newell, 1990). However, their basic argument is that deep understanding is necessary to index a book effectively and deep understanding is not feasible to do with current NLP technologies.

An example of a more elaborate approach is the FERRET conceptual information retrieval system (Mauldin, 1991). FERRET performs a robust, partial parse of its input texts and stores them in canonical case frames. Queries are also parsed, converted to a case frame representation, and matched against the frames stored for each document. The system also uses a machine readable dictionary to increase its lexical knowledge and a learning component based on genetic algorithms to extend the available case frame patterns. FERRET was compared with a standard boolean keyword system using just 22 sample queries. Recall performance using FERRET more than doubled from 19% to 52% and precision increased from 35% to 48%.

Some common information retrieval techniques may also be viewed as weak

natural language processing techniques. A common technique applied in information retrieval systems that fits this description is stemming. It is the process of reducing variations of a word to the same morphological root. For example, the words **runs** and **running** would both be reduced to **run**. This reduction seems desirable since the words are semantically similar and their presence in different documents provides some evidence that the two documents share certain semantics. In practice, however, stemming does not always help (Harman, 1991).

2.2 Question Answering

The previous section described some general information retrieval models and techniques. This section looks at question answering, a specific type of information retrieval. The two fields have many overlapping goals and frequently, they use the same methods, but typically, users' of a question-answering system have a more narrowly focused information need. Traditional information retrieval applications try to return a high percentage of the relevant documents. However, in question-answering systems, users only need an answer to their questions, not all the information available on the topic.

The question-answering literature is examined from three points of view: indexed information, natural language processing, and user-expert interactions.

2.2.1 Indexed Information Indexed information systems are information retrieval systems whose goal is question answering. Systems that fall into this category are given their own treatment here, apart from the systems discussed in Section 2.1. Frequently, performance of these systems is measured a little differently than using the standard recall and precision measures of the information retrieval world. The measure of interest in these systems is whether the user obtained a satisfactory answer or not.

Chaucer Keene (1987; 1991) built a retrieval system called Chaucer and tested it with users looking for answers to questions in an indexed document collection. The system was designed to accept unrestricted, typed English queries and locate relevant documents (or nodes) in a hypertext network.

Chaucer used its own, unique method for selecting index terms. The index terms, called minids, were substrings of words (beginning with the first letter in the word) that occurred more than once in the texts. This method of selecting index terms is similar to stemming in some respects, but it lacks a morphological basis.

Chaucer was tested with subjects solving a set of problems (presented as text with some auxiliary figures) in a particular domain. The subjects were instructed to use Chaucer to find the information they needed to solve the problems. A set of relevant target documents was identified for each problem and retrieval success was measured against this set. One limitation of the methodology was revealed by Keene in the analysis. Most of the subjects' queries were simply paraphrases or repetitions of the question they were assigned to answer. Consequently, the language of the original questions strongly biased the results of the study.

Keene identified two measures of correctness: query correctness (at least one target document is returned) and recognition correctness (subject identifies at least one target document as relevant). Keene reported a query correctness rate of

88% and 73% for recognition correctness. It should be noted that users were allowed unlimited individual queries per problem and the rates were calculated by treating all queries as a single unit. In other words, if any of the individual queries succeeded, the entire problem was scored as a success. Unfortunately, the study failed to report the average number of queries per problem and the average number of individual queries that succeeded.

SuperBook Bellcore's SuperBook project (Landauer et al., 1993) was a study in search time and accuracy for electronic vs. printed text. The system combined search capabilities with hypertext to improve access to on-line texts. SuperBook accepted hierarchically organized texts in many markup languages (e.g., troff, LaTeX, MS Word, etc.), retagged them for its own browser, and created a full text index and a dynamic, fisheye Table of Contents.

Multiple word queries were posed either by selecting words from the text or by entering them from the keyboard. An exact word match was used to identify relevant documents and the dynamic Table of Contents showed the sections with relevant documents. Evaluation of the project found that using the SuperBook electronic texts, subjects achieved better accuracy and speed in their searches than subjects who used the same material in printed form. The accuracy rate for SuperBook users was roughly 78%. Users of the printed book achieved a 64% accuracy rate in their searches.

HELGON A different type of information retrieval system designed to support cooperative problem solving was embodied in HELGON (Fischer and Nieper-Lemke, 1989). HELGON indexed information by categories, items, and attributes within a knowledge base. The system helped guide its users to formulate their questions about this large body of information by restricting the allowed inputs to the categories and attributes known to it. Users could progressively reformulate their query in a top-down fashion by making selections from the category hierarchy displayed on the screen. Alternatively, they could work bottom-up using a sample matching item the system presents. Users criticized the example to express attributes that were inconsistent with their information needs.

MURAX The MURAX system (Kupiec, 1993) was designed to answer general knowledge questions from an encyclopedia. These are questions answered with a noun phrase rather than a procedure description. Noun and verb phrases were extracted from the question and used as query terms. The query was submitted to a boolean information retrieval system that had an index of encyclopedia documents. All documents containing one or more of the query terms were ranked by degree and number of matches. Answer hypotheses were extracted from these documents by finding all simple noun phrases in them. Each phrase was also ranked based on a combination of the ranks of the documents in which it was found. Then, the answer hypotheses were combined iteratively with phrases from the question to make new queries. The new queries were submitted to the system and the returned documents were searched for lexico-syntactic patterns that validated one or more of the answer hypothesis phrases as a type or instance of the phrases found in the question. For example, a question beginning with the word **who** wants a person as an answer. Therefore, the search process tried to find answer phrases that appeared in lists

of people or that were specifically identified as people. Finally, the highly-ranked phrases that meet this criterion were returned as answers.

2.2.2 Natural Language Processing The NLP approach to question answering has usually been to produce some deep understanding of the question to identify or infer an appropriate response.

Early Systems Question-answering systems were some of the first natural language processing systems. In the 1960s, systems such as BASEBALL, SAD-SAM, and SIR did simple inferencing to answer questions about the information they contained (Barr and Feigenbaum, 1982). For example, given the information that Bill is John's father and John is Jane's brother, these systems were designed to answer the class of questions that include, **"Who is Jane's father?"** More complicated inferences were also supported. The LUNAR system differed because it did not try to store information for inferencing. It translated English questions to a database query language, submitted the query to the database, and returned the results (Woods, 1970). Other natural language interfaces to databases are covered in (Sap and McGregor, 1992).

Lehnert (1978) was the first to develop a comprehensive taxonomy of question types as part of the development of QUALM. QUALM was a question-answering system that worked in conjunction with the SAM and PAM (Schank and Abelson, 1977) story analyzers to answer specific questions about the events in a story. Lehnert argued that before a question could be answered properly, it had to be understood. Part of this understanding required the question to be parsed and placed in one of 13 conceptual categories. The conceptualization of the question allowed QUALM to answer with a pragmatically correct response. For example, the question **"Do you have the time?"** could be answered literally as a yes/no question. However, the speaker's intention was probably to ask what time it currently was.

The early NLP-based question-answering systems were not help systems. They were designed around story understanding. They analyzed a story and answered questions based on the analysis of the story and question. In the 1980s, the community began developing help systems to answer questions related to problem solving and task completion. The technology continued to focus on producing a detailed analysis of the questions.

The UNIX Consultant A representative natural language question-answering system developed during the 1980s was the UNIX Consultant (UC) (Wilensky, Arens, and Chin, 1984; Wilensky et al., 1988). UC allowed novice users to communicate with the UNIX operating system in ordinary English by typing questions at the terminal. It was not designed to replace the UNIX command interpreter; instead, it was to be used as a tool to aid new users in learning UNIX commands.

UC processed a question by running it through a series of language analyzers, goal analyzers, and planners. An expression component produced its responses. Extensible knowledge bases of UNIX and English were critical to this process. These components allowed UC to work well for the domain covered by the knowledge bases, but building and maintaining them was too difficult and time-consuming to be practical for building commercial software. However, the system designers stated that employing the system in a real-world setting was not a criterion for success.

OSCON OSCON was another natural language question-answering system in the UNIX domain. It was similar to UC in the sense that it was a knowledge-based approach to the problem of answering natural language questions. OSCON's development, however, differed from UC's in that two Wizard of Oz experiments (McKevitt, 1990; McKevitt and Ogden, 1990) were run to collect a corpus of questions that real users would ask. The Wizard of Oz technique (generally credited to Gould and his associates (Gould, Conti, and Hovanyecz, 1982)) was used to collect a corpus before building the system.

A person simulated the expected function(s) of the system to collect realistic data. The collected corpus was used to drive the development and evaluation of the system being built. UC's examples were, for the most part, "invented" by the system's designers. The examples seem plausible enough, but the researchers were only guessing that someone might ask the questions the system was being designed to handle. A corpus collected for the target domain provides insight into the range of capabilities the system must possess. Examination of the OSCON Wizard of Oz experiment logs[2] revealed that approximately 87% of the users' questions were simple or common (i.e., questions an automated system would be expected to answer correctly). For other examples of Wizard of Oz experiments and their use, see (Brunner et al., 1992; Fraser and Gilbert, 1991; Guindon, 1988; Guindon, 1991; Hauptmann and Rudnicky, 1988; Maulsby, Greenberg, and Mander, 1993; Salber and Coutaz, 1993).

The OSCON Wizard of Oz experiments further showed the importance of properly designing the experiment to avoid influencing the subjects. The first experiment used English descriptions of the tasks and the structure and vocabulary of the corpus paralleled that of the task descriptions. This result is similar to Keene's finding during the evaluation of Chaucer. A second experiment was designed using pictures to show the tasks to be completed. This approach avoided biasing the subjects with the experimenters' knowledge and familiarity of the task.

PORSCHE The approach taken by PORSCHE (Pilkington, 1992a; Pilkington, 1992b) was to customize help responses to the needs of the users of an email application. A dialogue component tracked the user-system discourse and built a profile of the user's knowledge. It used the profile to resolve ambiguity about the user's goals. PORSCHE was also tightly integrated into the application so it could monitor the program's state and use that information in forming its responses.

PORSCHE generated responses based on a detailed rhetorical structure (Mann and Thompson, 1987) model. The model was based on an analysis of questions asked by subjects in a Wizard of Oz study. The complexity of the required model was held in check by limiting the scope of the questions that could be asked. Users could ask questions only by composing them from one of three question types available on a menu and a list of topics presented in the help browser. The users could give very focused questions because the question types available corresponded directly to the rhetorical structures available.

Other NLP Systems Additional research on answering questions and

[2] I want to thank Paul McKevitt for sharing the OSCON logs with me.

providing advice with typed natural language interfaces has focused on discourse understanding and planning. Carberry and her colleagues are studying schedule planning dialogues between students and advisors. From their observations they are developing a model of discourse (Carberrry, 1989; Lambert and Carberry, 1991; Chu-Carroll and Carberry, 1994). The TRAINS project (Allen, 1991; Allen and Schubert, 1991) is an effort to develop an intelligent assistant for scheduling a fleet of trains. The research goals of the project include investigation of real-time plan reasoning and monitoring man-machine dialogue (particularly mixed initiative dialogues and those with a high incidence of discourse-level phenomena). Parsing of "real" utterances (both spoken and typed) is another of the project's goals.

2.2.3 User-Expert Interactions The user interface community has studied problem-solvers interacting with experts or tutors to gain insight into how help systems might better emulate an expert. Aaronson and Carroll (1987b) set up an experiment in which users could pose questions to experts through email. They analyzed the questions asked and the strategies employed by advisors in answering the questions. The questioner and answerer typically expected only a single interaction and thus attempted to anticipate follow-up questions or problems and solve them by providing sufficient information. The one-shot constraint limited the complexity of the queries as compared with another study of face-to-face help sessions. The advisor attempted to prevent the need for further questions by making assumptions about the user's goals, providing alternative solutions, explaining how the system works, and pointing to reference sources.

In another study, (Aaronson and Carroll, 1987a) studied 30 face-to-face advisory sessions. They found that 76% of the users' questions were verification questions, essentially a question like "Is this idea correct?" In other words, the users were suggesting a solution and just asking for verification from the advisor that the idea was sound. The authors suggested that an intelligent help system could take advantage of this information by extracting the proposed solution and using it while answering the question.

In contrast to (Aaronson and Carroll, 1987a), Hill and Miller (1988) found that only 8% of the questions asked in their study were verification questions. The difference in the studies was the communication medium. In (Hill and Miller, 1988), the subjects talked with the advisor through a computer terminal rather than face-to-face. In the experiment, subjects were given tasks to be completed with a statistics application. In three experimental sessions, the advisor could see the subjects' display and in the other three, this view was unavailable. Subjects submitted a total of 84 questions, 51 (61%) of which were requests for plans to achieve some task-specific goal. The advisor was asked to describe or identify a system object in 17 (20%) of the questions and the remaining nine (10%) were miscellaneous requests.

In the "full screen" sessions, the advisor used knowledge of the subjects' prior interaction with the system in 45% of the responses. The advisor's knowledge of the current screen state was required for 67% of the responses and 34% used both pieces of information. In the "no screen" condition, 40% of the exchanges consisted of more than one question/answer pair (while only 13% were a single pair in the "full screen" condition). Furthermore, 62% of the multiple-query exchanges addressed

information available on the user's screen.

Most of the advisor's answering strategies were similar to those found in (Aaronson and Carroll, 1987b). However, one difference in strategies was concerned with monitoring the subjects' high-level goals. The advisor verified that the subjects' questions and the low-level goals implied by those questions were compatible with their more important goals. If the advisor believed that subjects' reasons for using requested information were potentially harmful or not related to the current goal, the subjects were warned or the requested information was simply withheld. Pollack (1985) also found that advisors do not answer the exact question asked when it would impede progress toward the perceived top-level goals of the user.

Further study of these interactions (Hill, 1989) revealed that subjects followed prescriptive advice effectively and efficiently in just over half the cases. The primary reason for the failures was a faulty supposition of the subjects' knowledge by the advisor. When the advisor supposed that the subjects knew something about the interface that they did not, the advice was not correctly understood by the subjects.

2.3 Speech-based Interfaces

Talking to a computer has been envisioned for many years, and not just by AI researchers. Movies (e.g., 2001: A Space Odyssey) and television programs (e.g., Star Trek) have depicted voice communication with computers for decades. However, successful speech recognition systems have only recently begun to be available (Meisel, 1993).

2.3.1 Speech Recognition Parameters Speech recognition systems are characterized by three variables:

- speaker dependent vs. speaker independent,
- discrete vs. continuous speech, and
- vocabulary size.

Speaker dependent systems require a training period for the system to adapt to the speaker's voice, while speaker independent systems are intended to be used by anyone without training. Discrete speech systems require the speaker to separate each word with a short pause to help the system identify word boundaries easily. Continuous speech systems permit users to speak as they would in normal conversation. State-of-the-art systems support vocabularies of a couple thousand words for speaker-independent, continuous speech and up to 50,000 or more words for a speaker dependent, discrete speech system.

The spoken help requests collected and examined in this research are expected to come from a speaker independent, continuous speech system. Although speaking with discrete speech and training the recognizer improve the recognition accuracy, they are barriers to using the system. Users faced with a problem completing a task do not need to face additional problems learning to speak unnaturally nor do they want to take an hour or two to train the help system to recognize their voice. Consequently, the systems reviewed here are limited to speaker independent, continuous speech systems.

2.3.2 Spoken Language Systems Most of the research on spoken language systems is being conducted with systems designed to respond to information

requests different from the types of help requests examined in the previous section. Primarily, these systems provide speech interfaces to relational databases.

ATIS The majority of the research in speech recognition systems is being conducted in connection with the ARPA Spoken Language Systems program (Hirshman, 1994). Systems are evaluated on the Air Travel Information System (ATIS), a database of airline, schedule, and ground transportation information for nine cities. The major participants in this effort are MIT (Glass et al., 1995; Zue, 1994; Zue et al., 1991), CMU (Ward and Issar, 1995; Issar and Ward, 1994), BBN (Miller et al., 1995), and SRI International (Cohen, Rivlin, and Bratt, 1995; Moore et al., 1995). While the implementation details of each system differ, on a larger scale they each work by combining a speech recognizer with a natural language understanding (NLU) system. The NLU systems help the recognizers eliminate syntactically or semantically implausible sentences from the list of possible utterances. Discourse management facilities in the systems are necessary to track the conversation between user and system. Some systems can also initiate parts of the conversation when the user is ready to schedule a flight. The systems currently can process and answer correctly almost 90% of the spoken input given them (Hirshman, 1994).

BeRP The BeRP (Berkeley Restaurant Project) system (Jurafsky et al., 1994) is described as a knowledge consultant like the ATIS systems rather than a help system like UC and OSCON. The system contains a relational database of restaurants in the Berkeley area. Users orally provide a set of dining criteria and the system suggests several restaurants for the user to consider. Since the current research is primarily focused on speech recognition techniques, BeRP attempts to understand just enough of the dialogue to fill out a database query template. The system can prompt the user for missing template information such as the desired type of food or acceptable price range. When the template is filled, the system queries the database and presents the results to the user.

OGI Some of the work being done at OGI (Oregon Graduate Institute) is concentrating on modeling difficult sources of linguistic variability to produce more user-centered and robust interfaces (Oviatt, 1996; Oviatt and VanGent, 1996; Oviatt, 1995). One goal is to produce guidelines for developing speech-based interfaces that reduce the number of spoken disfluencies[3] and help predict when they will occur. For example, one experiment found that lengthy utterances accounted for 77% of the variability in spoken disfluencies (Oviatt, 1995). The data presentation was restructured to help guide the users' speech into shorter sentences and 60–70% of the disfluencies were eliminated.

[3] A disfluency is a spontaneous self-repair that interrupts the smooth flow of an otherwise coherent sentence.

CHAPTER 3

DATA COLLECTION

Proper design and construction of a spoken language help system requires a corpus of questions on which to base the system's design and evaluate its performance. The corpus provides a fixed test suite by which the effects of system modifications can be evaluated. Unfortunately, a realistic set of questions is best collected under real conditions. In other words, the data should be collected from real subjects with real tasks using a real system. Gathering the data with a real system creates a circular dependency since the reason to collect the data in the first place is to build and evaluate the system!

This dilemma can be overcome by simulating the desired system using a technique known as a Wizard of Oz (WOZ) experiment (Gould, Conti, and Hovanyecz, 1982; McKevitt, 1990; Dahlback, Jonsson, and Ahrenberg, 1993; Maulsby, Greenberg, and Mander, 1993). In a WOZ experiment, a human assistant, known as the wizard, simulates some portion of the system's functionality during the data collection effort. The data is then analyzed to determine whether it is practical and desirable to replicate in software the functionality that was provided by the wizard. In the experiment described here, the wizard provided the system's speech recognition and answer retrieval functionality.

3.1 Goals of the Wizard of Oz Experiment

There were two goals of the WOZ experiment described in this chapter. They were:

(1) To test users with a simulated system and make a subjective judgment about the utility of the proposed system based on their experiences and feedback.

(2) To collect a corpus of spoken questions to be used for building and testing the system.

3.1.1 Evaluate Reactions to the System The first goal of the WOZ experiment was to determine if subjects would find a natural language, speech-based help system useful. This investigation was important because some previous research questioned the utility of on-line help systems in assisting a user to solve problems encountered in completing a task (Mack, Lewis, and Carroll, 1983; Borenstein, 1985). The WOZ experiment hoped to establish whether subjects would avoid using a spoken language help system as they often do with existing on-line help systems.

3.1.2 Collect Spoken Language Data The second goal of the experiment was to collect information about the language (vocabulary, syntax, etc.) subjects would use and the interaction they expected from the system. Current speaker-independent, continuous speech recognition technology requires a grammar to guide the recognition process. Grammar construction could be based on the vocabulary and syntax exhibited by the subjects in the experiment. Development of

the system also required knowledge of the linguistic phenomena likely to be exhibited by the users. For example, if the users referred to specific objects on screen that the system needed to recognize or if they referred to prior questions or answers while asking a new question, the system must have the ability to handle this information properly.

Furthermore, the submitted questions were used as a test suite during system development. Answers were located within individual documents of the collection and the document identification was recorded with the question. This information was used to test how well the system was able to locate and retrieve appropriate answers within the collection.

3.2 Setting Up a Wizard of Oz Experiment

The domain chosen for the experiment and subsequent system development was word processing. However, every effort was made to keep the research as domain independent as possible. The experiment was run using FrameMaker, a commercial word processor running under the X/Motif window system. Consequently, the word processor itself could not be modified to permit communication between the subjects and wizard. A separate suite of software was developed to support communication between the two parties. This communication software had to support the needs of both the subjects and wizard.

3.2.1 Requirements In addition to the application software, the general requirements of the experiment included computer displays for both the subjects and wizard, a convincing speech-based help interface for the subjects, a mechanism for the wizard to hear the subjects, and recording facilities to capture the desired data. The subjects were isolated from the wizard during the experiment.

Additionally, the software for the WOZ experiment was developed with the following beliefs about the needs of the subjects and wizard.
The subjects needed:

(1) To easily and efficiently alert the help system that they wanted to ask a question.
(2) To be able to review the help system's response as they attempted to execute any directions given in the response.

The wizard needed:

(1) To distinguish between real subject questions and other utterances (e.g., when the subjects were talking to themselves or to the facilitator).
(2) To respond quickly and accurately to both common and novel questions.
(3) To respond consistently to semantically similar questions.
(4) To see the contents of the subjects' screen.

The last item in the list of wizard needs was to permit the wizard to respond correctly in the event the subjects referred to a specific part of the text or a specific object or interface component. The subjects might have referred to an object either by pointing with the mouse or by a verbal description of the object. Consequently, both situations had to be accounted for in the development of the wizard's communication interface.

3.2.2 Hardware Components The WOZ experiment was set up in adjoining offices. The subjects worked in a private office and the wizard was stationed in an adjoining lab. In addition, a facilitator was present with the subjects to assist when necessary and to supply the subjects with tasks. The experiment was performed without special purpose computer, audio, or video equipment. The subjects and wizard used monochrome X terminals on the local network as their display device.

Audio Subjects were outfitted with a microphone. They were instructed to speak normally when asking questions and the microphone would pick up the sound signal. A cable from the microphone ran to the back of the computer to give the impression that it was connected to the machine. From there, the cable was strung with the network cables into the next office. The microphone cable was plugged into a portable cassette recorder which also broadcast the sound signal to the headphone port while recording. With this configuration, the wizard could use headphones to listen to the subjects' questions while the subjects' speech was recorded directly. Recording all the subjects' speech (as opposed to just the questions submitted to the help system) proved to be useful during the data evaluation because the subjects often expressed some of their thoughts out loud while interacting with the system.

Video All sessions of the experiment were video taped by mounting a video camera behind and off to one side of the subjects. The camera was directed only at the computer screen. The resulting video tape was of fairly low quality due to the distance of the camera from the computer and the flicker of the computer display produced by its refresh cycle. The on-screen text was mostly illegible; however, the video clarity was sufficient to detect the subjects' actions within the interface.

3.2.3 Software Components Two pieces of software were developed to meet the needs of the subjects and wizard. The software components were named **xwoz** and **xspy**.

xwoz The first piece of software, **xwoz**, supported most of the communication between the subjects and wizard. It also logged the actions taken by the subjects with the help interface and kept a log of all of the wizard's responses. **xwoz** was designed to be reusable for many different WOZ studies. At start up, **xwoz** presented a dialog box that allowed the wizard to specify the name of the subjects' X display, the subjects' identification number (for logging purposes) and the name of a help text file. After the requested information was entered, **xwoz** opened the log file, loaded and processed the specified help file, opened a connection to the subjects' display, initialized the communication windows, and launched the **xspy** application for watching the subjects' display. These tasks are discussed in more detail below.

The **xwoz** interface consists of the main window on the wizard's display and help request and response windows on the subjects' display. The main window is made up of a menu bar containing various help topics, a read-only, scrollable text region for displaying the canned help text, a writable text region for composing responses, and a status bar that also contains buttons for sending the help text to the subjects and clearing the text from the response composition region (see the top-most window on the right in Figure 3.1).

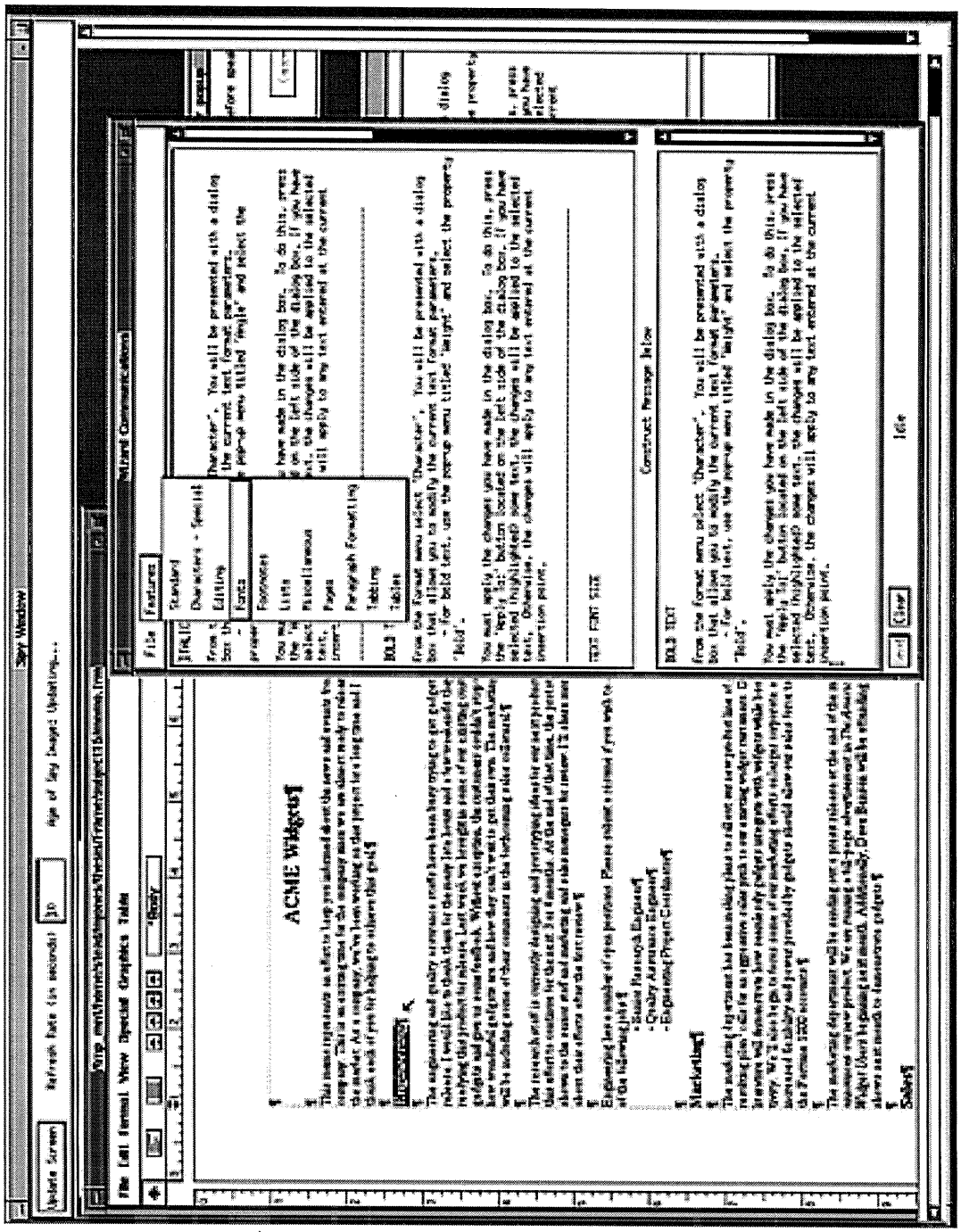


Figure 3.1. A view of the wizard's display showing the **xspy** program in the background and the **xwoz** response generation interface in the (right) foreground.

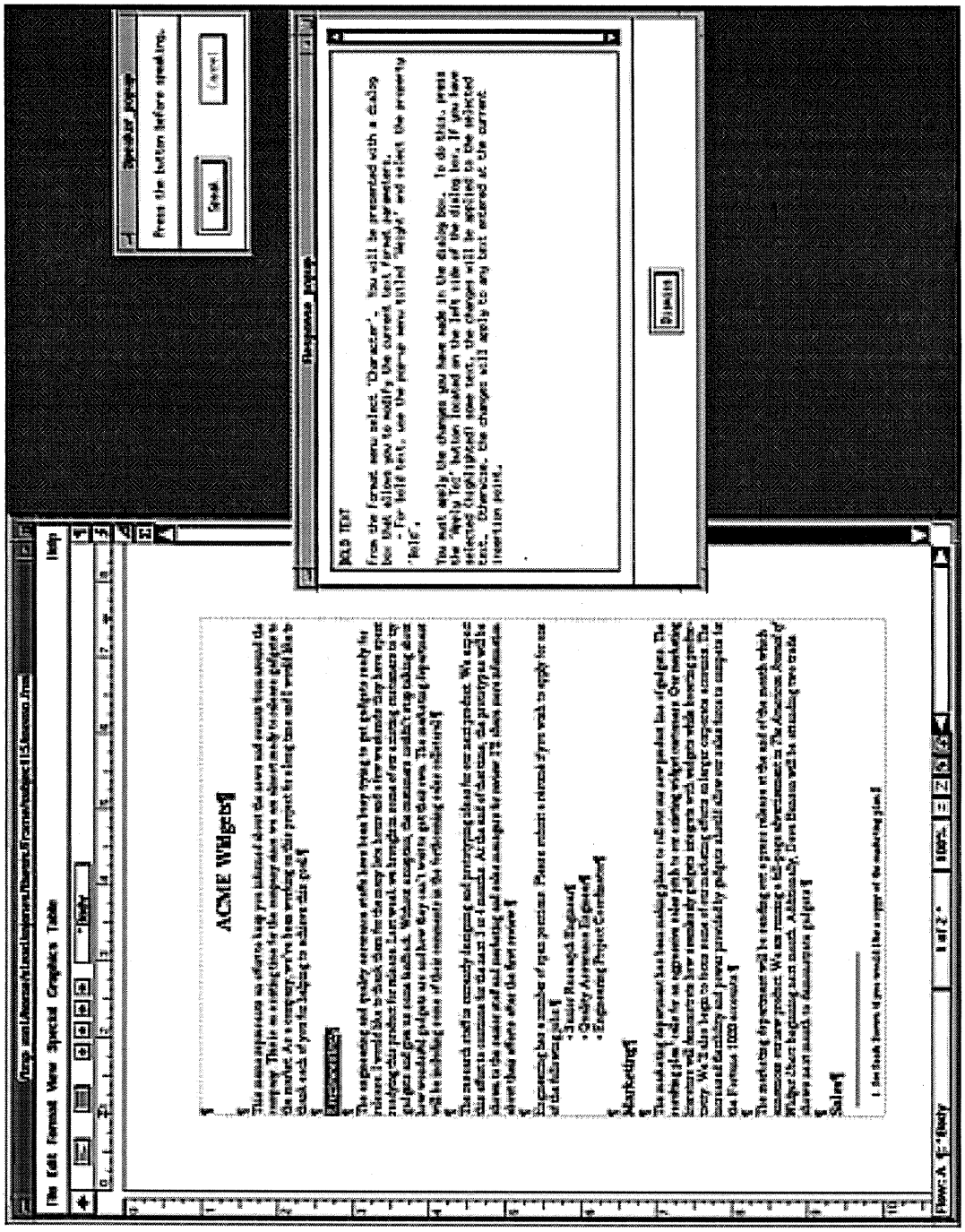


Figure 3.2. A view of the subjects' display showing FrameMaker on the left and the help request and help response windows on the right.

The help topics menu was populated automatically from the contents of the help text file. The file contained the canned text used by the wizard to answer most of the subjects' questions and it was easily modified to accommodate the addition, deletion, or modification of text needed by the wizard. The help text was organized using a simple markup language. The format of the markup tags was similar to the format used with SGML.

The details of the markup language are not important with the exception of the top-level tag, TOPIC. Each TOPIC block contained one or more pieces of help text that might be used to answer a question. The TOPIC blocks were organized to correspond to individual word processing tasks. **xwoz** scanned the help file when it was loaded looking for the TOPIC identifiers. From these it built a pulldown menu of topics. When the wizard selected a topic from the menu, the help text associated with the selected topic was presented in the main **xwoz** window.

The wizard could very quickly provide responses that were recorded in the canned help text file. The use of canned text also maintained consistency of responses for similar questions. Additionally, the wizard was free to compose a new response if the subjects asked a question for which there was not a canned response.

xwoz placed a help request window on the subjects' display (see the top-right corner of Figure 3.2). The window contained a button to request help and another to cancel the request. Only after the subjects pressed the **Help** button was the wizard's text **Send** button enabled. This design ensured two important things related to the consistency and realism of the interface. First, the wizard could not inadvertently press the **Send** button and generate a spurious communication on the subjects' display. Subjects would have seen this as a major glitch in the software and they may have become suspicious. Second, the wizard was not able to respond to valid questions unless the button was pressed. Because the wizard could hear everything the subjects said, there were times that the subjects asked a question without first pressing the button. It would have been up to the wizard to remember whether the **Help** button had been pressed (or else the subjects might begin to wonder why they sometimes got help without pressing the button). The wizard had enough to worry about without keeping track of this small, but important, detail too.

The final interface component was the help text window (the top-most display window in Figure 3.2). It popped up on the subjects' screen to display the response sent by the wizard. The window remained on screen with the most recent response so that the subjects could refer back to the help text while completing the assigned task. However, the window could be dismissed if the subjects wished to remove it from view. Then the next time the help system was used, the window popped back into view.

xspy **xspy** is a separate application used to watch another X display. There are other public-domain, spy-like programs available, but none met the requirements of this experiment. These programs either replicate only a single window or they were designed to be used in a classroom setting in which there is a master display and the others are all slaves (i.e., they do not permit the users to interact with the display contents). Spying on a single window was not sufficient for a

complex application like FrameMaker because it uses many different windows and dialog boxes. The master/slave configuration was not acceptable either because both subjects and wizard needed to interact with their own display.

xspy can display an image of any accessible X display within a resizable, scrollable window. The disadvantage of **xspy** is that it places a large burden on the network by requesting full screen snap-shots.[1] To lessen the network traffic, **xspy** contains an input field for specifying the frequency (in seconds) of each screen-shot request. In case a current snapshot is needed before the update timer expires, **xspy** also provides a button to request an immediate refresh of the image. **xspy** shows a (nearly) real-time image of the remote mouse cursor to allow the wizard to see where subjects were pointing.

The wizard's display in Figure 3.1 shows the reproduction of the users' display from Figure 3.2 in the background. The top portion of the window contains the controls described above. To the right of the highlighted text is a small arrow that represents the subjects' mouse cursor. As the subjects dragged the mouse around the screen, the wizard could see where the subjects were pointing or what object was being manipulated.

3.2.4 Sample Interaction The following simple scenario gives an idea of the actions involved in asking and answering a question within the context of the WOZ experiment configuration. Assume that the system has been initialized and a subject is now in the middle of a task in which the font of some text needs to be changed to bold.

(1) The subject decides to ask for help with the task. The mouse cursor is moved to the help request dialog and the **Help** button is pressed.

(2) The wizard's display beeps as a signal that a question is being asked, the **Send** button is enabled, and the status message in the **xwoz** main window changes to "Listening." **xwoz** also writes a time-stamped message to the log file indicating that the subject is requesting help.

(3) The subject speaks into the microphone and asks the question, "How do I make this text bold?" The question is recorded by the tape recorder while the wizard listens.

(4) The wizard selects the **Fonts** topic from the pulldown menu. The current contents of the canned text region are replaced with text explaining how to set various font and style properties.

(5) The wizard locates the section on "Bold Text," selects the answer text using the mouse, and pastes it into the response composition text region.

(6) Finally, the wizard presses the **Send** button. The button is disabled, the status message reverts to "Idle", the response is recorded in the log file, and the system's response is displayed on the subject's screen in the help text window.

[1] The snap-shots are generated as XImages.

3.3 Running the Wizard of Oz Experiment

Subjects in the WOZ experiment were asked to complete a series of common word processing tasks. The WOZ experiment was performed in two phases. The first phase was used to test the experimental set up. The questions collected from subjects participating in phase one were discarded. The second phase of the experiment was used to collect the question corpus that was analyzed and used to develop the help system.

3.3.1 Phase One: Fine-Tuning During the first phase, five subjects[2] participated in the experiment with knowledge that a human wizard was answering their questions. However, the subjects were asked to treat the system as if it were completely automated. The primary goals of this phase were to find and fix bugs in the software and to collect additional questions that had not been anticipated. Answers to the new questions were added to the canned help text database.

In addition to the bugs fixed and additional help text generated, this first phase proved to be beneficial in evaluating the assigned tasks. The first two subjects found the initial task to be too difficult and thus, it was a discouraging beginning. The tasks were redesigned and tested with the other subjects. The task presentation format (described in the next section) was also refined to make it clear to the subjects specifically what task was being requested of them.

3.3.2 Experimental Tasks The experimental tasks were selected to be representative of some fairly simple and common tasks that might be done with a word processor. FrameMaker was loaded with a partially complete document and the tasks were all designed to modify the document in some way.

An important constraint in designing such an experiment is to present the tasks in a manner that avoids influencing the subjects' vocabulary and syntax (McKevitt and Ogden, 1990). To achieve this goal, each task was presented as "before" and "after" views of the document. For each task, a paper copy of the document as it should currently exist was created and another copy of what it should look like after the task had been completed. The relevant differences were highlighted to make them easy to discern. In this manner, subjects were able to produce a mental image of the goal and formulate any questions without being biased by a lexical description of the task.

There were a total of fourteen different tasks; however, each task may include up to three new subtasks. Many of the tasks utilized skills that should have been learned during previous tasks so the number of new subtasks was reduced. For example, adding a title to the document involved entering new text, centering it, and modifying the font size and style. However, when the subjects were given this task, they had already added text and changed font properties during an earlier task so there was only one new subtask. The fourteen tasks and their associated new subtasks are outlined in Table 3.1.

3.3.3 Phase Two: Data Collection The second phase of the experiment was run with seventeen volunteer subjects. The subject population consisted

[2] Most subjects in phase one were fellow computer science graduate students.

Table 3.1: Description of the tasks assigned during the Wizard of Oz experiment.

Task Description	New Subtasks
1 Change the font of a journal name.	• italics font style
2 Modify the appearance of section headings.	• font size • bold font style
3 Add a new sentence with a non-English character.	• text insertion • foreign characters
4 Add a title to the document.	• text justification
5 Add a footnote.	• footnote creation
6 Rearrange the order of the document sections.	• cut and paste
7 Create a list with bullets.	• indentation • bullet symbols
8 Add and populate a table.	• table creation • table heading row • cell editing
9 Center the table horizontally on the page.	• table justification
10 Adjust the width of a column in the table.	• column size
11 Right justify the contents of some cells.	• cell contents justification
12 Remove the default table title.	• table title appearance
13 Remove the top and left table borders.	• table border style
14 Add page numbers.	• page number creation • page number justification

of students from many departments[3] on campus, computer science department office staff, and members of the community. Seven of the subjects were women and ten were men. All of the subjects had from one to five years of experience with word processors, but none had any prior experience with FrameMaker.

Before beginning the experiment, subjects were given about five minutes of overview and instruction. Subjects read a short description of the experiment in which they were encouraged to ask the "system" for help when they needed assistance completing a task. The system was described as experimental in nature and not fully implemented. However, each of the subjects in this phase seemed to believe that they were speaking directly to the computer. Subjects were also encouraged to speak naturally, "as if you were asking a friend or co-worker for help." Subjects were then instructed in the use of the help interface and permitted to submit one or two (usually out-of-domain) questions to get an idea of how the system worked.

A facilitator was present during the experiment. The facilitator's role was to present the tasks (paper copies of the document) to the subjects one at a time. Each new task was given only after the subjects had completed the previous one. The facilitator was also present to assist the subjects in case they became completely stuck. For example, one subject forgot to press the help request button before asking a question. When the system did not respond, the subject became confused. The facilitator's reminder to press the button before asking a question allowed the subject to proceed with the task.

Subjects proceeded through the experiment completing as many tasks as possible in the allotted time. The subjects were not informed how many tasks they would be asked to complete. The only information they were given was that the entire session would last approximately an hour. This approach gave the facilitator the flexibility to shorten the experiment before subjects became too frustrated while still allowing them to believe that they had completed the entire experiment. None of the subjects completed all[4] the available tasks within the specified time. On average, subjects completed 9.4 tasks containing an average of 16.9 subtasks. The distribution of the number of tasks completed is shown in Table 3.2. During the

Table 3.2: Distribution of the number of tasks completed by subjects.

Total Tasks Completed	# of Subjects
8	6
9	3
10	6
11	0
12	1
13	1

course of the experiment, subjects asked a total of 274 questions of the system. The

[3]The students came from such diverse departments as chemical engineering, mathematics, astrophysics, geology, political science, advertising, Spanish, and education.

[4]Subjects were not expected to complete all of the tasks.

analysis of this corpus is reserved for chapter 4.

At the conclusion of the tasks, the facilitator conducted a short interview with each subject. The subjects were asked to reflect on their experience with the interface. A discussion of their comments is given in section 3.4.2.

3.4 Discussion

At the beginning of this chapter, two goals of the WOZ experiment were identified. The first goal was to evaluate users interacting with the system and get feedback from them. This section discusses some general observations made during the experiments and summarizes the feedback obtained during the postexperiment interviews. The second goal was to collect spoken language data. As mentioned, that data is analyzed further in chapter 4.

3.4.1 Observations The experiment was designed to allow subjects to interact with a fairly intelligent system. Speech recognition errors were not intentionally simulated, though sometimes the wizard could not understand what was said and was forced to ask the subjects to repeat a question. Also, the wizard was instructed to exhibit fairly good understanding. That is, the wizard was permitted to judge how much interpretation should be applied to a question. If the question was badly garbled or unintelligible, the subjects received a message that the system did not understand the question or that it was too vague. This response only occurred a total of 12 times during all 17 sessions. The system was also quite accurate and specific with its responses because the help text was written with a bias toward the specific tasks to be completed.

There is a trade-off in simulating an intelligent system as was done. The downside is that current technology does not allow such a well-behaved system to be built. Current technology is such that speech recognition errors are unavoidable. This experiment was not designed to evaluate how the overall system would be perceived in the context of reduced recognition accuracy. Also, canned, general help texts cannot be as focused as those used here and text generation technology is not advanced enough to provide similar responses. Furthermore, the wizard was permitted to draw on personal knowledge of semantics and the assigned tasks to interpret the questions, neither of which is available to a general-purpose help system.

On the other hand, simulation of an advanced system provides insight into the requirements of such a system. For example, the subjects did not try to engage the system in dialog (see section 4.2.2 for further discussion) in spite of the system's advanced understanding capabilities. Many of the subjects also used caution in asking their questions. At times, the subjects would pause, appear to collect their thoughts, and then ask a question. Many subjects acknowledged in the interview that they made an effort to limit their vocabulary so the system would understand them.

It is unclear if the presence of the facilitator had any effect on the subjects' behavior. At worst, the subjects may have been more careful while asking questions. One subject expressed embarrassment at saying something stupid to a computer. This feeling may have been a result of the facilitator being present to overhear the subjects ask questions. Some subjects occasionally spoke to the facilitator during

problem solving, but this probably had little effect on their questions and impressions of the experience.

One of the problems observed during the experiment was the delay in receiving a response after asking a question. Subjects also identified slow responses as the thing they disliked most about the system. Most of the delay was attributed to network load and the increased traffic caused by the screen images being sent across the network. This "feature" made the experience more realistic with respect to the behavior that might be expected in a more intelligent help system. Many of the subjects adapted to the delay in an interesting way. When given a new task, they immediately asked a question and then proceeded to try to complete the task on their own. If they completed the task on their own, they felt that little time had been wasted. However, if the system responded before they figured out how to accomplish the task, they turned to system's response and followed the directions given.

3.4.2 Interview Summary Feedback from the subjects about their experience was gathered during an interview immediately following the completion of the assigned tasks. The following questions were asked during the interview.

(1) How did the help system's usefulness compare with other on-line help systems you have used?
(2) How did the help system's responses compare with those you would have expected from a person (e.g., the facilitator)?
(3) How much English do you think the help system understands?
(4) Did you feel you had to be careful about how you spoke or what you said? If so, in what ways?
(5) How did it compare with your expectations?
(6) What were your expectations?
(7) What bothered you most about the help system or what was most frustrating about it?
(8) What did you like best about the help system?
(9) Do you think such a help system is useful? Would you use it?

Subjects' responses to these questions are summarized in the following paragraphs. Not all subjects responded to every question.

(1) Only eleven subjects responded to the first question. In comparing the experience with other help systems, three subjects made positive comments about the efficiency of the voice interface. Four subjects preferred being able "to just ask a question and receive a response without hunting through a list of keywords."

The other prevalent response (four subjects) was that the help text was easier to understand than most help text they had encountered. I believe this statement to be a general criticism of the poor quality of help text available in current commercial applications. It supports the findings of other researchers (cf. (Mack, Lewis, and Carroll, 1983; Borenstein, 1985; Lang, Graesser, and Hemphill, 1990; Rieman, 1994) Help text is not often written as if answering a question about how to accomplish a task. In fact, very little of the canned text used in this experiment was extracted from the existing documentation because it was so confusing that it did not actually help with performing the tasks either. The answers recorded in the

canned text database were a result of experimenting with each feature and recording the steps needed to use it properly.

(2) Subjects made two interesting observations when asked to compare the system with the interaction they would expect to have when asking a human for help. Six of the fourteen subjects who answered this question made the point that if an expert was present (as opposed to placing a telephone call to technical support), performance of the task could be shown or demonstrated. Many commercial applications have recognized this fact and they now include coaches or tutors that can drag the mouse cursor around the screen to demonstrate the actions that should be taken.

The second observation is that the expert would probably respond verbally. Six subjects indicated that they preferred the textual responses provided by the system. This format permitted the subjects to proceed at their own pace and to refer back to the instructions while completing each step in the task. They were concerned about needing to ask a human (or system that responds verbally) to repeat the instructions.

(3&4) The subjects' perception of the system's language capabilities were probed with questions 3 and 4. In general, subjects were surprised at the amount of English understood by the system, but they still felt that they needed to limit their vocabulary to certain keywords that the system could recognize. Only four of the seventeen subjects believed that the system understood everything and they made no attempt to limit the interaction in any way. Two additional subjects thought the system understood nearly everything. They said that they started out trying to be careful about what they said, but as the session proceeded, they gained confidence in the system's abilities and they stopped consciously restricting their utterances. The remaining eleven subjects felt that they needed to be careful to limit the vocabulary of their questions to use keywords that would be found in an index.

(5&6) The questions asked about the subjects' expectations revealed little useful information. Nearly half of the subjects had no prior expectations and therefore, could not answer these questions. The others stated that the system was about what they expected or better, but their comments were mostly uninformative. A few subjects expressed surprise at how well the system had answered their questions, but the others who responded to this question expected a verbal menu selection task.

(7) Six subjects did not express any complaints about the system or interaction. Six identified slow system responses as the most annoying aspect of the system. Three subjects had unique complaints. They expressed frustration over the use of unfamiliar terms, dissimilarities between FrameMaker and WordPerfect, and the difficulty of orienting while tracking between the help text and the task.

The final two subjects took the opportunity to suggest improvements to the system. One did not care to push a button before speaking and suggested that the system should be voice activated. The final suggestion relates to the slowness of the system. The subject requested a feed-back mechanism for the system to indicate that it had received the question and was searching for an answer. These suggestions have been incorporated into the final version of SAGE (see Chapter 8).

(8) The comments made by the ten subjects who answered question eight were mostly reiterations of previous observations. Two subjects mentioned the ability to refer back to the text as the thing they liked best. Clear help text was identified by others. Four of the subjects identified questioning the system rather than hunting through menus or manuals as their preferred feature. Finally, two mentioned that they liked it because it was fun or new to them.

(9) The final question asked the subjects if they thought the system was useful and if they would use such a system. The responses were unanimous in the affirmative. This result is not surprising since participants in an experiment are more inclined to react positively than negatively to the experience. However, the usage patterns and behaviors exhibited by the subjects that have been described and their responses to the other interview questions lead me to the conclusion that the simulated help system is useful, particularly for novice users of an application. The issue for further research is thus, how close can we come with current technology to reproducing the simulated system or how much less functionality will users accept before the system ceases to be useful?

3.5 Conclusions

This chapter described an experiment to collect a corpus of sample questions from real users performing real tasks in a commercial word processor. The users were given access to a simulated, intelligent help system they could use to find answers to their questions about the application. The system responded to spoken, natural language questions. An analysis of the questions collected is reserved for Chapter 4.

After the subjects completed the tasks assigned to them, they were asked for their impressions and opinions of the help system. Subjects reacted favorably to the system. Most thought it was superior to traditional help systems. They appreciated the presentation of answers as text as opposed to verbal responses. This format permitted them to review the instructions without being required to ask the question again. Subjects were surprised at the amount of English understood by the system, but they still felt that they needed to limit their vocabulary to certain keywords that the system could recognize.

The conclusion from this experiment is that the simulated system was useful for answering task specific questions. Although the system's response time was slow, users adapted to it quickly enough that they could still make effective use of it. Having established the utility of an intelligent help system such as this, the remainder of this dissertation presents an implementation of the system (in which the speech-recognition is still human-assisted). Furthermore, an evaluation of the implemented system shows that it remains useful to users and their reactions to it are not radically different from those described in this chapter.

CHAPTER 4

CORPUS ANALYSIS

The previous chapter described an experiment to collect a corpus of sample questions. This chapter takes a closer look at that corpus. Analysis of the corpus gives insight into the kinds of questions users ask and the interactions they expect to have with the system. The chapter concludes with a discussion of how answer documents were identified and associated with each question.

4.1 Corpus Statistics

The analysis begins with a few basic statistics about the corpus. Subjects submitted a total of 274 questions during their interactions with the help system. Of these, 12 questions were omitted from further analysis. Questions were omitted either because they were a repetition of the previous question or because the question was judged to be too specific to the particular software product's interface. Repeated questions occurred either because the subject was asked to repeat a question when the wizard could not understand it or because the subject dismissed the help text and then forgot how to complete the task. Sometimes, the subjects asked an identical question to retrieve the dismissed text and these duplicates were eliminated. The corpus of questions is given in Appendix B.

Since one goal of this research is to try to avoid application-specific issues, some questions judged to be too specific to FrameMaker were discarded. Examples of this type of question are

- "What is a FrameMaker Alert?",
- "What is the page menu?", and
- "Centering page numbers on master pages."

The first two questions are examples of questions that deal with objects (dialog boxes and menus) unique to the application. The third question asks about a specific concept, master pages, used in FrameMaker. Furthermore, the later two questions were influenced by the help text given in response to the previous question. They were asked after the subjects were given a piece of help text that specifically referred to those interface objects (i.e., the **page menu** and the **master page**).

Before dismissing them, however, these questions provide evidence for an important point to be made to help text authors. The information contained in the help text can lead users to ask clarifying questions about the interface objects being described. Consequently, care should be taken to provide additional information about all of the objects mentioned in the help system, where they may be found, how they can be identified, and what their purpose is.

Descriptive statistics for the corpus are summarized in Table 4.1. Each user asked between 11 and 25 questions, with a mean and median near 15. Each assigned task had one or more subtasks (refer to Table 3.1). The mean and median number

Table 4.1: Descriptive statistics for the corpus of users' questions.

Number of Questions	262
Range (per user)	11–25
Mean	15.4
Median	15
Standard Deviation	4.4

of questions per subtask are both 1.00 with a standard deviation of 0.25.

On average, the questions submitted were not very long. However, the range of the number of words per question is quite large. Five questions consisted of a single word and one contained 22 words. The mean number of words per question is 6.7 and the standard deviation is 3.0. Table 4.2 summarizes the word count statistics for the corpus and Figure 4.1 plots the word count distribution.

Table 4.2: Corpus word count statistics.

Words in Corpus (tokens)	1764
Unique Words (types)	256
Unique Stems	221
Range (words/question)	1–22
Mean	6.7
Median	6
Standard Deviation	3.0

The corpus contains a total of 1764 words or tokens in all the questions, but only 256 unique words or types (or 221 if they are reduced to their root form). Of these, 84 are function or noncontent words. Function words are those typically found in the syntactic classes of pronoun, article, conjunction, preposition, etc. These words are considered noncontent words because they add little to the meaning of a sentence. On average, questions contain 2.6 content words. The distribution has a median of 2.0 content words per question and standard deviation of 1.2. The distribution of content words per question is shown in Figure 4.2.

While the number of unique words is small, the rate at which they are introduced is significant. On average, each question has a 50% chance of introducing a new word to the corpus. This rate includes the noncontent words. If these words are ignored, the chance a new word is used only drops to 40% per question asked (see Figure 4.3). This curve presents both good and bad news for the system developer. The positive point is that the rate drops quickly before it levels off after data has been collected from six to eight users. This implies that most of the vocabulary used in a domain can be collected from a relatively small set of users. However, the curve also suggests that it is probably impossible to collect 100% of the domain's vocabulary. This phenomenon is similar to the vocabulary problem described in (Furnas et al., 1987). New users think up new ways to express their information

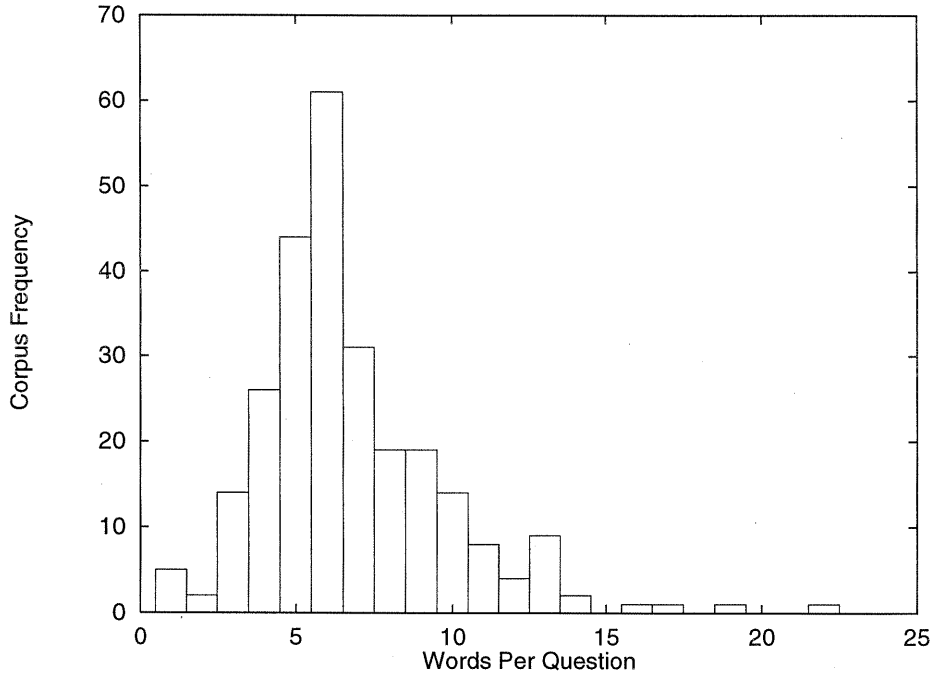


Figure 4.1: Distribution of word counts per question.

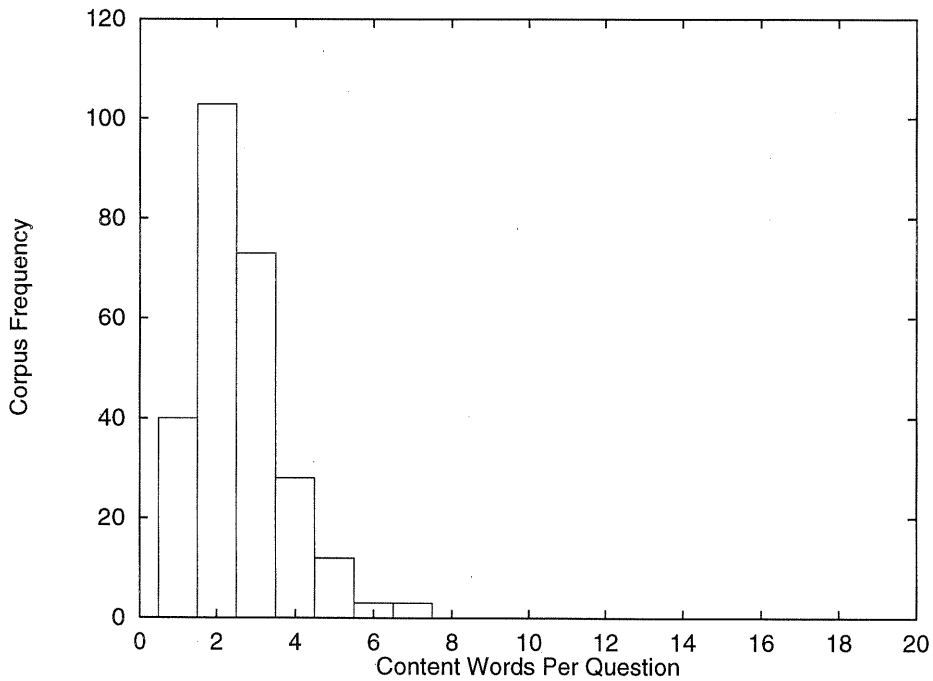


Figure 4.2: Distribution of content word counts per question.

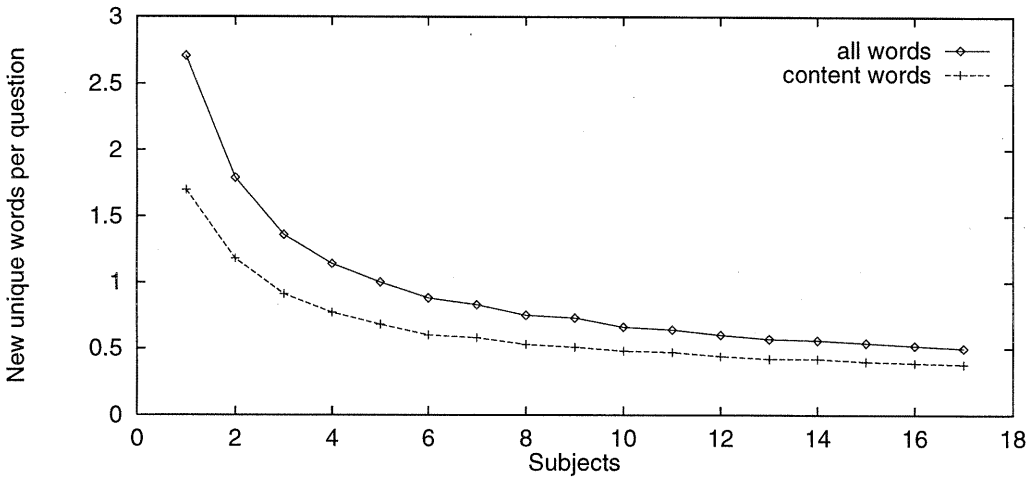


Figure 4.3. Rate at which unique words are added by users for each question asked.

needs and the number of unique words continues to grow. Intuition says that the rate observed in the figure must begin to drop at some point, but data from a larger sample of users would need to be collected to learn how quickly this happens. The rate at which new words are added presents a potential barrier to successful use of state-of-the-art, continuous speech, speaker independent speech recognizers.

4.2 Human-Computer Communication

In a normal day's worth of communication, people use a variety of linguistic devices to express themselves. The corpus contains indicators of the types of linguistic phenomena users of an intelligent help system employ. The question-answering methodology proposed in this research is based on weak natural language processing techniques. Consequently, wide variability in the linguistic attributes of the users' speech would hinder the system's ability to answer questions accurately. Furthermore, if users attempted to engage the system in dialogue, the system would need to track the discourse to respond intelligently. The corpus indicates that there is little reason to be concerned with either of these potential problems.

4.2.1 Linguistic Phenomena Most (85%) of the questions submitted were complete sentences. Of the 37 questions that were not complete sentences, 29 were submitted by two subjects and the remaining eight came from four other subjects. These questions were in one of two general formats. The first kind resulted from the subjects treating the system as a keyword system. Examples of these include the requests **"Footnotes."** and **"Right justify margins."** The other group contained phrases that were missing the subject and an information request verb. The sample phrases

- "Information on bullets please.",
- "Help with table row size.", and
- "Assistance on centering please."

could all begin with the words **"I need"** or **"I want"** as subject and verb to make a complete sentence.

One feature of the Wizard of Oz software permitted the wizards to see the subjects' display. This was included in case subjects referred to screen objects that the wizards needed to know about to answer the questions properly. Two subjects generated the following seven referring expressions.

- "I'd like to change this to italic. How do I do that?"
- "I'd like to change this font to italics."
- "How do I make this bold?"
- "I'd like to put a bullet in front of these."
- "I need to know how to change a cell in this table."
- "How do I center this table?"
- "How do I add a title to this table?"

Four of the seven resulted from a system message reporting that the preceding question was too vague. The subjects tried to be more clear by specifying the particular object of interest.

As it turned out, the ability to see the subjects' screen was not necessary. In each case, the wizard did not need to know the object of the expression to answer the

question. In the first example, knowing which table is to be centered does not change the answer because the process is the same for all tables. In the second two sentences, the item referred to is some piece of text, but again, it does not matter which piece of text the user wants to modify. Changing text to bold and inserting bullets are tasks performed identically regardless of the application state. This observation benefits the construction of an automated question-answering system. Because the use of referring expressions did not affect the correct answer, the system does not need to include special processing techniques for them.

4.2.2 Discourse Management An important consideration in the construction of an intelligent help system is whether the system should include discourse management capabilities. The corpus was analyzed to see if users engaged the system in an extended conversation or whether they expected it to remember portions of the prior interaction. Brunner et al. (1992) make a distinction between moded and unmoded user-system exchanges. Moded interactions are characterized by subdialogues or user-system exchanges that depend on prior exchanges. In contrast, each exchange in an unmoded interaction stands on its own. The distinction is important to the success of an interface because moded interactions require some type of discourse manager if they are to respond appropriately to the user's information request. Currently, effective discourse management is a difficult problem at best.

Previous work studying natural language requests made to a help or advisory system focused on typed input (Guindon, 1988; Guindon, 1991). These studies found that users' interactions with the system were typically unmoded. However, research studying both written and spoken (human-human and human-computer) interactions for other types of information-seeking tasks (mostly travel planning) have found a moded style of interaction (Brunner et al., 1992; Chapanis, 1981; Kowtko and Price, 1989; Bates et al., 1993) between the users and system. The question of interest in this research was whether the interactions would be unmoded like the written help requests or moded as the interactions were in the speech-based travel planning sessions.

Examination of the test corpus revealed that the user-system interactions were distinctly unmoded. This was true even when opportunities for moded interactions presented themselves. A few users dismissed the answer display window before completing a particular task. Then they realized that they still needed the information the system had just given them. Their follow-up question did not use the fact that the system had just provided the desired information. Instead, a self-contained question was asked. For example, one subject said, **"I'd like help on italics again."** While the use of **again** does refer to the preceding interaction, the question can be answered without remembering what had already taken place. If the user had expected the system to remember the previous interactions, the more simple statement, **"Show that again."**, might have been said.

The system was designed to initiate very little of the interaction. However, when users asked vague questions (six instances) the system asked them to be more specific. Each time this happened, they chose to ask a complete, new question rather than restricting their utterance to a clarification of the previous question.

For example, one user said, **"I'd like to find out about editing a cell in a table."** and the system responded that the question was too vague. The user's next submission to the system was, **"I need to know how to change a cell in this table."** The user understood the system's response as a request to identify a specific table. This request could more easily have been satisfied with a statement like, **"I meant this table."**. Rather than make a short clarifying statement, however, the user chose to ask a new question that contained all the information the user thought the system would need to answer it. None of the subjects' utterances required any discourse management on the system's part. Consequently, these capabilities are not necessary (nor do they seem to be expected) in an intelligent help system.

4.3 Relevance Judgments

Before the corpus of questions could be used for testing and guiding the development of SAGE, answer documents needed to be identified. Rather than finding answers for FrameMaker, SAGE was developed with the help texts from WordPerfect. This choice was made to keep the development from being linked too closely to a particular application. Answering the questions with a new help text collection provides evidence that the questions are general in nature and that the techniques used to build SAGE can be applied to other applications and domains as well.

The help texts for WordPerfect are organized as a hypertext system. Each hypertext page was treated as a document throughout this research. The author examined each document and determined whether it was relevant to each question. If a document answered a question, it was considered relevant to that question. There may be some concern that only one judge made all the relevance decisions. However, two things may be said in defense of this situation. First, empirical evidence suggests that variations in relevance judgments among judges do not cause a noticeable difference in retrieval performance (Burgin, 1994). Second, and more important, all relevance judgments were impartial because they were made before a single test of the system was made.

All of the questions collected were assigned at least one relevant document. However, not all of them directly answered the question asked. In a few cases, the user would need to infer an answer from the text. The worst of these involves questions about inserting text, because no document specifically describes how this is done. The document assigned as relevant is an overview document that states "When WordPerfect starts, you can begin to type ...." With this document, the user would be expected to deduce that nothing special needed to be done to insert text. However, the poor match between question and answer hinders SAGE's ability to retrieve this document.

Three different classes of questions were identified during the relevance judgment process. These classes are

(1) simple, unambiguous questions,
(2) simple, ambiguous questions, and
(3) compound questions.

The first type accounts for 90% of the questions in the corpus. Although more than

one document may contain an answer to these simple questions, they are specific enough that a single document contains all the information necessary to answer them.

The simple, ambiguous questions are not as focused as the first group. While the first group might contain a question about page numbering, the second would ask only about numbering. Not only can pages be numbered, but so can lines or list items. The context of the experiment provides enough information to disambiguate which type of number was requested; however, SAGE is not privy to that information and so the question is treated as ambiguous. The corpus contains only 18 (just less than 7%) ambiguous questions. For SAGE to answer an ambiguous question correctly, it must return an answer for each possible interpretation of the question. This approach requires the user who asked the question to recognize which interpretation was intended. During system testing, partial answers were counted for each of the possible interpretations addressed in the list of returned documents.

The questions in the third group are compound because they are requests for two distinct answers. An example from this question class is, **"How do I indent and make bullets?"** The question easily could be two separate questions, one asking about indenting and the other about creating bullets. Eight questions in the corpus (3%) fall into this category. As with the second class, SAGE is required to provide answers to both parts of the question, but answering half the question scores half credit in counting the number of questions answered. Two questions in this class contain an ambiguous request as one part of the compound. Therefore, three answer documents are required to answer these questions correctly. However, two documents account for less than 1% of the corpus.

4.4 Summary

The corpus analysis showed that the questions asked were fairly short and manageable for an automated system. Novel words were added to the corpus at a small, constant rate once data from six to eight users was collected. While this result implies that most of the vocabulary can be collected with a few users, it also suggests that creating a full coverage grammar would be impractical if not impossible. Furthermore, the analysis showed an absence of discourse indicators in the user-computer interaction, thus freeing speech-based help systems from the need to include discourse management capabilities.

For each question in the corpus, a set of documents from the WordPerfect help collection that answer the question was identified and recorded. Most of the questions could be answered with a single document. These relevance judgments became the basis for developing and testing the performance of SAGE.

CHAPTER 5

SAGE DEVELOPMENT

One purpose of collecting, analyzing, and making relevance judgments for the questions was to aid the development of the SAGE (**S**earch for **A**nswers; **G**rant **E**nlightenment) help system. This chapter describes the core retrieval engine used in SAGE and the measures used to evaluate SAGE's performance. The next two chapters further develop SAGE's abilities. Chapter 6 explores the effects of advanced information retrieval techniques on SAGE's performance and Chapter 7 extends SAGE further with natural language processing and machine learning techniques.

5.1 System Architecture

SAGE is composed of many tools and techniques working together to answer questions. A system architecture diagram is given in Figure 5.1. This chapter's focus is on the information retrieval component depicted in the shaded region of the figure. Development of the preprocessing and postprocessing components (the rectangular objects outside the shaded area) is discussed in subsequent chapters.

The left-hand side of the system diagram shows that an answer corpus is first given to a process that works on it before handing it off to the information retrieval engine. Next, the engine produces an index of the corpus. The right-hand side of the figure shows the flow of a user's question through the system. A question can be submitted as text or by voice, in which case a speech recognizer is required. The question is preprocessed and used in conjunction with the index to look for relevant documents. A list of all documents ranked by relevance to the question is produced. A postprocessing component takes the ranked document list and the user's question as input and (potentially) reranks the documents. Finally, the top ranked documents are presented to the user as an answer to the question.

5.2 Latent Semantic Indexing

The information retrieval engine of SAGE is based on Latent Semantic Indexing (LSI) (Dumais et al., 1988; Deerwester et al., 1990), an extension of the vector space retrieval model discussed in Section 2.1.1. The basic idea behind LSI is that lexical synonymy and polysemy tend to introduce "noise" into the standard vector space model. In other words, the occurrence frequency of any particular word in a document may not accurately reflect the true semantics or meaning of the document. Additionally, words consistent with the document's semantics that could have been used may not appear at all. Consequently, the LSI model treats the word usage data as an unreliable estimate of the concepts expressed in the document.

Using statistical techniques to estimate the underlying document concepts, LSI attempts to generate an index space in which semantically similar documents are located "close" to each other even though they may exhibit non-overlapping

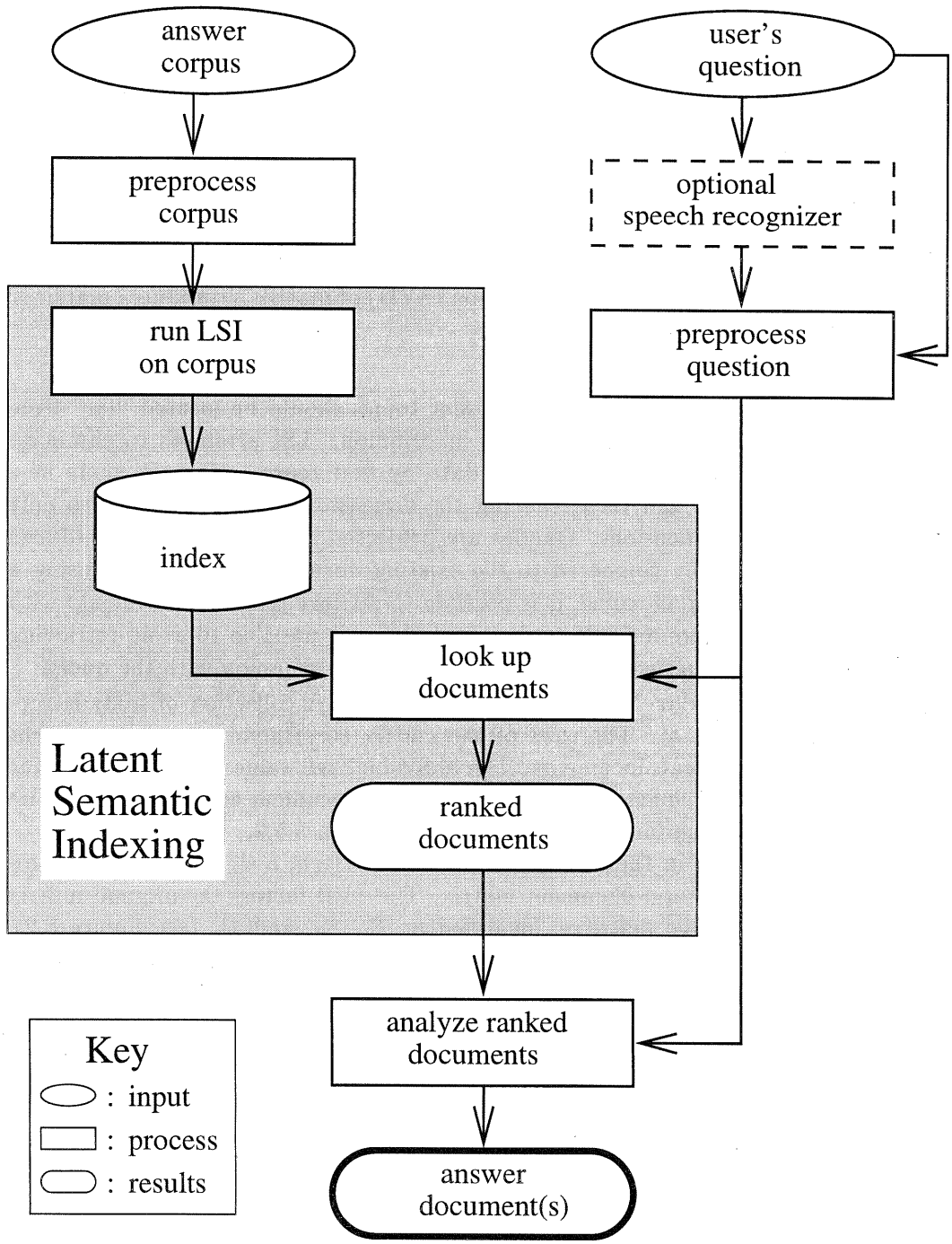


Figure 5.1: SAGE system diagram.

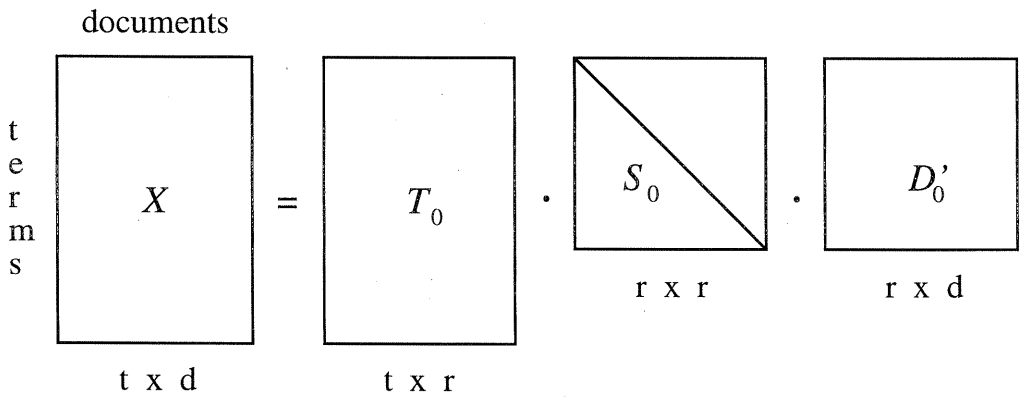


Figure 5.2. The singular value decomposition (SVD) of matrix X produces matrices T_0, S_0 and D_0'.

term usage. Similarly, documents on different topics should be located "far" from each other even if they have some terms in common. LSI attempts to eliminate the noise from the raw word occurrence data by first representing the texts in a high-dimensional space and then reducing the dimensionality of the space to only the most significant dimensions. Queries are projected as vectors into the reduced space where they can be compared to the existing document vectors. By using a proximity measure such as cosine, it is possible to retrieve relevant documents with queries that do not share a single term. Similarly, it is possible to avoid retrieving irrelevant documents that contain one or more terms in common with the query.

5.2.1 Singular Value Decomposition A collection of texts is represented in matrix format. The rows of the matrix correspond to words or terms and the columns represent documents. The individual cell values are based on some function of the term's frequency in the corresponding document and within the entire collection. The cell value function is discussed in section 5.5.2.

The next step in the LSI algorithm is to perform a singular value decomposition (SVD) on the term-document matrix. The SVD factors the original matrix into the product of three matrices, identified as T_0, S_0, and D_0' (see Figure 5.2). The T_0 matrix is a representation of the original term vectors as vectors of derived orthogonal factor values. The matrix D_0' (the transpose of D_0) is a similar representation for the original document vectors. The S_0 matrix is usually referred to as the singular value matrix. It is a diagonal matrix[1] of rank **r** (the smaller of the number of terms and documents). The SVD sorts the singular values in decreasing order along the diagonal. They represent a scaling factor for each dimension in the T_0 and D_0' matrices.

Multiplying T_0, S_0, and D_0' together perfectly reproduces the original representation of the text collection. However, the theory behind LSI is based on the idea that there is noise in the original matrix. Consequently, LSI constructs an approximation of the original space that captures the most important concepts or

[1] A diagonal matrix is a square matrix in which the only non-zero values are located along the the upper left to lower right diagonal.

semantics of the texts by reducing the dimensionality of the T_0, S_0, and D_0' matrices. A chosen number of the least important singular values in S_0 can be eliminated to give a new matrix, S, of rank **k**. Usually, the number of factors **k** to be retained is selected empirically. A similar reduction is made in T_0 and D_0' by retaining the first **k** columns of T_0 and the first **k** rows of D_0' as depicted in Figure 5.3. The product,

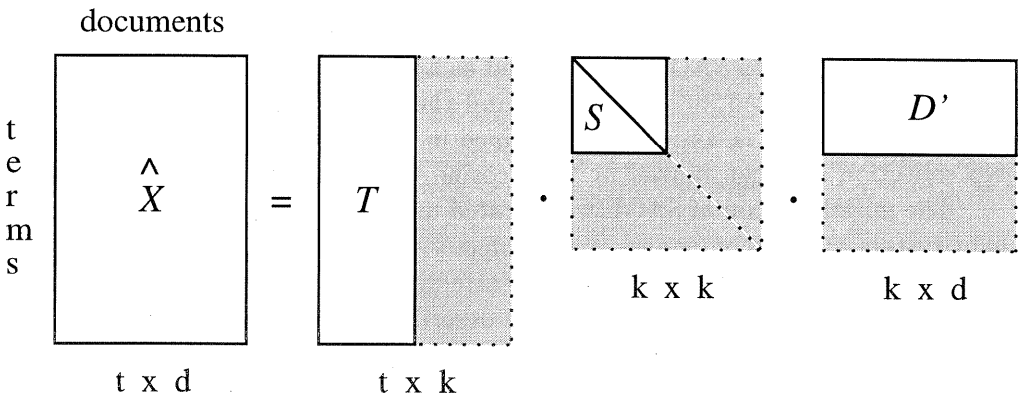


Figure 5.3. The T, S and D' matrices resulting from a reduction to rank **k**. Recombining the reduced matrices gives $\hat{X}$, a least squares best fit reconstruction of the original matrix.

$\hat{X}$, of the resulting T, S, and D' matrices is a least squares best fit reconstruction of the original (Eckart and Young, 1939). The reconstructed matrix represents or predicts the frequency each term might appear in a given document (Landauer and Dumais, 1997). Thus, conceptually similar documents ought to be located near one another in the LSI space since the predicted frequencies for each term in similar documents should also be similar.

5.2.2 Query Processing Once an LSI space has been created, representations for additional documents may be projected into the space. These projections are known as pseudo-document vectors. They are created by taking all the term vectors that correspond to the words in the document, scaling them according to the terms' weights (see Section 5.5.2), and adding them together (Berry and Dumais, 1994). This weighted vector sum creates a pseudo-document vector at the centroid of the term vectors used to create it. Formally, the pseudo-document vector q may be represented as a **k**-dimensional vector $\hat{q}$ via

$$\hat{q} = q'TS^{-1}.$$

Queries are treated as pseudo-documents and represented in the LSI space in the manner just described. With a vector for the query projected into the LSI space, similarity judgments can be made between the query and each document in the help collection. There are a number of possible similarity judgments that can be used (see (Jones and Furnas, 1987) for a treatment of these), but the most commonly used measure is the cosine between vectors. The cosines between the query and each document vector are sorted in decreasing order and the documents ranked

accordingly. This procedure produces a complete rank-ordering of the collection and the top **n** are returned to the user. The value of **n** is discussed later in section 5.4.1.

5.3 The SAGE Kernel

SAGE uses LSI as the basis for its information retrieval engine (see the system diagram in Figure 5.1). It produces an index for a given help collection and answers questions by computing their similarity to documents in the index. When a question is submitted to SAGE, it is treated as a query into the index. A pseudo-document vector is created for the question and the cosine between that vector and all the document vectors is computed as outlined in Section 5.2.2. The documents in the help collection are then ranked by their cosine values and returned to the user.

The initial version of SAGE was created using the documents in the hypertext help system that accompanies WordPerfect's word processor. Each hypertext page in the help system was treated as a separate document. The collection of documents was given to LSI to create an index as described in Section 5.2.1. Creation of the index requires many parameters to be set which inform LSI how to treat terms and documents. These parameters are described in Section 5.5.

Since selection of these parameters must be done empirically, a method for comparing the performance among various parameter values given a set of queries is needed. The questions collected in the Wizard of Oz experiment are used to evaluate the performance of the various indices. The next section introduces the metric used to compare the retrieval performance for the many combinations of parameter values tried. The subsequent section then discusses the parameter values chosen for the baseline performance index.

5.4 Evaluation Method

Because there are many parameter values and combinations of those values possible, a metric of an index's performance that automates the selection of parameter values would be useful. Ideally, a single value that summarizes the performance of the index over a set of questions should be used. This approach permits a simple performance comparison to be made.

The measures typically used to evaluate an information retrieval system are precision and recall. Recall is the percentage of all relevant documents returned and precision is the percentage of returned documents which are relevant. Recall is generally accepted to be a good measure to use. However, the literature contains a great debate about the usefulness of precision compared with other measures of effectiveness (cf. (Robertson, 1969a; Robertson, 1969b) for a compelling argument on the use of fallout rather than precision.) The primary argument in favor of precision is that it is intuitive for users to interpret what it means (Salton and McGill, 1983a).

Fortunately, the nature of SAGE's goal frees us from this debate. SAGE need not return all the information possible about a given topic; its goal is to answer questions. The users' primary goal in using SAGE is to obtain enough information to complete the task at hand. Users do not need to read all the material available on a topic, just enough to proceed with the task. This situation is contrasted with other (more typical) retrieval tasks such as a literature search in which users want

to examine all (or at least many) of the documents about the topic of interest.

The type of recall users want in a help system is what I'll call information recall (as opposed to the traditional notion of recall which I refer to as document recall). In other words, users do not care about recalling all of the documents on a topic; they only want enough information to complete a specific task. SAGE strives for 100% information recall (questions answered) within the fewest number of returned documents. Measuring lower levels of recall is not of much use to the user. Traditional recall-precision or recall-fallout measures are thus unnecessary for evaluating SAGE. Consequently, a measure based on the percentage of questions answered correctly within a given document return set is used. The details of the measure are provided below.

5.4.1 Returned Document Set Even though the entire document collection is ranked, the user is not expected to look at the entire collection (even in traditional retrieval tasks). As a result, many retrieval systems choose a threshold[2] and return only those documents above the threshold. This threshold is referred to as the document cutoff level.

The choice of a document cutoff level must take into account the nature of the collection and the goals of the user when using the system. For this experiment, the collection is of moderate size (approximately 1000 documents) and there is little redundancy in the content of individual pages. For the queries in the test corpus, 100% information recall can be attained with very few documents. A single document is sufficient to answer 90% of the questions. Another 10% require two documents and less than 1% of the questions need three documents from the collection to provide all the requested information. The small number of documents required for 100% recall[3] indicates that the user should not expect to read very many documents before either finding an answer to the question or else stopping because no documents are relevant. At this point, it is difficult to know how many documents users will examine before abandoning their search, but a reasonable guess is that they probably will not look at many more than five or ten.

Ultimately, SAGE's performance should be maximized within the top five documents based on this expectation of user behavior. However, attaining this level of performance is not the job of the SAGE kernel. Its role is to rank the relevant documents high enough that later postprocessing techniques can identify them without searching too many documents. Consequently, selection of an initial set of LSI parameters was based on the kernel's performance at a higher document cutoff level.

The postprocessing techniques described in Chapter 7 examine the documents within a certain cutoff level and eliminate obviously irrelevant ones while boosting the rank of others that appear to be relevant. Recall should be highest within this range to increase the possibility of examining relevant documents during postprocessing. Response time would be too slow if too many documents are examined and so a reasonable number not much larger than five must be chosen. Based on these constraints, a document cutoff level of 20 was chosen (somewhat

[2]The threshold is usually a fixed number of documents or all documents within an accepted degree of relevance.

[3]From this point forward, **recall** refers to information recall unless otherwise specified.

arbitrarily) for automatically comparing the performance of SAGE based on various LSI parameter choices.

5.4.2 Performance Measure The system's performance is computed as the percentage of questions in the test corpus answered by the documents ranked above the document cutoff level. In other words, each question is submitted to SAGE, the collection is ranked, and a tally is kept of whether an answer is obtained by examining the top 20 documents. The performance score is the percentage of questions in the corpus for which an answer is obtained.

Occasionally, two variations of the system answer the same percentage of questions. In this case, the preferred system is the one with the higher-ranked documents (i.e., those with the smallest ordinal positions in the ranked document set). To disambiguate the otherwise equivalent variations, the inverse of the average rank of relevant documents within the cutoff level is divided by ten[4] and added to the correct percentage. The following example helps clarify this computation.

Suppose two different hypothetical systems are used to answer five questions. Further suppose that the correct document for each question is ranked as shown in Table 5.1 by each system. Questions 1 and 2 are ranked equally well by

Table 5.1. Sample rankings of five questions by two hypothetical system configurations.

Question	System 1 Rank	System 2 Rank
1	3	3
2	1	1
3	4	8
4	33	28
5	12	18
score	80.02	80.013

both systems. System 1 performs better than system 2 for questions 3 and 5 because it ranks the relevant document nearer to the top. Since the performance is being computed at the 20 document cutoff level, the only relevant fact about question 4 is that both systems fail to answer it within the top 20 documents. The fact that system 2 ranked the correct document higher is irrelevant. Both systems answer four of the five questions so the decimal portion of their score is 80. Consequently, the distinguishing part of the score must be the fractional part. The average rank of the questions answered by system 1 is 5 ($\frac{3+1+4+12}{4} = \frac{20}{4} = 5$) and the average for system 2 is 7.5 ($\frac{3+1+8+18}{4} = \frac{30}{4} = 7.5$). The total score for each system can now be computed as follows:

$$\text{system 1 score} = 80 + \frac{1}{5 * 10} = 80.02$$

[4]The inverse of the average document rank within the cutoff level is divided by ten to ensure that it is less than one. If it were permitted to be larger, it may affect the value in the decimal portion of the performance score.

and

$$\text{system 2 score} = 80 + \frac{1}{7.5 * 10} = 80.013$$

The scores reflect that the two configurations answer the same number of questions correctly, but that system 1 ranks the answer documents better than system 2.

5.5 Base LSI System Parameters and Performance

As previously mentioned, there are a number of parameters which can be varied in the creation of an LSI index. These parameters can be loosely grouped into term selection, term weighting, and number of singular values or factors to retain. The best combination of values tends to vary by corpus and the desired performance trade-offs between recall and precision. As mentioned in the previous section, the goal in selecting a good set of LSI parameters is to maximize recall at the cutoff level of 20 documents.

There are approximately 10,000 unique combinations of parameters possible. Due to the high computational cost of creating an LSI index, it was impractical to search this space of parameter value combinations completely. However, it was possible to find a good (if not the best) combination of values using a random-restart, hill-climbing search to explore a portion of the space of all value combinations. First, random values for each of the parameters (described below) were set. Then, one of the parameters was randomly selected and an LSI space was generated for each possible value of the chosen parameter. Each space was generated with 500 factors. Performance with a subset of the available factors could also be tested without reconstructing the space. Consequently, the factor value parameter was never included in the bag of parameters to be set randomly. Instead, its range of values was tested exhaustively for each combination of the other parameters selected. The parameter value that produced the best result was chosen and fixed for that parameter. This procedure was repeated until each parameter had been tested and set. Then the whole process was repeated with new, random initial values. Figure 5.4 gives pseudo-code for the random-restart, hill-climbing search described.

Some parameters always achieved their best performance with a certain value, regardless of the values of the other parameters. For the parameters whose values seemed to depend more on the values of the other parameters, an additional, directed search was performed to test all value combinations. The two searches combined to examine about 15% of the total parameter space. The parameters and their selected values are discussed briefly in the following sections.

5.5.1 Term Selection Term selection is the process of creating word tokens from the document collection and deciding which unique tokens (or types) are used to build the index. All uninterrupted alphabetic sequences in the collection are considered as possible indexing terms. There are four parameters related to term selection in LSI.

- minimum term length,
- minimum term frequency within a document,
- minimum term frequency within the collection, and
- term elimination with a stop list

```
N = 20;                          # number of times to run the loop
LOOP 1 to N
    [initialize {parameters}];   # create a set of parameters to test
    FOREACH p in {parameters}    # iterate through each parameter
        p = RANDOM({p-values}); # select a random value for parameter

    WHILE p = RANDOM({parameters})  # randomly select one parameter
        best_s = 0;              # initialize the best score variable
        FOREACH v in {p-values} # try all possible parameter values
            p = v;
            [build LSI index];
            s = [test index];    # get evaluation metric (score)
            IF s > best_s        # save values if the best (so far)
                best_s = s;
                best_v = v;
            ENDIF
        DONE

        p = best_v;              # fix parameter with its best value
    DONE

    FOREACH p in {parameters}    # iterate through each parameter
        [report p];              # report best value for each parameter
DONE
```

Figure 5.4. Pseudo-code for the random-restart, hill-climbing search algorithm used to explore the LSI parameter space.

The first parameter deals with the minimum length (number of characters) of a term. Values of two, three, and four were tested, with three giving the best results. The two-letter tokens are either function words (e.g., of, by), parts of a contraction (e.g, 've, 're), pieces of example names for files, macros, regular expressions, etc., or command line options. Elimination of the command line options as index terms might produce poor results for a question about the functionality of a specific option by name. Since there are no examples of this in the question corpus, there is not a way to test the results accurately. However, the question may also contain other words (e.g., option, command line, startup) that would produce the correct document. Though there are a lot of nonsense three letter tokens as well (40%), omitting all three letter tokens would eliminate some useful terms. For example, **cut** as used in "cut and paste text" would be eliminated. Being more selective about which three letter tokens are used might produce a better index still.

The next two parameters select terms based on their frequency within an individual document and within the collection as a whole. Within the information retrieval community it is common to use words as indexing terms only if they appear in two or more documents. Also, when the documents are very long, words might be required to appear more than once in the document as well. As was expected with the help text collection, requiring any term to appear more than once in the document or in more than one document reduces the average performance. This result is probably due to the nature of the collection. The documents are relatively short and there is little overlap in topics. Consequently, there are important terms that only occur once in the whole collection.

The final term selection parameter indicates whether to use a stop list to eliminate certain terms from the index. A stop list is a set of words which are common and (probably) contribute little information value. Typically, the list contains the closed class words such as prepositions, pronouns, conjunctions, auxiliaries, and some adverbs. There is some debate about whether a stop list is helpful, particularly with LSI (Landauer, personal communication). However, most systems, including SMART (Salton, 1971; Salton and McGill, 1983a; Salton, 1989; Buckley et al., 1995) and others at TREC-3 (Cooper, Chen, and Gey, 1995; Dumais, 1995; Kwok, Grunfield, and Lewis, 1995), are using a stop list. The search process found that using the 439 word stop list bundled with LSI improves performance by about five percent at the selected cutoff level of 20 documents. Additional investigation reveals that it has little effect for very small cutoff levels. Consequently, use of a stop list is incorporated into SAGE.

5.5.2 Term Weighting One of the most common and effective methods for improving retrieval performance is term weighting. This technique is an attempt to give more weight or importance to those terms that are important in a particular subset of the collection. In other words, a word that appears in many documents is not likely to discriminate relevant documents very well so it is deemed less important than those words which only appear in relatively few documents. Hence, if a non-discriminating term is used in a query, its power or influence in selecting documents is diminished. Typically, term weighting formulas are the product of both a local and a global measure of a term's importance.

The local weight component measures the influence of a term in a particular document. The local weighting is usually the frequency of term **k** in document **i**, tf_{ik}, or the log of this frequency, $log_2(tf_{ik})$. The local weight is applied to individual cells in the term-document matrix.

The global component is a weight of a term's overall importance in the document collection and it is applied to each row of the matrix. The most common global weight is known as the inverse document frequency (idf) weight. The idf of term **k** is computed by:

$$idf_k = log_2(\frac{ndocs}{df_k}) + 1$$

where $ndocs$ is the number of documents in the collection and df_k is the number of documents in which term **k** appears. This formula emphasizes those terms that occur in the fewest number of documents because terms occurring in a large number of documents do not have the power to discriminate among them.

An alternative global weighting function is based on the entropy of a term within the collection. Evidence supporting the use of entropy has been presented in (Dumais, 1992; Harman, 1986; Lochbaum and Streeter, 1989). The entropy of term **k** is computed as follows (Charniak, 1993):

$$entropy_k = -\sum_{i=1}^{ndocs} \frac{tf_{ik}}{gf_k} log_2 \frac{tf_{ik}}{gf_k}$$

where $ndocs$ is the number of documents in the collection, tf_{ik} is the frequency of the $\mathbf{k}^{th}$ term in the $\mathbf{i}^{th}$ document, and gf_k is the global frequency of **k** (i.e., the number of times term **k** appears in the entire collection). Exploration of the various weighting combinations found that using entropy over idf increased performance by 11%! There is little difference between using tf_{ik} and its log_2, but tf_{ik} shows a slight edge.

The final parameter that affects term weighting is whether to normalize the weights based on the length of the document. Experimentation with a normalizing factor on this collection showed that it decreased the average performance by a couple percentage points.

5.5.3 Factors The selection of an appropriate number of factors or singular values to retain must be done empirically to find "what works best" (Dumais, 1992). Landauer and Dumais (1997) show that the performance of an LSI index of an encyclopedia rises steadily until approximately 300 factors are reached and then the performance begins to decline. Their collection is much larger than the number of help texts indexed by SAGE and so we would expect a larger number of factors to be useful in their LSI space. Other researchers (see (Dumais, 1992; Streeter and Lochbaum, 1988; Lochbaum and Streeter, 1989; Foltz, 1995)) attained good performance on smaller collections with 100 factors. Consequently, the best performance was expected to be achieved with some number of factors between 100 and 200 based on a comparison of collection sizes. The factor values tested were 80, 100, 120, 130, 150, 170, 180, 200, 300, and 500 to permit detailed exploration within the expected range and a sanity check using a few values outside the range. Experiments found that retaining 180 factors produced the best overall performance.

5.5.4 Baseline Performance The parameter values selected to establish a performance baseline are summarized in Table 5.2. These parameters pro-

Table 5.2: LSI parameters selected to establish SAGE's baseline performance.

Parameter	Value
minimum term length	3 characters
minimum term frequency within a document	1
minimum term frequency within the collection	1
term elimination with a stop list	yes
document-level term weighting	term freq. (tf)
collection-level term weighting	entropy
factors	180

duce an LSI index of 3742 terms by 1073 documents for the WordPerfect help collection. Figure 5.5 shows the performance of the baseline SAGE information retrieval component plotted at various document cutoff levels from 1 to 20. The vertical axis on the graph indicates the total percentage of questions answered at a particular document cutoff level. The figure shows that the LSI kernel answers 33% of the questions correctly with the first document and 71% of the questions when the top 20 documents are examined.

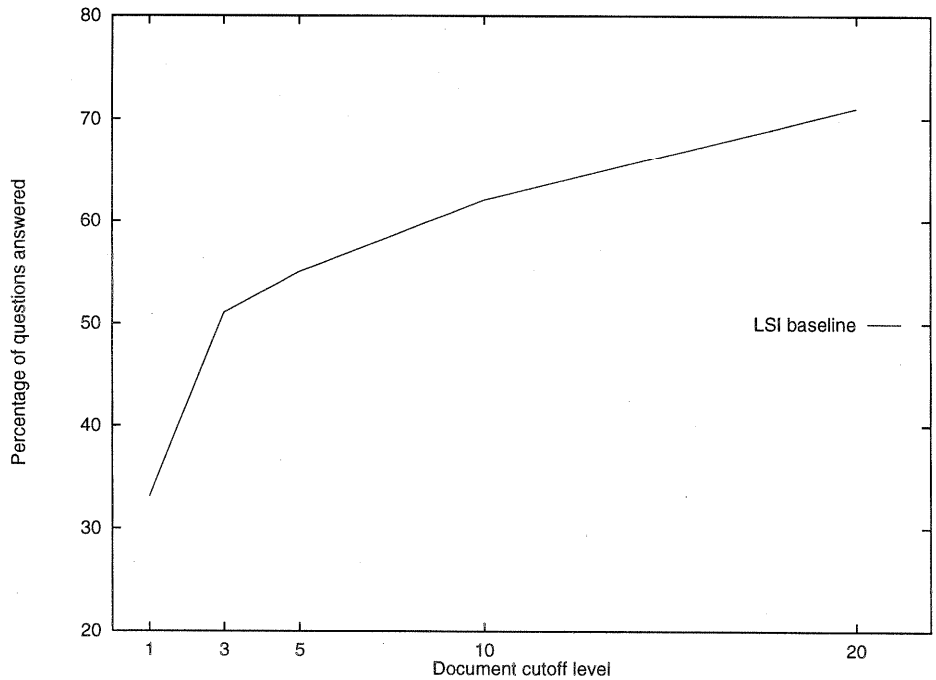


Figure 5.5: Baseline performance of SAGE's LSI index.

5.6 Comparison with the Standard Vector Space Model

The performance of the standard vector space model (described in Section 2.1.1) is equivalent to an LSI model in which none of the dimensions are eliminated. Figure 5.6 shows the performance of the vector space model next to the baseline SAGE performance curve. Error bars for 50% and 95% confidence intervals

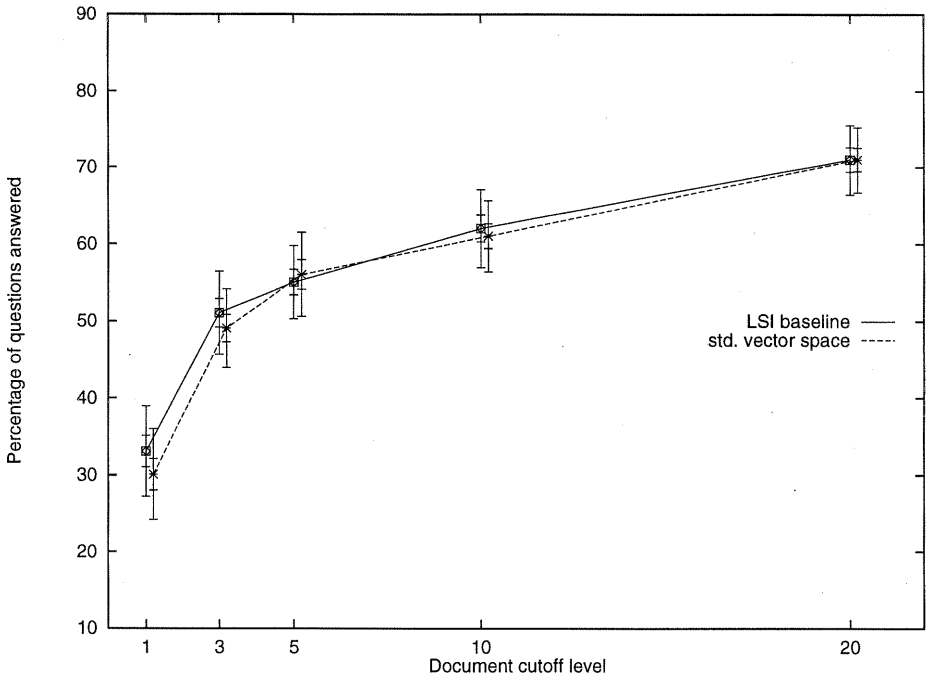


Figure 5.6. Baseline performance of SAGE's LSI index compared with the standard vector space model.

have been added at each document cutoff point sampled. Note that the second curve is offset slightly from the first to keep the error bars from overlapping. Subsequent graphs in this document that contain error bars also offset the plots for the same reason. The figure shows the performance of the two systems to be fairly comparable, but LSI gives some improvement at the first document cutoff level.

5.6.1 Statistical Analysis An interesting question to ask about the differences in the performance curves is how different are they? The error bars in the figure provide information about how likely the true performance means of two different systems are to be near one another. They give a range within which the system's true performance is expected to lie for subjects from the sample population. The two different intervals are used to indicate that we are either 95% or 50% confident that the system's performance lies within the indicated range.

The small sample size (17 users) makes finding statistically significant results difficult. However, when choosing one system configuration over another, the difference may not need to be statistically significant as long as we can be reasonably certain we are choosing the better system (Landauer, 1988). Two statistical measures, effect size and odds ratio, give an intuitive sense of whether one system

truly is better than another or if the observed results are from an anomalous set of data. These two measures are applied in the analysis of these curves and the rest found throughout the dissertation.

Effect size measures how large the impact is that a variable has on an experiment's results. It is computed as the number of standard deviations of change produced by the experimental variable. So, a variable with an effect size of 1.0 is said to have a fairly large effect because it causes a one standard deviation change.

The odds ratio is an interpretation of confidence in the results. Using this measure, we can say that the odds are X to Y that there is some difference in the true performance of two different systems. This analysis gives an intuitive feeling for whether it is worthwhile to choose one system over the other. The odds ratios reported here are computed by finding the probability, P, that there is no difference between two systems. A two-tailed, paired-means t-test on the subjects' mean performance scores for each system tested is used to compute P. The probability that there really is a difference is given by $1 - P$, so the odds are $1 - P$:P that one system is better than the other.

Suppose that a new configuration of SAGE is tested and its performance measured. Assume that the t-test reports a probability of 0.06 that the new configuration is no better than the original. By standard statistical measures this result is not significant and we must conclude that the systems are essentially identical. However, the odds ratio tells us that there are 94:6 or nearly 16:1 odds that choosing the new system would result in improved performance. If we are only interested in choosing the better of two systems, the odds favor choosing the new system. It should be noted that an odds ratio of 95:5 or 19:1 is equivalent to statistical significance at the 0.05 level. Any odds ratio better than this one is an indication that the results are statistically significant.

5.6.2 Performance Summary With the effect size and odds ratio, the differences between the baseline LSI system and standard vector space model can be analyzed. Table 5.3 shows these measures for each of the selected sample points in the data. The effect sizes reported are the amount of impact on SAGE's performance

Table 5.3. Effect sizes and odds ratios comparing LSI with the standard vector space model.

	1	3	5	10	20
Effect Size	0.25	0.11	-0.1	0.19	0.03
Odds Ratio	14:1	6:1	2:1	5:1	0

LSI had over the standard vector space model. The odds ratios indicate how likely it is that LSI would outperform the vector space model at each cutoff point. The negative effect size at document cutoff level five comes from the standard vector space model having a higher average performance at this sample point. None of the effect sizes are very large. As seen in Figure 5.6, the largest effect is at cutoff level one. The odds that there really is an advantage at this point to using LSI are 14:1. Although this value is not statistically significant, it suggests that chances are pretty

good that LSI is the better choice.

5.7 Comparison with WordPerfect's Help System

Figure 5.7 plots the performance of the question corpus when using the keyword-based help system that comes with WordPerfect (WP) 6.0 running under Windows 3.1. The WP help system performance data is provided as a basis for

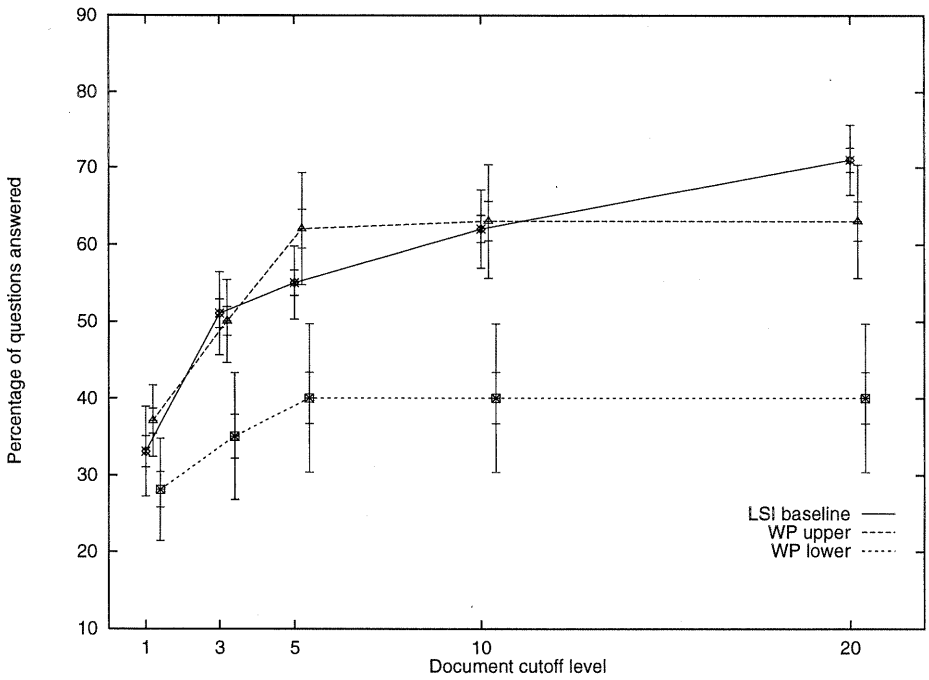


Figure 5.7. Performance of SAGE's LSI index compared with an estimate of the upper and lower performance bounds of WordPerfect's help system.

judging how well LSI's full-text index works compared with this keyword-based index. Construction of the WP help system required someone to assign keywords to each document in the collection. Users search the help collection by typing a word (or two). As the users type, a sorted list of keywords updates itself automatically to show the set of keywords that share the same initial letters as those that have been entered. Users locate and select a keyword from this set and a list of topics or document titles that are indexed by the selected keyword are presented. Selecting one of these items displays the corresponding document.

Evaluation of this keyword system in the same manner as was done for SAGE is difficult because it does not accept full natural language input. The WP help system is intended to be used as an interactive search tool. Furthermore, it is conceivable that a user of this system would chose an entirely different keyword than one of the words present in the spoken utterance. Consequently, the data points for the WP help system should be used only as a rough estimate of the system's true performance. Figure 5.7 shows both an (approximate) upper and lower bound on the performance of the WP help system using the same corpus of questions used to

evaluate the baseline performance of SAGE.

5.7.1 Performance Estimation The two performance curves for WP were obtained by estimating the number of actions required to access the document(s) that answered each question. In traditional information retrieval systems that rank the document collection, examining each document title in rank order can be thought of as an action. In this manner, the rank of a document is exactly equal to the number of actions necessary to find it (loading the document text is not counted as an action in this case). Similarly, counting the number of keywords and titles examined or other actions performed in the WP system between entering the initial keyword and loading a relevant document can be used to estimate the rank of a relevant document. Because the position of a relevant document in the list of suggested titles is closely related to the number of actions needed to access it, the description of each performance bound that follows sometimes uses the position of a relevant document in this list as its rank. The two bounds arise only by the amount of "human effort" present in the search methods. Stop list words are removed from the questions in both cases.

The upper performance bound was obtained by selecting the word (or words) from the question that (hopefully) would be most relevant or helpful in locating a correct answer. This word was used as the keyword submitted to the help system. After the keyword was entered, the WP help system compared it with the sorted list of known keywords. The position of the entered keyword within the list was determined and the six keywords that follow this position were displayed. If the first keyword in the list of candidates did not lead to the right answer, some reasonable variation of the keyword or the next most promising word in the sentence was used as a new keyword and a penalty of one action was incurred. Reasonable variations included adding or removing suffixes to the keyword to reposition it within the sorted list of all keywords. For example, suppose the question **"How do I center a table?"** was asked. The keyword phrase **"center table"** would be submitted to the help system. If the top three phrases returned by the system were

(1) centering lines,
(2) centering pages, and
(3) centering tables,

the upper bound estimate would add the appropriate suffix to the submitted phrase to match the third item and count a penalty of one action. The position of the correct document in the list of document titles (plus any additional actions) was recorded and used to generate the graph. For compound questions only partially answered by a single document (e.g., "How do I make letters bigger and in bold?"), the process was repeated for the second half of the question.

The lower performance bound was obtained much more systematically. A keyword phrase was identified as follows.

(1) Remove any initial part of the question that was used to express an information need (e.g., **"I want information about"** or **"I need help with"**).
(2) Eliminate words that appear in the LSI stop list.
(3) Use the remaining terms as the submission to the keyword system.

The topics associated with the first keyword listed were examined for a correct answer

and the position (if present) of the relevant document was recorded.

An example will help make this more clear. One question submitted was **"How do you change the size of a font?"** The phrase **"change size font"** was submitted to the keyword help system because the first three words were all removed by the stop list. The only document selected by the help system was one that describes how to set up automatic font size changes for different attributes (e.g., subscripts and math symbols). The lower bound estimate did not permit further searching. Since this document did not answer the question, the lower bound did not receive credit for this question. The upper bound estimate was obtained through additional searching. A different set of terms, "font size", was submitted and a penalty of one action was incurred. The new query succeeded in returning the desired document in the second position. The resulting score for the upper bound estimate was a successful answer with a rank of three.

5.7.2 Performance Summary Figure 5.7 shows that the first returned document answers 37% and 28% for the upper and lower performance bounds, respectively. At a document cutoff level of 20, the performance is 63% and 40% respectively. Note that the performance nearly flattens out after a cutoff level of five. This behavior is due to the limited number of suggested documents displayed by the WP system for any given query. Since the system does not rank the entire collection as SAGE does, the relevant document is either present in the given list of up to six documents or it is not. The upper performance bound increases a little in the range of five to ten documents from action penalties incurred searching for answers.

The effect sizes and odds ratios of using LSI compared with the two WP performance bounds are given in Table 5.4. Figure 5.7 shows that the upper bound

Table 5.4. Effect sizes and odds ratios comparing LSI with the WordPerfect help system.

LSI and WordPerfect upper bound					
	1	3	5	10	20
Effect Size	-0.29	0.12	-0.6	-0.04	0.84
Odds Ratio	2:1	1:1	11:1	0	39:1

LSI and WordPerfect lower bound					
	1	3	5	10	20
Effect Size	0.34	1.35	1.55	2.19	3.22
Odds Ratio	7:2	6×10^3:1	253:1	6×10^3:1	249×10^3:1

of WP performs better than LSI at the one and five document cutoff levels. The odds that there is a difference are roughly 2:1 and 11:1, respectively, in favor of the WP system. The effect sizes of WP at these cutoff levels are roughly 0.3 and 0.6. In other words, WP has a small to moderate effect size on performance at these cutoff levels. At the 20 document cutoff level where LSI's performance has surpassed WP's, the

odds are 39:1 that LSI truly is performing better. The effect of LSI here, 0.84, is a little larger than seen for WP. The odds that LSI is better than the lower WP bound at answering questions with the first returned document are only 7:2. However, the odds that LSI gives superior performance by the third returned document explode to 6×10^3:1! The effect size jumps to 1.35, placing LSI's performance above the 80^{th} percentile[5] of the expected lower bound performance.

Although the baseline LSI index performance falls somewhere within the range of WordPerfect's keyword help system for small document cutoff levels, there are two advantages to using LSI. First, the WP help system requires someone to examine each document and assign appropriate keywords. This laborious process is not necessary using LSI. Second, LSI performs better at larger document cutoff levels. This fact provides for the possibility of increasing the performance at smaller levels by including some additional query and document processing facilities as described in next two chapters.

5.8 Discussion

This chapter outlined the LSI kernel used in developing SAGE. Since there are many LSI parameters that must be set empirically, a random-restart, hill-climbing search was used to establish suitable values for these parameters. From these values, an LSI index was produced to establish a baseline of performance against which other configurations can be tested. The performance of SAGE is based on evaluating how many questions it answers correctly after examining a specified number of the top-ranked documents. Furthermore, the statistical measures, effect size and odds ratio, were set forth as measures on which to judge the impact on performance that a particular technique or system configuration has over another.

The results of comparing SAGE's kernel with the standard vector space model and WordPerfect's help system suggest that there is reasonable evidence to believe that using LSI is advantageous. Although, the differences are not statistically significant, the odds are sufficiently high to warrant choosing LSI as the basis for SAGE's information retrieval engine. The next two chapters build on this kernel to improve performance further. However, since there is not a large difference in the performance of the two models, the techniques tested and the results obtained in the subsequent chapters are probably applicable to the standard vector space model too.

[5] Over 80% of the area under the normal distribution curve is contained within 1.35 standard deviations of the mean. Thus, the performance of LSI is above 80% of the norm.

CHAPTER 6

EXPERIMENTS WITH SAGE-IR

Ultimately, the goal of a help system is to assist its users in accomplishing their tasks efficiently. Therefore, the real test of a help system should be done in an interactive setting. Such an experiment is discussed in chapter 8. However, before subjecting the system to user testing, the system should be equipped with the best question-answering facilities possible. Initial evaluation of the retrieval mechanism is best done in a more controlled environment. This chapter describes the results obtained by extending SAGE's query and document preprocessing abilities.

There are many techniques described in the literature used to improve the performance of information retrieval systems. Six of these were chosen for testing. The effect each technique is designed to have on the construction of the index or handling of the question is described. The techniques were applied independently to the LSI kernel and the resulting performance was measured[1] using the test corpus. The following techniques were tested:

- Phrase Creation
- Stemming
- Query Expansion
- Relevance Feedback
- Passage Retrieval
- Supplementing the Collection

After the discussion of these experiments, the techniques that improve SAGE's performance are collected and a new baseline performance standard is established. Because each technique is tested alone and compared with the current baseline LSI system, any interactions among techniques that might exist are not explored.

6.1 Phrase Creation

Phrase creation is the technique of using more than one word as a single indexing "term." Grouping multiple words into a single term is an effort to overcome the effects of polysemy (words with more than one meaning) in the index terms by providing more information about the specific context in which a polysemous word occurs. Packaging additional words with a polysemous word to create a single term is more likely to help disambiguate the meaning of the polysemous word. For example, the word **bank** has a different meaning when used with the word **piggy** than it does when used with the word **river**. Including the phrases **piggy+bank** and **river+bank** as index terms could help narrow the set of returned documents for a query about banks provided the query also contains the other word in the pair (or another means is available to select one of the phrases).

[1] Chapter 5 outlines the evaluation measures used in this chapter.

Phrases may be created in several ways. One approach is based on using terms that have a low global weighting to construct phrases. They appear frequently in the document collection and are thus more likely to be polysemous. The "phrase cohesion" approach makes phrases from words that co-occur more frequently than chance. See (Salton and McGill, 1983a) or (Smadja, 1993) for the details of these approaches. A simpler approach is to use bigrams or adjacent pairs of words as phrases. Jones and Martin (1997) showed that bigram phrases can have a positive effect on an LSI index used for disambiguating selected words.

The bigram approach to phrase construction was tested with SAGE. The addition of bigram phrasal index terms did not improve the performance over the baseline system as seen in Figure 6.1. The failure of bigrams to improve performance

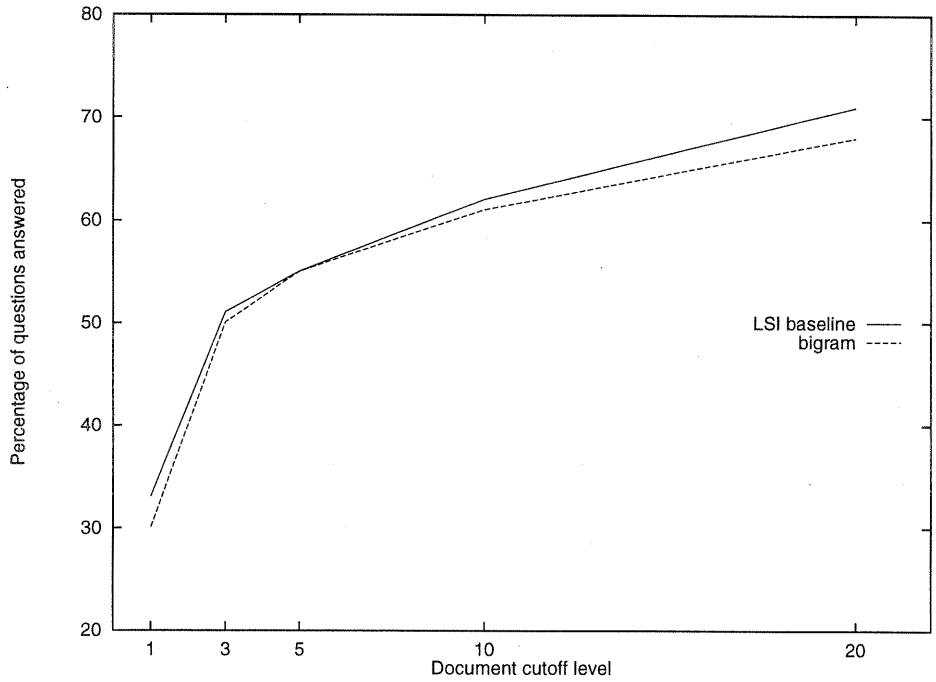


Figure 6.1. Performance of SAGE's LSI index augmented with index terms created from bigram phrases.

is probably due to the content of both the documents and the queries used in this experiment.

Bigrams help improve performance when important word order distinctions should be made (e.g., Venetian blinds vs. blind Venetians). They can also be helpful in disambiguating between different word senses (e.g., piggy banks vs. river banks). The results suggest that neither of these cases is prevalent enough in the collection nor queries to make a significant improvement in performance. Only a few of the questions in the test corpus contain words such that creating bigrams helps disambiguate one of the terms. From the test corpus, the phrases **accent+mark** and **spell+check** help disambiguate **mark** and **check** respectively. However, the single words **accent** and **spell** are sufficient by themselves as query terms to retrieve the

correct documents.

A notable exception to this rule occurs when both words are ambiguous, as in the example **page+number**. Producing bigrams with this pair of words does improve performance. The test corpus contains 27 questions that ask about various operations on page numbers. Isolating the performance to only these questions gives the performance curves shown in Figure 6.2 (note that the y-axis range has increased). Bigrams improve the performance significantly within the range of cutoff

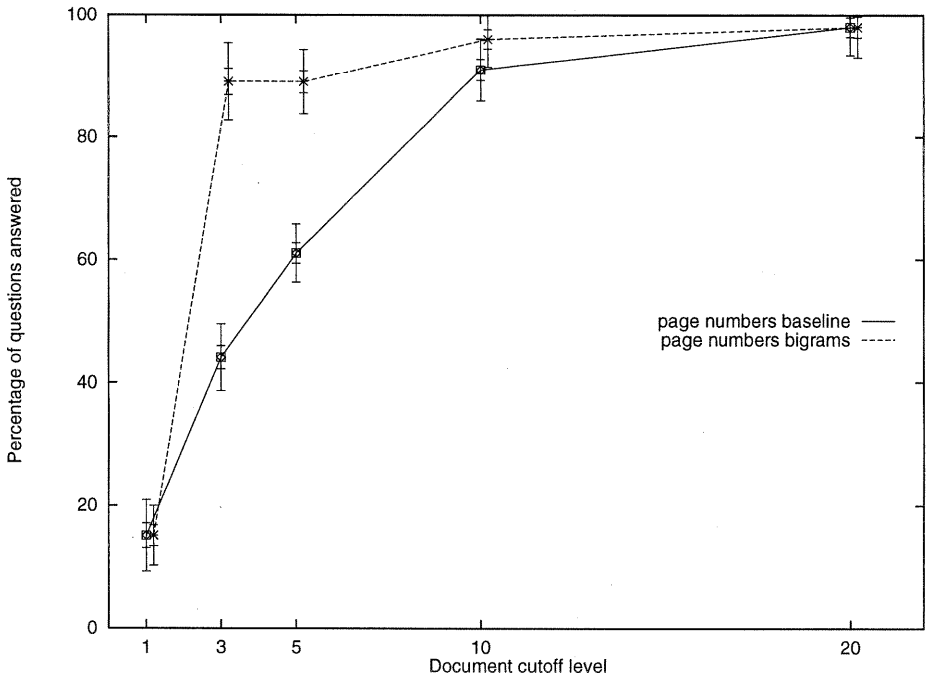


Figure 6.2. Performance of SAGE's LSI index augmented with index terms created from bigram phrases for questions about page numbers.

points examined, though they show no difference at the end points. Table 6.1 shows the effect at the three and five document cutoff levels to be particularly strong. The odds that there is a significant effect are also very high, 5×10^3:1 and 60:1, for the sample points at cutoff levels of three and five documents respectively.

Table 6.1. Effect sizes and odds ratios of bigrams on questions about page numbers.

	1	3	5	10	20
Effect Size	0	1.4	0.8	0.2	0
Odds Ratio	0	5×10^3:1	60:1	5:1	0

This increased performance must be offset by some deteriorated performance on other queries since the total performance using bigrams changes very little compared with the baseline system, An effort was made to identify general characteristics of the poorly performing queries. However, a general explanation for the

cause of the deteriorated performance has not been identified yet.

6.2 Stemming

Word stemming is the process of converting a word to its morphological root form by removing any suffixes it may have. Prefixes of a word are usually not removed since they often change the meaning of the word to its opposite meaning (e.g., dependent vs. independent, common vs. uncommon). The theory behind stemming is that reducing words with slightly different morphological endings, but similar meanings (e.g., **retrieve**, **retrieving**, and **retrieval**), to the same base form (e.g., **retriev**) helps make documents appear more similar to one another even if they use different forms of the same root word.

Stemming may also introduce errors (unless the algorithm is particularly sophisticated) because some words with different meanings may reduce to the same root word form. For example, the words **head** and **heading** are likely to be reduced to the same root, as are **numerical** and **numerous**[2]. Conflating two semantically distinct words to the same term often degrades the precision of the retrieval system.

In spite of the potential problems with stemming, most information retrieval system designers seem to think it is a good idea (cf. (Buckley, 1985; Cooper, Chen, and Gey, 1995; Salton, 1971; Strzalkowski, Carballo, and Marinescu, 1995)). Harman (Harman, 1991) tested the effects of stemming using a simple S-stemmer (i.e., stem plural words only), Lovins' strict algorithm (Lovins, 1968), and the weaker Porter algorithm (Porter, 1980). None of the methods showed any significant improvement in retrieval results; however, Harman still recommends a weak stemming algorithm for two reasons. The first reason is purely practical. Since the performance did not change significantly in her experiments, stemming can save space in the index by reducing the total number of index terms used. The second reason is a matter of users' expectations. Harman believes that users expect information retrieval systems to stem words and thus they expect documents containing words with different suffixes to still be retrieved.

One note about these results is that the standard information retrieval collections used in Harman's experiment contain mostly document titles and some abstracts. The text in these portions of the document may not benefit from stemming as much as the full text of the documents can. Document titles do not vary morphologically as much as the contents of the full text. In contrast to Harman's results, some experiments with LSI (Lochbaum and Streeter, 1989) have shown modest improvements in retrieval performance using a weak stemming algorithm on the abstracts of technical documents.

SAGE used Porter's stemming algorithm (Porter, 1980) in the tests reported here. Like the Lochbaum and Streeter results, the improvement in SAGE's performance was sufficient to warrant further use. Figure 6.3 shows that stemming produces a raw 5% to 12% performance improvement at all document cutoff levels. The effect sizes and odds ratios when using a stemmer are also significant. Table 6.2 shows these values for each document cutoff level. The stemmer produced a moder-

[2]Stemming problem words taken from (Riloff, 1996).

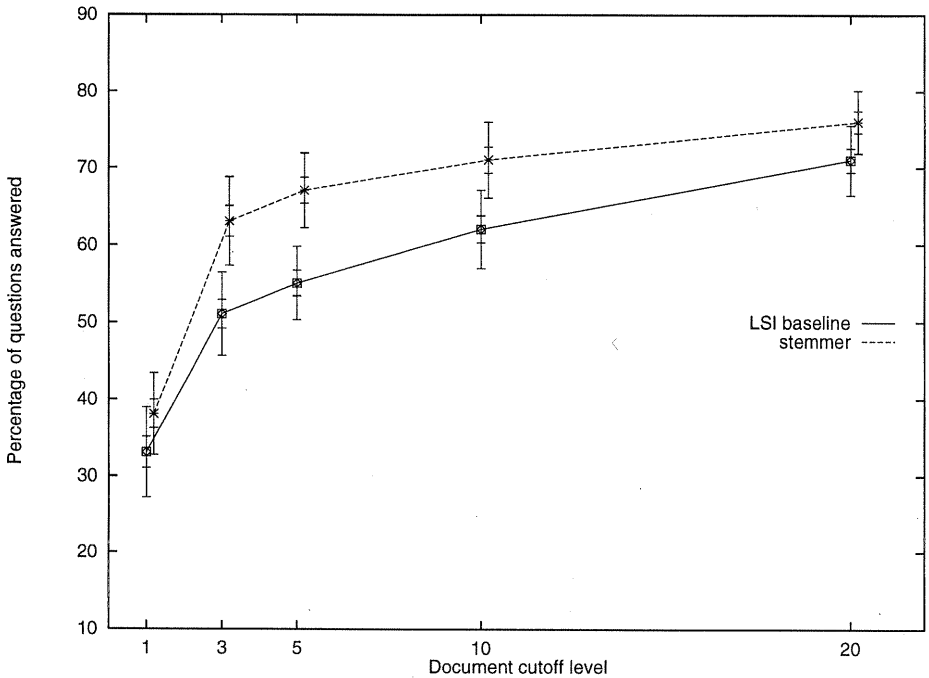


Figure 6.3: Performance of SAGE's LSI index augmented with a word stemmer.

Table 6.2: Effect sizes and odds ratios of using a stemmer.

	1	3	5	10	20
Effect Size	0.49	1.08	1.14	0.72	0.5
Odds Ratio	14:1	6×10^3:1	14×10^3:1	7×10^3:1	1×10^3:1

ate to large effect size on performance. More striking, however, are the odds that it improves the system's performance. At a document cutoff level of five, the odds are almost 14,000:1 that using the stemmer is better!

6.3 Query Expansion

Query expansion is a technique commonly used to increase the recall of a system. Query expansion may be done before or after an initial retrieval. In the later case, the expansion is typically called relevance feedback. The role of relevance feedback in query expansion is reviewed in the next section.

Salton is a proponent of expanding the query before doing an initial lookup (Salton and McGill, 1983a; Salton, 1989). Short queries lack many semantically similar words and so they may fail to retrieve many relevant documents that do not contain any of the query words. LSI was designed to overcome this word synonymy problem and it does to some degree. However, it may still benefit from "wordier" queries.

The biggest problem with this technique is finding an appropriate means of doing the expansion. A common approach is to extract text from a thesaurus and add it to the query. The thesaurus may be preconstructed or automatically generated. In either case, multiple senses of a word cause problems for selecting the correct text. Constructing a thesaurus for a particular domain by hand is time-consuming and tedious. Electronic versions of general, preconstructed thesauri are available, but the computer application sense of many words is not present. This is primarily due to the age of the available thesauri (e.g., the 1911 version of Roget's thesaurus).

Recent work has been done in automatically constructing thesauri from large text collections (Grefenstette, 1994). This approach has the advantage of being able to generate domain-specific thesauri efficiently. However, building one of these applications is beyond the scope of this work. Consequently, SAGE must use text that is already available.

The best text available is the glossary contained within the help text itself. Each term or short phrase used in a question is looked up in the glossary. If present, the definition text is added to the question text and the expanded query is then used for ranking documents. The results of this method are presented in Figure 6.4.

The difference in performance between the query expansion and baseline methods is minor. The probable reason is that only 15% of the questions contain terms that appeared in the glossary. Thus, few questions are actually expanded. A second possibility is that the text used for augmenting the queries is already captured in the index. In doing the dimension reduction of the space, LSI should

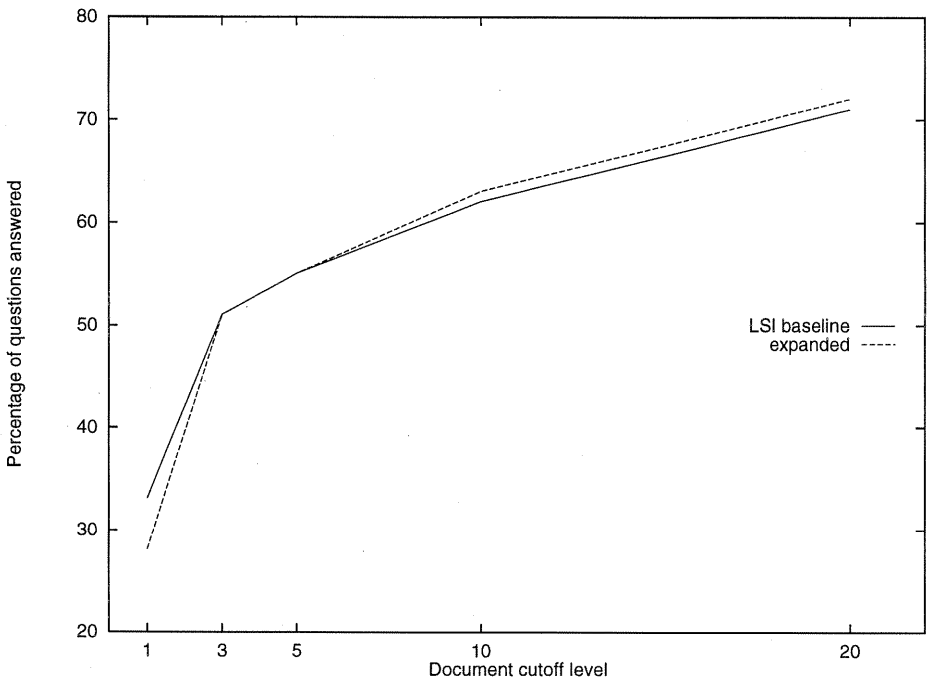


Figure 6.4. Performance of SAGE's LSI index using queries expanded with glossary definitions.

have captured some synonymy provided in the glossary text by itself. For query expansion to be effective in this problem domain, it is probably necessary to use an external thesaurus that can map unknown query terms to synonymous words found in the index.

6.4 Relevance Feedback

Like query expansion based on a thesaurus, relevance feedback is a query expansion technique designed to increase the recall of a system. In a typical interactive environment, a set of documents is returned to the user who makes a judgment about the relevance of each. Those that are relevant are then used to modify the original query. Several techniques may be employed to update the query. One is to adjust the weights of the query terms, increasing the weights of the terms found in the relevant documents and decreasing the others' weights (Robertson and Sparck Jones, 1976). A common method used with the vector space model is to combine relevant documents into one large query (Deerwester et al., 1990). By using whole, relevant documents as queries, more important terms are expected to be included in the query. Consequently, the increased term count should help find more relevant documents (increased recall) and disambiguate polysemous terms (increased precision).

Recent research (Harman, 1995a) has shown improved retrieval performance by simply assuming that the top **n** documents are all relevant. A relevance feedback step is performed with those **n** documents before presenting the user with a document return set. This technique is known as pseudo-relevance feedback since it does not involve a user in the relevance judgement. The parameter that must be chosen in this approach is an appropriate value of **n**.

This pseudo-relevance feedback technique is successful for retrieval tasks in which a large recall is important. However, this experiment shows that it is not necessarily effective for question answering (particularly with the WP help collection). Figure 6.5 plots SAGE's performance using various values of **n** from one to 20. Though the plot is congested, particularly at low document cutoff levels, the LSI baseline performance curve can be seen rising above the others for larger cutoff levels.

This result is not too surprising since SAGE's goal is not to return all the relevant documents it can. If a relevant document is found within the top **n** documents, returning more that are like it does not help the performance as measured in this experiment. In this case, the only chance of improving performance is if lower ranked, relevant documents significantly outnumber the higher ranked, nonrelevant documents. Then, perhaps by their superior number, they can rise to a higher ranking. The likelihood of these circumstances occurring seems small and from the experimental data, this situation does not appear to occur frequently enough to be of benefit.

6.5 Passage Retrieval

A technique that has recently proven effective in the information retrieval community is to split documents into sections or passages before creating the index

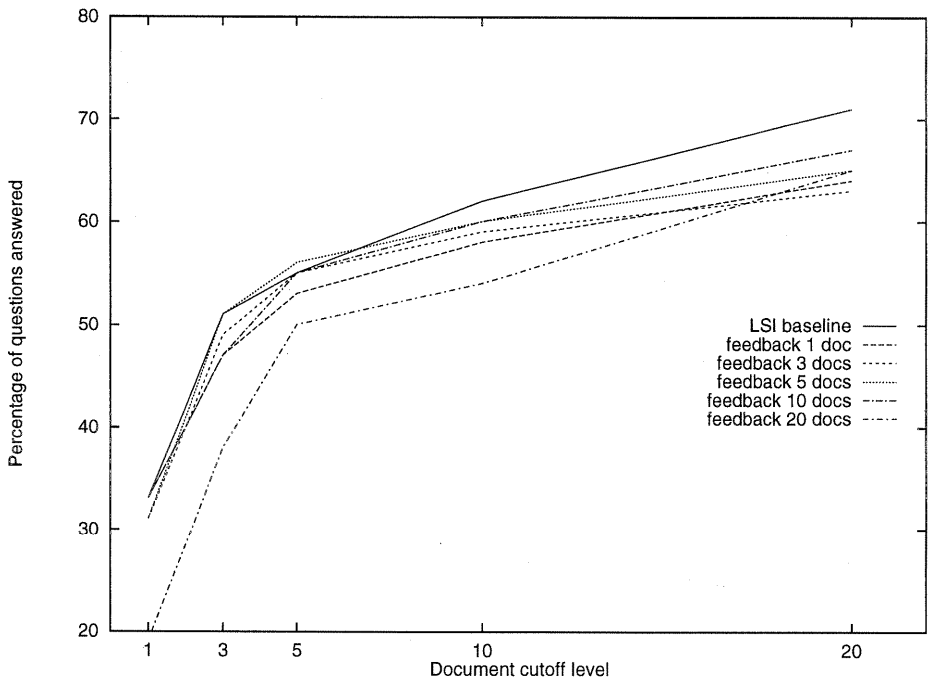


Figure 6.5. Performance of SAGE's LSI index augmented with pseudo-relevance feedback. The top **n** documents (shown in plot key) are assumed to be relevant.

(Harman, 1995a; Kwok, Grunfield, and Lewis, 1995). This technique is of particular use when the relevant information is in one or two paragraphs of a document whose main topic differs from the focus of the question. By breaking the document into smaller pieces, the topic of individual passages is not dominated by the topic of the entire document.

Since some documents in the help collection exhibit this characteristic, it seemed reasonable to believe that this technique would improve SAGE's performance. Some documents contain summary or overview information in which individual sections briefly mention different aspects of a broader topic. For example, the collection has a summary document on fonts with different sections on modifying font styles, sizes, and families. Other documents that contain directions about completing a task typically contain a section describing the purpose of a particular feature and additional sections describing how to use the feature and where to find related information. Treating each section as an individual document could allow the specific information in the section to be found more easily.

In practice, however, the results of passage retrieval with this collection are inconclusive. Figure 6.6 shows that indexing by passages of various lengths degraded SAGE's performance with one exception.

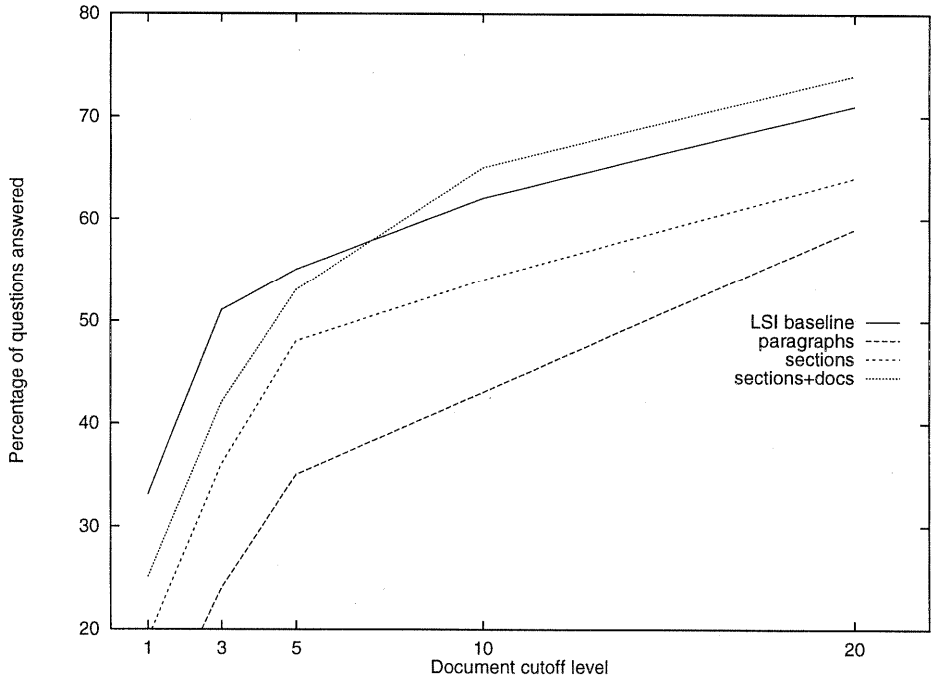


Figure 6.6: Performance of SAGE's LSI index using different passage sizes.

Three different methods of passage indexing were tested. The first method treated each paragraph (or anything that looked like a new paragraph) as a distinct document in the LSI matrix. Paragraphs were easily recognized using the <**P**> HTML tag. This method caused items in lists to each be treated as a separate document. As the results show, this method performed very poorly. Most of the

text passages were very short and consequently, LSI did not have enough material with which to work.

The second method split a help page into sections. Section boundaries were marked by multiple, subsequent paragraph tags. The passages in this method were longer and the text was more logically grouped. For example, using this method, all the items in a list were grouped together in a single section. The performance of this method was an improvement over the paragraph method, but it still fell short of the baseline.

The results for some individual questions using this method were compared with the baseline system. The section method did better than the baseline on some questions though the overall performance is worse. To see if LSI could take advantage of the strengths of both methods, the final passage method tested was a combination of the section method and the baseline system. In other words, the index was constructed from the entire collection of documents plus the sections. Figure 6.6 shows that this method (called **sections+docs**) was still inferior to the baseline for small document cutoff levels. However, a cross-over point occurs near seven returned documents at which the combined method improves over the baseline performance.

The effect size and odds ratio for each document cutoff level of the last section method are shown in Table 6.3. The negative numbers in the first three

Table 6.3: Effect sizes and odds ratios of using sections plus complete documents.

	1	3	5	10	20
Effect Size	-0.68	-0.76	-0.28	0.21	0.31
Odds Ratio	193:1	988:1	14:1	3:1	13:1

columns of the table are indicative of the detrimental effect of adding sections to the index. The odds that the performance degradation was not an abnormality are very high, 193:1 and 988:1 for cutoff levels one and three respectively. The odds that the improvement seen at larger document cutoff levels is worth considering are much smaller, only 3:1 and 13:1 at cutoff levels of ten and 20 documents. Consequently, this technique is not incorporated into SAGE.

6.6 Supplementing with Additional Texts

The reason for supplementing the original help collection with new texts was to provide additional evidence to LSI for discovering synonyms. LSI can more easily uncover synonymous relationships among words with a larger body of related texts because a larger collection typically displays more word choice variability. Also, more evidence for the presence of certain words in particular contexts increases the reliability of the predictions LSI makes with its reconstructed approximation matrix. Based on these observations, the help collections from other popular word processors were added to the WordPerfect help collection before indexing in an effort to improve the performance of questions with mismatched vocabulary.

Many participants in the question collection experiment were regular users of other word processors. Their experience with the language and terminology of

other systems probably affected their choice of words while formulating their questions. Supplementing the WordPerfect help collection with the help text from other word processors could increase the performance of the system by providing a mapping between the vocabulary used in WordPerfect and the vocabulary of other popular word processors.

Figure 6.7 shows SAGE's performance when the help texts from Microsoft Word and Lotus AmiPro are added to the index. The addition of the Word help text did not affect SAGE's performance (though it caused a slight decrease in the middle of the cutoff range examined). However, the addition of the AmiPro collection had a negative effect, particularly for the smaller document cutoff levels. This effect was also seen when Word and AmiPro's texts were both used to augment the LSI index. The reason for this detrimental effect is unknown.

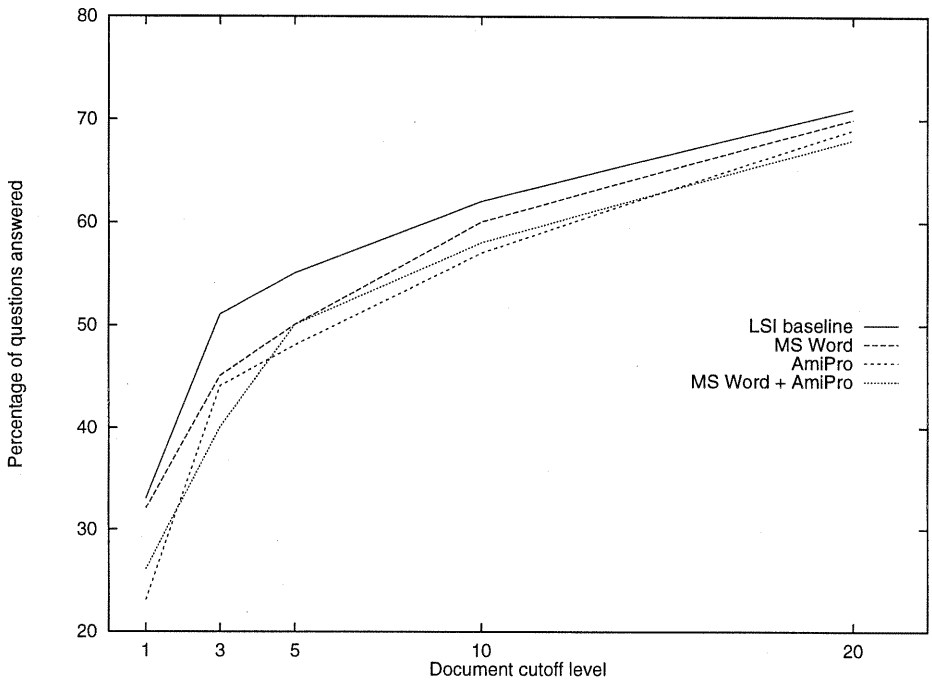


Figure 6.7. Performance of SAGE's LSI index augmented with help texts from Microsoft Word and Lotus AmiPro.

6.7 Discussion

The results of using traditional information retrieval techniques show them to be mostly ineffective in improving performance. Predictably, the two techniques designed to increase recall, query expansion and pseudo-relevance feedback, are unsuccessful in improving the system's performance because the question-answering task is not improved by returning more than one relevant document. The techniques designed to influence the index terms and their weights directly are the most successful. Term weighting (from section 5.5.2) and stemming are two good examples of this effect. The bigram, passage retrieval, and supplemented collection techniques

were attempts to increase performance by modifying or augmenting the document collection being indexed. They did not provide a performance improvement as they were expected to do.

As far as the development of SAGE is concerned, the results described in this chapter suggest that the base system should only be augmented by the addition of a morphological stemmer. Inclusion of document sections in the index might also prove useful for those users who frequently search more than ten documents, but the detrimental effect it has in lower cutoff levels leads to its exclusion in SAGE. Thus, the performance of the information retrieval component of SAGE, designated SAGE-IR, is shown in Figure 6.3.

6.7.1 SAGE-IR Weaknesses An analysis of the errors made by SAGE-IR reveals several interesting insights into its performance and suggests improvements that might be made.

Some questions retrieve irrelevant documents because they contain words that are not related to the subjects' real questions. One subject stated his need for help with the utterance, **"I need to make a text larger and in bold letters."** This sentence contains the words **need** and **make** that are not germane to the real information need. These extraneous words usually affect the returned document set negatively.

Some objects used in constructing a document (e.g., text, tables, etc.) are discussed in many help text documents. SAGE-IR does a good job at ranking some of these documents highly. However, it lacks the ability to focus on both the object and the desired action on that object. For example, a question about creating tables mostly returns documents about tables. However, SAGE-IR judges some documents to be more relevant to the question simply because they are "more about tables" than the document that describes how to create a table. A likely reason for SAGE-IR's judgment is that the density of the word **table** in the highest ranked documents is greater than the combined density of the words **create** and **table** in the correct document.

Another problem is that some highly ranked documents discuss the right topic, but they answer the wrong question. For example, one question in the corpus is **"What is a bullet?"** A document that describes how to add bullets is certainly related to the topic, but it does not define what a bullet is. The glossary page for bullet is the correct page to return. Similarly, a question about adding bullets to the document should not be answered with a definition of what a bullet is.

Finally, section 4.3 alluded to another problem. The answers to some questions are obscure or perhaps indirect at best. The problem is not only a vocabulary mismatch problem. The correct solution to a problem must sometimes be inferred from a text. It is unclear how an information retrieval technique could handle this type of problem.

6.7.2 Intelligent Solutions The problems described above ought to be at least partially overcome by handling the questions more intelligently. For example, doing a simple analysis of the question to remove nonessential words from it should not be difficult. Identifying the nouns and verbs in a sentence would permit SAGE to consider both objects and the actions to be performed on them.

Classification of a document's purpose coupled with an identification of the type or style of information expected by a particular question should increase the likelihood of returning a correct answer.

Each of the suggested solutions can be provided with some very simple natural language processing (NLP) and other artificial intelligence tools. These are developed and discussed in Chapter 7. The final problem mentioned above is a little more difficult to handle with weak NLP methods. A different approach is examined in which the questions submitted by some subjects are used to augment the LSI index. The requirements and limitations of this approach are also examined.

CHAPTER 7

EXPERIMENTS WITH SAGE-NL

SAGE was developed with a loosely coupled architecture designed to permit different components to be "plugged in" without disrupting the rest of the system. The only requirements for components are that they work within the data flow of the system. In other words, when they are plugged into a particular location in the architecture, they must handle the incoming data in the format available at that point in the pipeline and they must produce output of the expected format for the next component. Components can preprocess the collection or a question before handing it over to LSI or they can postprocess the ranked documents generated by LSI.

This chapter extends SAGE with new components employing techniques inspired by the natural language processing and machine learning communities. Two components were added to SAGE to process the document list produced by LSI. They take this ranked list and rearrange the document rankings based on a weak analysis of the language used in the query. A third technique greatly enhances SAGE's performance by preprocessing the document collection to allow LSI to learn from other users' questions.

7.1 Part of Speech Tagging

The first component added to SAGE analyzes and tags the words in the questions with their parts of speech. The part of speech tagger (Brill, 1993) uses the tags from the Penn Treebank. It makes six tense distinctions for verbs, four for nouns, three for adjectives, and three for adverbs. However, the separate distinctions are not used by SAGE. Correct identification of the nouns, verbs, adjectives, and adverbs in the questions is the most important aspect of the tagging process.

7.1.1 Word Elimination The tagged questions are processed in two ways. First, a set of hand-developed patterns is matched against the tagged words to identify and eliminate any expression of an information need. Use of the part of speech tags simplifies and generalizes the patterns. They are also more robust because a pattern does not need to be created for every possible combination of words in the corpus. This is an automation of the question processing described in Section 5.7.1 for identifying key phrases to submit to the WordPerfect help system. For example, in the question **"I'd like to find out how to put in page numbers,"** all the words before the word **put** are eliminated from the question before handing it over to LSI. The removed words often confuse the search process rather than clarify it.

When LSI projects a vector into its high-dimensional space, it uses all the words it is given. By eliminating the information request words from the question, LSI does not create question vectors along the wrong dimensions in its space. If

not removed, the word **find** in the sample question becomes a potential source of confusion for LSI. The WordPerfect help collection contains various documents on finding and replacing text or other items within the document being edited. However, this information is not what the user was asking for. Nevertheless, the question vector created by LSI projects some distance along the "find" dimension unless **find** is removed from the question. Removing it significantly lessens the chance of LSI returning documents about finding and replacing text.

7.1.2 Document Reranking The second use of the tagged questions is to make a distinction between objects in the interface and the actions applied to them. Depending on the global weights of the words used in a question, either the object or the action is likely to be favored in the ranked document set. The tagged words are used in a postprocessing operation to form a new query designed to balance the selection between objects and actions. This new query is used to (possibly) reorder the top ranked documents before returning them to the user.

When LSI produces a ranking of the entire collection of documents, it is based on each document vector's similarity to a single question vector created from the terms in the question. Its direction in the LSI space is based on the global collection weight each term was assigned at the time the space was constructed. The important point here is that some terms in the help collection are weighted more heavily than others and they project more strongly along their related dimensions, thus dominating lower weighted terms when similarity comparisons are made.

Consider the question "**How do I center a table?**" The weight-bearing words are **center** and **table**. When this question is submitted to LSI, the top-ranked documents are about centering different objects and performing various operations on tables. However, the problem with the ranking is that the document on centering lines is judged to be more relevant than the document on table formatting because the weight of the word **center** is twice the weight of **table**. Although the table formatting document contains both concepts asked about in the question, it is not the highest ranked document. LSI appears to favor documents strongly focused on higher weighted terms (i.e., centering) over those that contain both topics. With a small set of candidate documents identified, the search for an answer needs to be refocused on the fact that both the object and the action are important in answering the question. A solution to this problem is to isolate the top twenty documents and ask LSI to rerank them using new weights for the terms.

The steps of this reranking algorithm are:

(1) Tag words with their parts of speech.
(2) Group nouns and adjectives together and verbs and adverbs together.
(3) Compute the average weight of each group.
(4) Find the ratio of the noun to verb groups' average weights. Label it X:Y.
(5) Create a new word list that contains Y copies of the noun group and X copies of the verb group.
(6) Create a vector in LSI space from this list of words.
(7) Rerank the top 20 documents returned using the original question based on their similarity to the new vector.

SAGE identifies nouns and their modifiers and verbs and their modifiers. It groups

the nouns and adjectives into one set of words and verbs and adverbs into another. Next, the idea is to adjust the weights of each group so that they are (approximately) equal to each another. This component is external to LSI and cannot change the term weights stored in the index, so it must reweight terms in a different way. The adjustment is done by creating a new list of words in the right proportions so that, when weighted, the total group weights are equal. This is accomplished by computing the average weight of each group. Since the local weighting algorithm used in SAGE is simply a term's frequency, the inverse ratio of these values dictates the number of copies of each group to include in the final word list submitted to LSI. In the sample question, the ratio of the weights of **center** to **table** is 2:1. Therefore, a new query is created with two copies of the word **table** and a single instance of the word **center**. The net effect is that LSI creates a vector that is equally weighted between the "center" dimension and the "table" dimension. The new vector refocuses the search to look for documents that discuss both topics of interest (i.e., those that discuss table centering).

To summarize, a user's initial question is used to generate a complete ranking of the documents. This step serves to narrow the list of documents to the right topic(s). Next, a new query is formed that equalizes the importance of objects and the actions to be performed on those objects. A vector for the new query is projected into the LSI space and the top 20 documents from the original ranked list are reordered based on their similarity to the new query vector. Finally, the reordered document list is presented to the user.

7.1.3 Results Since this component is limited to use only the top 20 ranked documents, it cannot possibly give answers that are not already found by LSI within the sampled document cutoff level. The performance expected from this component is an improvement in the ranks of the correct documents. In information retrieval terminology, this component is designed to increase precision without affecting recall. Figure 7.1 plots the performance of SAGE using the reordering technique described here. The other curve is the version of SAGE produced at the end of Chapter 6. The figure shows that the correct answer is reranked to position number one for an additional 4% of the questions. Table 7.1 shows the effect sizes and odds ratios for the expected improvement using this component. Based on the

Table 7.1: Effect sizes and odds ratios of the part of speech tagging technique.

	1	3	5	10	20
Effect Size	0.38	-0.02	-0.03	-0.03	$\approx$0
Odds Ratio	41:1	0	0	0	30:1

performance curve, it is not a surprise that the only significant difference occurs in the first ranked document. While the effect size is low to moderate at this cutoff value, the odds, 41:1, are significant that a real improvement in performance exists in this position.

The high odds ratio at 20 documents is a side-effect of the fact that the total number of questions answered correctly cannot change with this technique.

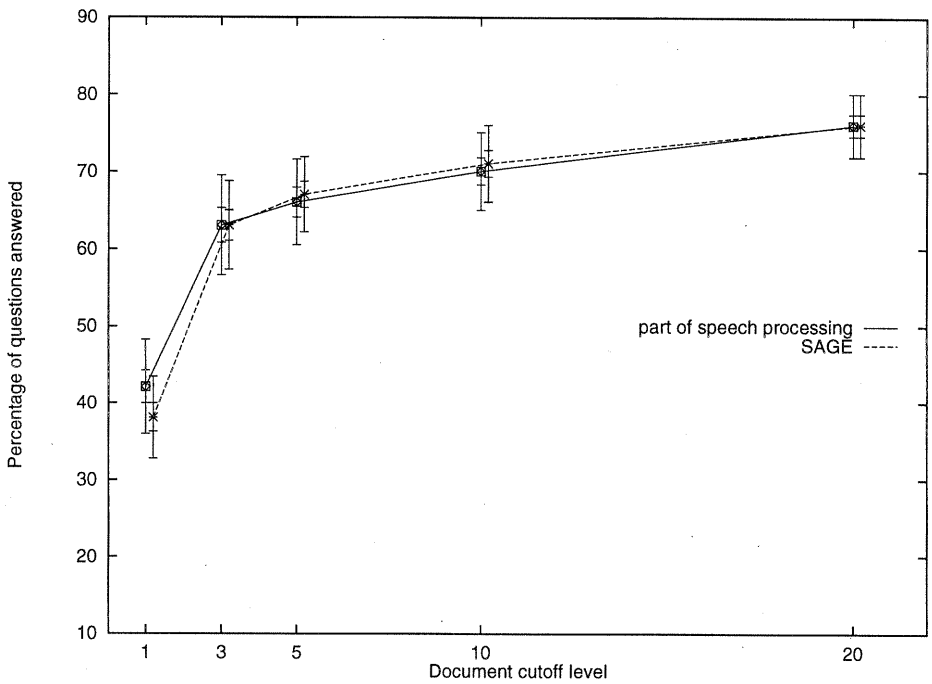


Figure 7.1. Performance of SAGE using part of speech tagging to reorder the top 20 documents.

Recall that the scores being compared contain a decimal portion representing the number of questions answered correctly and a fractional portion that represents the average correct document rank. This technique cannot affect the decimal portion of the score at the 20 document cutoff level because it does not change the contents of this set. Only the ranks can change at this level and since they have little influence on the system's total score, the differences in scores between the two systems are consistently small. Furthermore, the effect size at cutoff level 20 is affected similarly. The effect on the total number of questions answered (the dominant portion of the score) should be zero. Thus, the real question should focus on the technique's effect on the ranks of the documents. Since the decimal portion of each score is the same for all users at the 20 document cutoff level, it can be eliminated to compute the effect size for the average rank of a correct document. The effect size computed in this manner is 0.25. Finally, the odds ratio can be interpreted as 30:1 odds that this technique has a small, positive effect on the average rank of the correct documents.

This technique has now been added to SAGE and the performance curve shown in Figure 7.1 is considered the new standard. The next technique developed compares its performance with this new version of SAGE.

7.2 Document Type Filtering

An examination of the ranked documents produced for various questions reveals that sometimes the top-ranked document is related to the topic of the question, but the purpose of the document does not address the focus of the question. For example, the top-ranked document returned for a question about creating bullets is a document from the glossary defining what bullets are. This example shows that improvements in SAGE can be made by ensuring that the returned documents' purpose or function addressed the type of question asked. A postprocessing component based on this idea is created, plugged into SAGE's architecture, and tested.

Before describing the implementation of this technique, it is important to establish that the goal in this section of the research is to explore the idea of filtering based on document types rather than develop a general method of identifying a document's purpose. The focus is on whether filtering the return set by document type is possible and what effect it has on the performance. To simplify this exploration, portions of the implementation are quite specific to this help collection and question corpus. An open issue remains to discover whether document purpose classification can be done on a general scale.

7.2.1 Identifying a Document's Purpose

A small document classifier was built to identify the purpose of a document. SAGE knows about five different document types. The different document types were assigned the following names:

- Definition
- Hint
- HowTo
- Navigation
- Summary

The classifier uses a few hand-coded rules and it relies heavily on the consistencies found in the format and language of the different types. Each document type and the method for identifying it are briefly described below. The document types are tested in the listed order so once a type is identified, it can be reported. Thus, a condition for each type is that it is not one of those listed before it.

Definition documents define terms found in the glossary. These are easy to identify in the WordPerfect help collection because they contain the string "(def)" in their titles.

Hints are usually short pages of text with little tidbits of additional information that may be of interest to the reader. They either contain the word "Hint" in them or they contain fewer than 100 words.

A HowTo document describes how to use a feature or accomplish some task. This is the most important class of documents to SAGE. Documents that contain a purpose statement and a list of steps to follow are classified as HowTo documents.

Navigation documents are designed to help the users find their way around the collection. Examples of navigation documents are table of contents and glossary pages. Other navigation documents are those that briefly list the contents of a menu and provide links to more detailed documents (usually HowTo documents) that describe the features available from a particular menu hierarchy. Navigation documents are recognized by their high ratio of words in hypertext links to total words. SAGE uses a threshold of 70% hypertext link words to classify a document as a navigation document.

If a document is not recognized as one of the previous four types, it is assigned the Summary classification. Typically, these documents provide a high-level overview of a set of related features. For example, one gives an overview of fonts and briefly describes the attributes of fonts, such as their size, family, and style. Occasionally they contain enough information to answer users' "How do I ..." questions.

Besides identifying the different document types, SAGE needs to identify the type of question being asked so that it can make sure the documents provide the expected information. Question type tagging is even more simple than document classification. Question types are identified by searching for an instance of any of the words **how**, **what**, **why**, **where**, or **when** (in that order) and marking the question type according to the word found. If none of these words occur in the question, the question is assumed to be a how question. This decision was based on the observation that most questions submitted to an application's help system are requests for assistance in accomplishing some task. As further evidence of the validity of this assumption, each question in the corpus that does not contain one of these words can loosely be interpreted as a request for information on how to perform an action. Examples of the other types of questions in the corpus all contain the corresponding "Wh-" word.

SAGE implements document filtering by examining the top 20 ranked documents returned by LSI and the part of speech postprocessing component. Navigation documents are alway eliminated from this set because they are not designed to answer questions. Definition and Hint documents are also eliminated if the question is

a How question because they do not contain procedural information. Furthermore, How questions cause the LSI similarity score of HowTo documents to be increased by 10% and Definition documents receive a 10% boost in score for What questions. It should be noted that the decision to boost scores 10% for document and question type matches was made arbitrarily. Early experimental results showed that a boost would improve performance, but the value 10% was chosen simply because it proved to be adequate, not because it was rigorously tested to select an optimal value.

7.2.2 Results The performance of document filtering is shown in Figure 7.2. The other performance curve in the figure is the version of SAGE that includes the part of speech processing developed in the previous section. Filtering

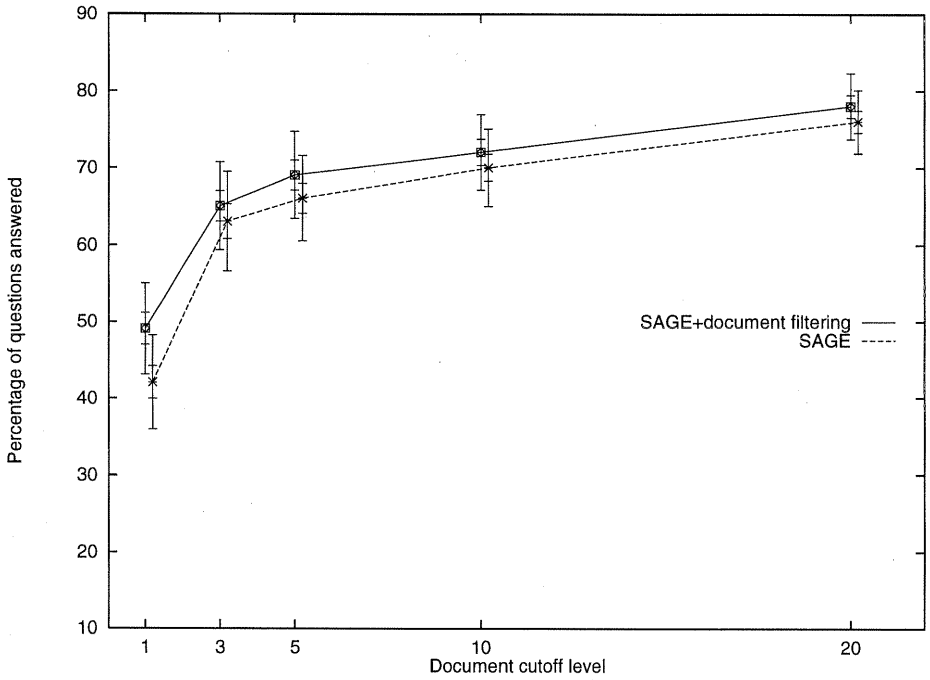


Figure 7.2. Performance of SAGE using document type filtering. In this figure, SAGE includes part of speech processing.

out incorrect documents and boosting those more likely to be correct effect a positive increase in performance at all cutoff levels. The effect size is moderate in the first document position and small at the other positions (see Table 7.2). The odds ratio

Table 7.2: Effect sizes and odds ratios of the document type filtering technique.

	1	3	5	10	20
Effect Size	0.47	0.18	0.18	0.16	0.14
Odds Ratio	1×10^3:1	66:1	66:1	88:1	4:1

is extremely high in the first position and high in all the others except at position 20. Therefore, it is extremely likely that document filtering is effective at increasing SAGE's performance.

7.3 Learning From Users

Some of SAGE's failures are the result of word mismatches between the help text author's description of a feature and the users' request for it. This problem is known in the information retrieval community as the vocabulary problem (Furnas et al., 1984; Furnas et al., 1987; Chen, 1994).

7.3.1 Addressing the Vocabulary Problem Furnas proposed an adaptive indexing scheme (Furnas, 1985) to overcome the vocabulary problem. In this scheme, new indexing terms are added to an on-line index incrementally by users of the index. The system remembers each query submitted by users trying to find some information. After the information has been retrieved, the system asks the users if the terms that were unsuccessful in finding the information should be added as index terms for that particular information. Thus, new searches may be aided by the work of previous users.

An adaptive index may work well for a community index, but it is unlikely to help much in an application designed to run on a single-user machine. When only a single user is using the index, it must come prepackaged with the additional vocabulary to help a novice user. Therefore, data must be collected from a representative set of users before building the index. Then the data can be added to the text collection and used during construction of the index.

The development of SAGE has been guided by just such a corpus of data. Before indexing the help text collection with LSI, the questions submitted by users during the WOZ experiment are added to the documents relevant to those questions. Thus, LSI can learn new mappings between terms and documents with the augmented collection. Furthermore, LSI is given the information necessary to make inferences about synonymous relationships between words. When one user asks about centering titles and that question is added to the document on centering text, LSI can infer that titles are related to text. When a future user asks about inserting a title, LSI is more likely to return the document on inserting text.

7.3.2 Results An important principle taught by the machine learning community is that a system should not be tested with the same data used to train it. Consequently, a cross-validation methodology (Russell and Norvig, 1995) was used to train and test this learning method. The questions from one user were withheld as a test set and SAGE was allowed to learn from the questions supplied by the other 16 users. The performance on the withheld subject's questions was computed and recorded. This process was repeated until the data from each of the 17 users had been used as a test set. Finally, the results from the 17 different runs were averaged. Therefore, the results presented in Figure 7.3 may be viewed as the average performance of learning from 16 users. The learning technique improves SAGE's performance considerably. As expected from the difference in performance curves, the effect sizes and odds ratios (shown in Table 7.3) at all document cutoff points are significant. The effect size is larger than one standard deviation for all cutoff points. The odds ratios are so high that this technique can be considered a sure bet to improve performance.

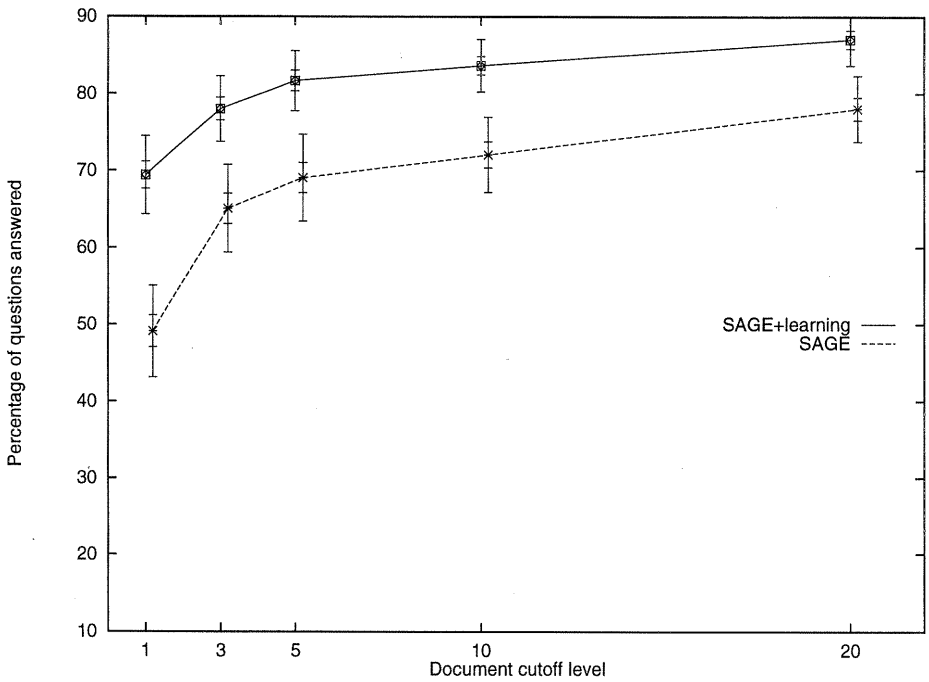


Figure 7.3. Performance of SAGE after learning from other users. In this figure, SAGE includes the part of speech processing and document filtering techniques.

Table 7.3: Effect sizes and odds ratios of the learning technique.

	1	3	5	10	20
Effect Size	1.65	1.08	1.08	1.15	1.07
Odds Ratio	3×10^6:1	2×10^6:1	1×10^6:1	690×10^3:1	120×10^3:1

7.4 Learning Sensitivity Analysis

The results of this experiment show that SAGE performs very well after learning from users. However, system designers who wish to employ this technique are faced with some important questions. First, how many users should be employed during data collection? Also, collecting questions from subjects doing every possible application task available is difficult. So the second important question is, what effect on performance does this technique have when users ask questions for which learning has not taken place? This section seeks answers to these questions.

7.4.1 User Effects The results shown in Section 7.3.2 were produced using the data from 16 users because that was how much was available from the Wizard of Oz experiment. However, it may not be practical (or necessary) to collect data from that many users. Average performance was computed with data from four, eight, and 12 users to compare how the results change SAGE is given more data from which to learn.

The training and testing data were obtained by placing subjects' questions into different sized groups depending on the desired training set size. For example, to compute the results for eight users, the subjects were divided into a group of eight and another of nine. Data from the first group was used to train SAGE and the second group was used as the test material. The roles of the two groups were then reversed to compute results for the first group. The scores from both experiment runs were averaged to produce the performance curve.

A similar procedure was followed for computing results with four and 12 users. With four users, subjects were divided into four groups and then each group was used separately as training data while the others were used as testing data. Results for 12 users were obtained with the same four groups with the exception that each combination of three groups was used as training material and the remaining group was withheld for testing.

The performance obtained with each of these methods is plotted in Figure 7.4. The performance curve obtained from SAGE without learning is also given for comparison. Error bars for these curves are not given because the plot becomes too cluttered when they are included. Each set of effect sizes and odds ratios given in Table 7.4 represent incremental differences as more learning data is used. In other words, the first group of statistics reflects the performance difference in going from zero to four users as the training set size. The second group summarizes the performance change between four and eight users and so forth until all 16 data sets are included.

Though the effect size of learning with four users is not as large as for 16 users, the odds that there is a performance improvement are still considerable. Using

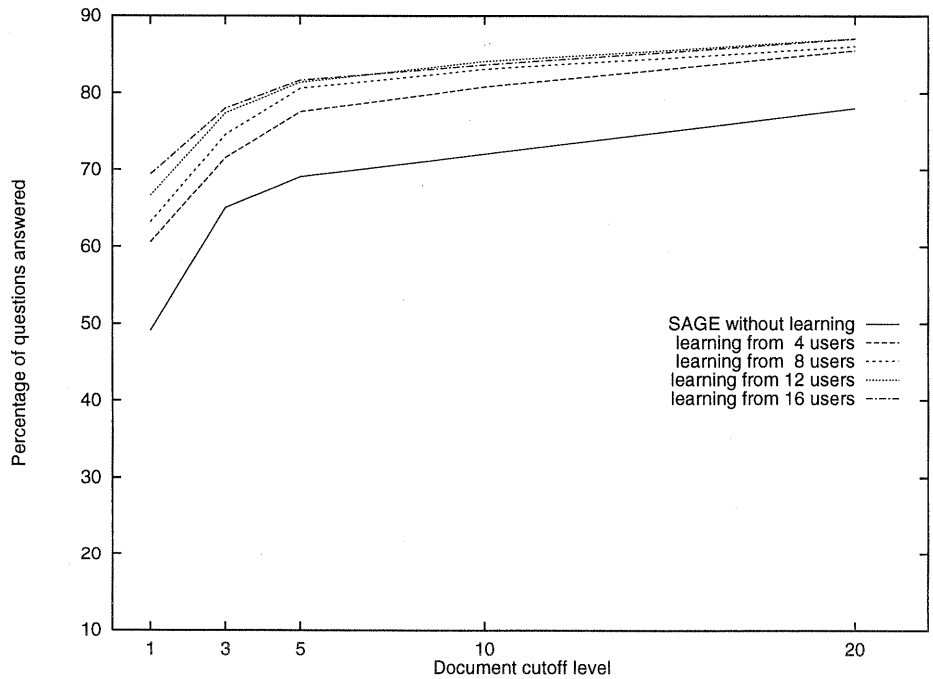


Figure 7.4. Sensitivity analysis of SAGE's performance based on learning from different numbers of subjects. In this figure, SAGE includes the part of speech processing and document filtering techniques.

Table 7.4. Incremental effect sizes and odds ratios from adding training data of additional users.

Learning From Four Users					
	1	3	5	10	20
Effect size	0.92	0.54	0.72	0.84	0.88
Odds ratio	37×10^3:1	688×10^3:1	85×10^3:1	278×10^3:1	171×10^3:1

Learning From Eight Users					
	1	3	5	10	20
Effect size	0.19	0.26	0.34	0.31	0.05
Odds ratio	29:1	110:1	783:1	327:1	1:1

Learning From 12 Users					
	1	3	5	10	20
Effect size	0.29	0.65	0.17	0.16	0.19
Odds ratio	322:1	17:1	11:1	2.7	6:1

Learning From 16 Users					
	1	3	5	10	20
Effect size	0.2	0.09	0.01	-0.03	0
Odds ratio	11:1	3:2	0	2:1	0

data from eight subjects provides another small to moderate improvement and the odds that the effect is not spurious are also significant. Except at the first document cutoff level, the performance difference begins to tail off when four more data sets are added. Finally, the differences between learning with 12 and 16 users are negligible. Thus, these results suggest that collecting training questions from 16 users is more than necessary. Some number between eight and 12 is probably ideal, but even as few as four still provides significant performance improvement.

7.4.2 Task Effects The other important question probes the effect learning on some tasks has on the performance for novel tasks. A cross-validation learning experiment was used to compute this answer also. The cross-validation was run by withholding all the questions submitted for a given task as a test set and using the remaining questions as learning data. Again, these steps were repeated for all the tasks and an average performance was computed.

The results are shown in Figure 7.5. Note that the comparison curve given

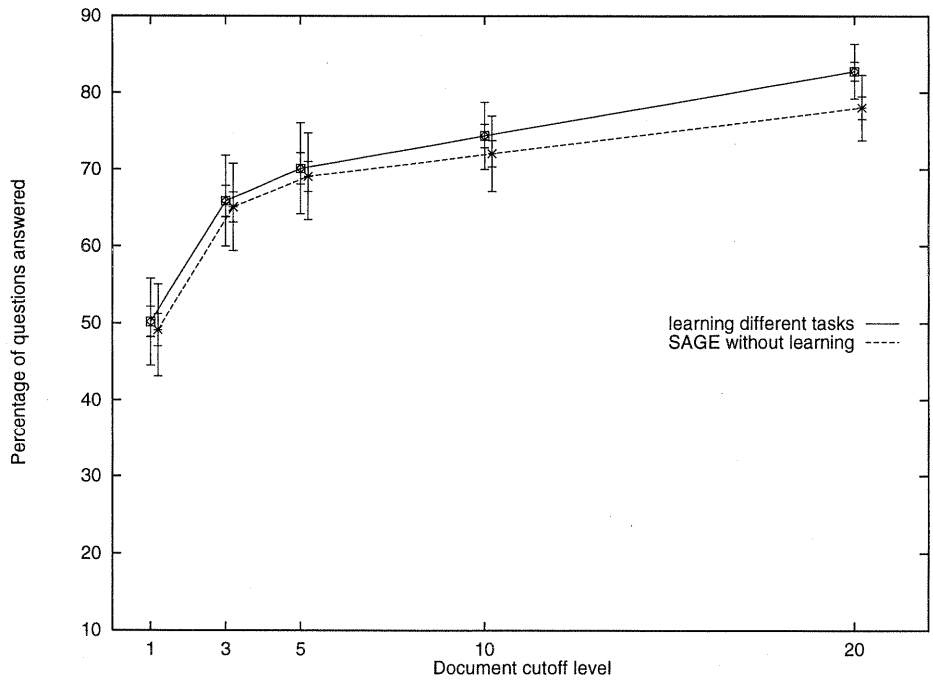


Figure 7.5. Sensitivity analysis of SAGE's performance on novel tasks when learning has taken place. In this figure, SAGE includes the part of speech processing and document filtering techniques.

in the figure is the performance SAGE attained with document type filtering (from Section 7.2). This curve is used for comparison rather than the learning-based curve of Figure 7.3 because both plot the performance of SAGE on tasks for which learning has not occurred.

An interesting effect can be seen in the figure. When SAGE learns from questions submitted by users doing some tasks, a positive effect takes place such that performance is improved for questions about new tasks at the larger document

cutoff levels. The effect size at 20 documents is a moderate 0.56 (see Table 7.5). The odds ratios of a difference in performance at the two larger cutoff levels are

Table 7.5. Effect sizes and odds ratios of the performance on novel tasks after learning on different tasks.

	1	3	5	10	20
Effect Size	0.07	0.05	0.14	0.24	0.56
Odds Ratio	1:1	1:1	12:1	66:1	372:1

also significantly large. However, differences in performance are minor up through the first five documents returned. This small effect is almost certainly a result of the way the corpus was segmented during training and testing. Segmentation was done by assigned tasks, not necessarily by question contents. For example, the term **highlight** was used ambiguously in the corpus. In most cases, users meant **select** when they used it. However, one user referred to bold text as highlighted text. Since selecting text and making text bold were two different tasks, one appears in the training set for the other and has a small affect on the results. The most important result of this experiment is that performance for novel task questions is not negatively affected by learning on other tasks.

7.5 Summary

SAGE's ability to answer questions accurately was extended by the addition of several postprocessing components. These components perform some weak natural language processing of the questions submitted to the system. The first technique uses the question words tagged with their part of speech to create a new LSI vector with a more balanced emphasis on both objects and the actions applied to them. This new vector is used to rerank the top 20 documents identified with the original question. The second technique looks at the question to determine the type of document that best addresses it. Then SAGE compares highly ranked documents' types with the expected type. It eliminates those with a poor match and increases the similarity for documents with an excellent type match. Both techniques show a moderate performance effect at the first document position with significantly high probability.

SAGE was extended further by permitting it to learn from the questions submitted by users. Performance using this technique increased significantly at all document cutoff levels. A sensitivity analysis of the effect learning had on performance was performed. The results from this analysis suggest that an ideal number of training questions can be gathered from eight to 12 users. Furthermore, the analysis showed that when SAGE learns from questions about a subset of tasks, performance is not degraded when users ask about new tasks.

CHAPTER 8

SAGE EVALUATION

The final evaluation of SAGE was an experiment conducted with real users. The experiment was designed to be similar to the Wizard of Oz (WOZ) experiment described in Chapter 3.

This experiment had goals similar to those of the first experiment. They were:

- To evaluate subjects' behaviors and attitudes toward the implemented system and
- To collect a new, but related corpus of questions to test SAGE's performance.

The WOZ experiment simulated a nearly perfect help system. The "system" knew what the tasks were. The help text was tailored to the tasks and the subjects usually received an answer that helped them solve the task on their first attempt. Since SAGE does not work as well as the simulated system, assessing whether subjects' reactions to it changed was important. Additionally, the texts used by SAGE were obtained from an external source and thus, they are not specifically written with the knowledge of the tasks being solved in the experiment. Users' attitudes toward the system may be affected when they are given less specific responses to their questions.

Furthermore, the development of SAGE was heavily based on its performance using the first corpus of questions. Evaluation with a new corpus of questions provides assurance that it is not too biased toward the original set of questions. Comparison of the performance on the new corpus with the original also provides evidence that the WOZ experiment produced a representative collection of questions.

8.1 Experimental Conditions

Though the experiment was designed to be similar to the original WOZ experiment, a few modifications were necessary to use WordPerfect as the word processor. Since much of the experiment was similar to the first WOZ experiment, the reader is referred to Chapter 3 for any details omitted here. This section provides a high-level description of the experiment as a reminder and details only the relevant differences between the two experiments.

Two relevant differences were the result of suggestions made by subjects in the first experiment. In that experiment, subjects had to press a button to alert the system before asking a question. In this experiment, the users alerted the system by first speaking the word **computer** and then asking their question. The second modification was the inclusion of a message to inform the users that the system was searching for an answer. This message was a signal to the users that they had been heard and the system was processing their question.

8.1.1 Participants Seventeen volunteer subjects were recruited from among members of the campus and local communities. Subjects were paid $10 for their participation. As before, seven subjects were women and ten were men. However, subjects in this experiment had more relevant experience than in the original. Subjects' experience with any word processor ranged from two to ten years and they had from zero to ten years experience with WordPerfect. One male subject with ten years experience knew how to do each of the assigned tasks. He completed all of them without using SAGE. Since no questions were collected from this subject, the results reported are based on the other 16 subjects.

All but one of the seven wizards helping with this experiment were paid volunteers as well. Two of them each transcribed for a single subject and the others transcribed for three subjects. The impact of using so many different people as wizards cannot be completely assessed. However, since their only role was to type what they heard, their effect on the results was expected to be minimal.

8.1.2 Tasks Most tasks used in the experiment were taken from the WOZ experiment. Tasks 2, 11, 12, and 13 (see Table 3.1) were eliminated and three new tasks were added. The new tasks required the users to:

- add a double border to the heading row of a table,
- adjust the vertical page margins, and
- add a heavy, double separator line to the document.

The last two tasks were the last ones given and so not all subjects attempted them.

8.1.3 Hardware Components The hardware used in the experiment consisted of the same audio listening and recording devices used during the first experiment. Again, the subjects were outfitted with a microphone into which they asked their questions. A cable ran to the back of the computer and then to a portable cassette recorder that recorded the session. Headphones plugged into the recorder permitted the wizard to listen to the subjects' questions.

In this experiment, the subjects worked with two displays. The first was a VGA color display attached to a PC computer. This was the primary work space. The subjects used WordPerfect's word processor running on this machine to do the assigned tasks.

The second display was a monochrome Sun workstation used to display SAGE's responses. This configuration was necessary to permit SAGE to display texts through remote control of the display interface. Establishing the required communication and application control with the PC was impractical. SAGE's responses were displayed in Netscape, a popular browser for the World Wide Web. Netscape running within X Windows has a remote control feature[1] that permitted SAGE to post answers to the users' questions automatically.

8.1.4 Software Components The WordPerfect help text was converted to use the HTML markup language. In this format, Netscape could display the text such that its appearance was similar to the original. SAGE responds to users' questions by generating an HTML page that contains the question (entered

[1] See the WWW page http://home.netscape.com/newsref/std/x-remote.html for information on how to control Netscape remotely.

by the wizard), a list of the 20 top-ranked documents, and the text of the top-ranked document. Figure 8.1 shows the response page layout. The users' questions appear in an edit field at the top of the page. This field could also be used to submit written questions to SAGE. Below that is a list of document titles corresponding to the top-ranked documents. This list resides in its own scrollable frame. The bottommost frame occupies two thirds of Netscape's display space. This is independently scrollable to accommodate long help messages. Initially, it contains the help text from the first document in the return set. When users select other documents by clicking the mouse on their titles, the contents of the selected document replace the text in the bottom frame.

The wizards in this experiment used a command line interface to enter subjects' questions. They were instructed to transcribe the questions exactly as they were spoken. Spelling accuracy was a concern because incorrectly spelled words affect SAGE's performance and cause users to wonder why the system is not more accurate. Consequently, the typed questions were processed through the Unix spelling checker, ispell, before being sent to SAGE. Any detected errors were reported to the wizards who were asked to correct or confirm the spellings. While this arrangement is not foolproof, it caught most of the potential errors.

When the wizards pressed the **Return** key to send the question, the subjects received an initial message (in Netscape's window) that SAGE was searching for an answer. This feedback informed the subjects that they had been heard and the system was responding to them. Preliminary feedback of this type was important because the time between asking a question and receiving an answer was 20 seconds on average. The delay resulted from the time required for the wizard to type the question, process it through **ispell**, possibly make corrections, send it to SAGE to compute the ranks of all documents, build a Web page, and send it over the network to Netscape.

8.1.5 Procedure The instructions and description of the experiment given to the subjects were similar to those given in the first experiment. During the first five minutes of the session, subjects read a short description of the experiment in which they were encouraged to ask for help when they needed assistance completing a task. Subjects were instructed to speak the word **computer** before asking questions to alert the system that it was being addressed. They were guided through an interaction using a sample question about running the spelling checker. During this instruction, the various components of the display page were described and demonstrated. The subjects also had a chance at this time to experience the delay expected in receiving an answer.

When the subjects asked their questions, one of seven wizards listened and transcribed the question for SAGE. The difference in their typing speeds was negligible compared with the total response time. However, the wizards' typing and spelling accuracy had a small, observed effect on the subjects' behavior. These effects were caused by spelling errors in three questions (one each for three different subjects). The words were flagged as misspelled by **ispell**, but for two of them, the correct spelling was not provided and the wizard did not take the time to try alternate spellings. With the third error, the wizard sent it to SAGE by mistake.

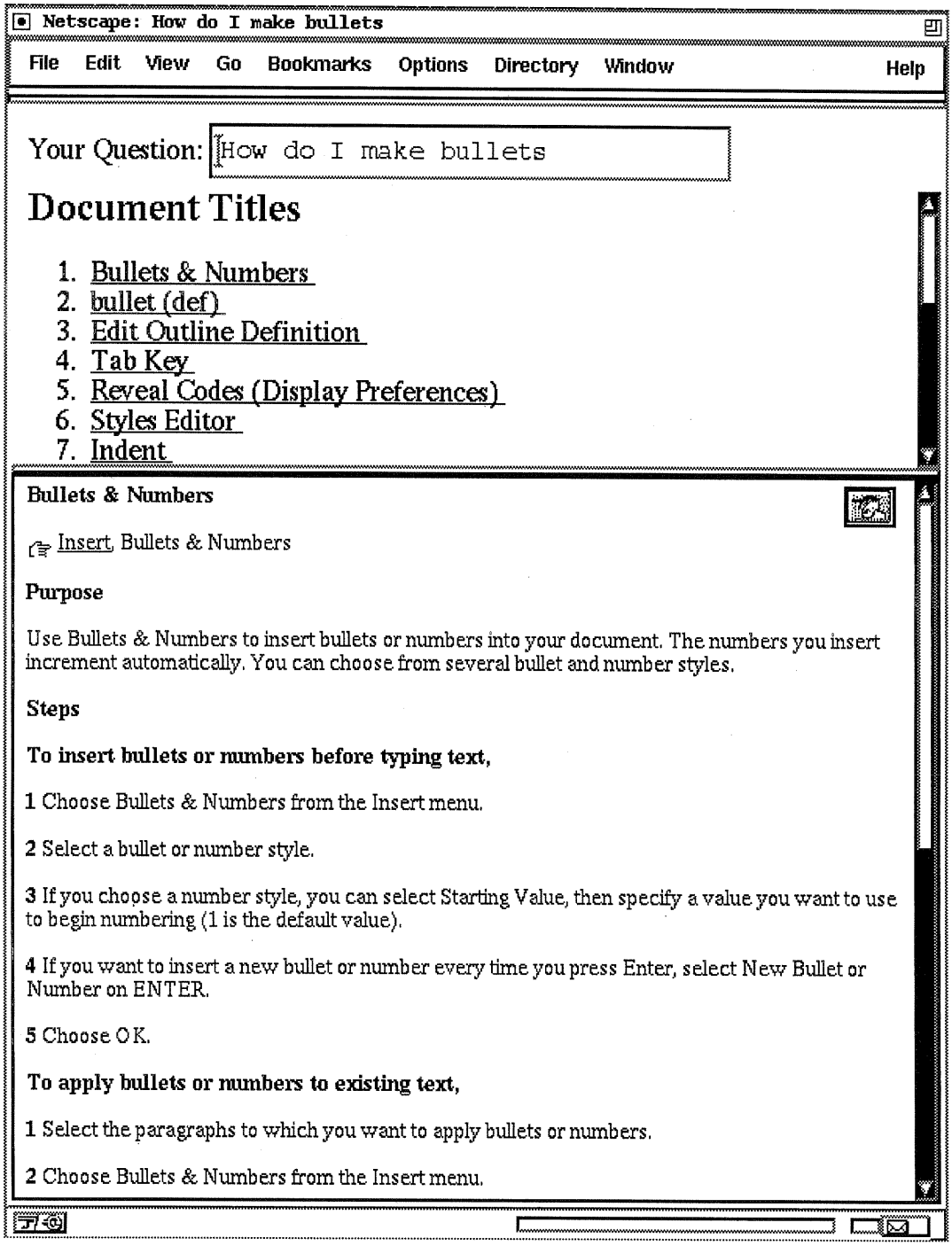


Figure 8.1. A view of Netscape showing the page layout of SAGE's response to a sample question.

The misspellings appeared to influence the subjects to annunciate more clearly. One also spoke more deliberately after the error occurred. One of the others became temporarily stuck because she thought the misspelling was a result of the system's ignorance of the term and she struggled to think of a synonymous word to use in its place. After a short delay, she rephrased her question, received a helpful response, and completed the task. Presumably, using fewer wizards should reduce the spelling inconsistencies.

After the subjects finished the tasks or when 45 minutes had passed, a short interview was conducted. The subjects were asked about their general impressions of the system, how they would compare it with other help systems they had used, and what they liked and did not like about it. This interview lasted no longer than ten minutes.

8.2 Results

Subjects completed more tasks during this experiment, but submitted fewer than half the number of questions compared with the first user study. They submitted 110 questions during the completion of 151 tasks. Another 16 tasks were attempted but not finished. Questions were asked at a rate near 0.66 questions per task, or about two questions for every three tasks. Even this value is a little misleading because some tasks evoked more than one question and many others were completed without assistance from SAGE. Of the completed tasks, half (76) were done without assistance.

This rate is well below the rate seen in the preliminary experiment. Overall, subjects in this experiment had more computer background (average of six years) and word processing experience (average of five years) than in the first experiment. However, the largest factor contributing to the reduced usage of the help system was probably the difference in word processors used. A majority of the subjects were familiar with Microsoft Windows-based word processors. The designers of these applications try to follow certain look and feel conventions that make using a new one much easier to learn. Subjects' experience with other Windows-based word processors permitted them to complete many tasks by looking in the menus where they would expect to find a feature in a similar word processor. In contrast, the use of FrameMaker running in the X Windows environment was more foreign to the subjects. They did not feel as comfortable searching for a feature because the interface was too unfamiliar to provide an adequate starting point for their search. Consequently, they turned to the help system more quickly than their counterparts in the second user experiment.

8.2.1 System Performance The questions collected in this experiment were assigned relevant documents in the same way described in Section 4.3. During the experiment, two subjects completed tasks using a document that was not considered relevant. In creating an indented, bulleted list they discovered that they could follow SAGE's advice and change the bullet style to produce the desired appearance. During the initial round of relevance judgments, this was not considered a reasonable solution. Therefore, to maintain consistency with the results computed during the development of SAGE, this document was still not considered relevant.

With the relevant documents identified, the raw performance of SAGE was computed for these questions. The performance graph is shown in Figure 8.2. Performance using the original corpus is also plotted for comparison. SAGE does not

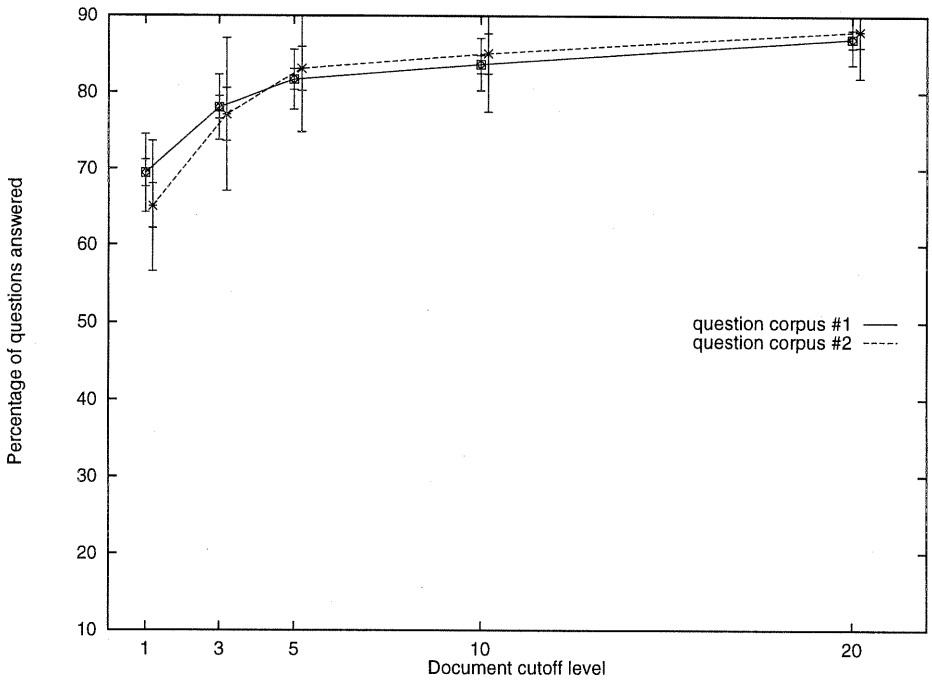


Figure 8.2: Performance of SAGE on a new corpus of questions.

do quite as well at the first sampled cutoff level with this new corpus of questions. However, a very respectable 65% of the questions are answered by the top document and 77% of them within the first three. By the time three documents are examined, performance on the two different corpora is essentially identical. Thus, SAGE is effective at finding answers to new (but similar) questions and it can be effective for users of different experience levels.

The statistical analysis of the differences between the two corpora is shown in Table 8.1. In this analysis, the test corpus changed rather than the system. Consequently, the odds ratios presented here were computed using a t-test for two samples assuming equal variances rather than the paired-means t-test used in previous chapters. As Figure 8.2 shows, a small, negative effect occurs at the first document cutoff

Table 8.1: Effect sizes and odds ratios of the difference between the two corpora.

	1	3	5	10	20
Effect Size	0.2	0.09	0.01	-0.03	0
Odds Ratio	11:1	3:2	0	2:1	0

level using the new corpus. The odds are reasonable, but not significant, that there truly is a difference at this position. Both the odds and observed effect size are

very low (3:2) after three documents are examined. Beyond that point there is no observed difference and the odds that a difference exists drop to zero.

8.2.2 Task Completion The real goal of a help system is to assist its users in completing their tasks. Ninety percent of all tasks attempted were completed whether the subjects needed help or not. For those tasks on which users requested assistance, the completion rate was just above 82% in this experiment. Users asked questions for 91 tasks and they completed 75 of them. Four of the sixteen subjects failed to complete two tasks and eight others did not succeed in completing one task. The remaining four subjects completed all the tasks they attempted.

Various factors affected the subjects' ability to complete certain tasks. These factors can be classified into four different causes of breakdown in the task-execution, help-request interaction. The causes were:

(1) **No help given.** In this case, four users tried to find some desired information, often with more than one information request. However, vocabulary mismatches between the subjects' questions and the WordPerfect help text prevented SAGE from finding the correct answer. For example, one user trying to create a table asked about boxes. SAGE's answer described how to create graphics boxes. The subject recognized that this information was not helpful and decided to stop doing this task.

(2) **Wrong question asked.** This category is similar to the previous one, but here, SAGE's answer is close enough to what the subjects think they want that they are misled into thinking that they have found a solution. Two users asked about table styles when given the task to modify a table's border. The help text they received outlined how to apply one of the predefined table styles. This answer looked promising and it would work for some tables. However, none of the predefined styles provided the desired look, so this response did not help. At this point, the subjects chose not to pursue a different solution and they gave up on the task.

(3) **Correct answers not recognized.** In nine of the sixteen uncompleted tasks, SAGE provided the best document available within the top five documents. However, returning the best document available was sometimes insufficient to help the subjects finish their work. For two of the subjects, the relevant documents' titles were not appealing enough for subjects to select them and look at their contents. Two others read the best document, but they did not try following the directions. Presumably, they did not think the feature described was the one they were seeking.

In the remaining five cases, the subjects read the texts and started following the instructions. This involved selecting a menu item to bring up a dialog box. Then, the subjects either failed to figure out how to use the dialog correctly or decided to pursue a different solution. Frequently, the help text was not detailed enough in its description of the dialog box, its features, and its functionality. One subject was attempting to add a Spanish character to the text. SAGE directed her to the dialog that contained twelve different characters sets for adding international characters, small graphics, math symbols, etc. The help text failed to describe adequately the character sets and how to select them.

The desired character was not found in the default character set presented and the subject chose to continue with the next task.

(4) **No help requested.** In one instance, a subject stopped a task before asking for help. The subject was trying to create an indented, bulleted list. He asked for help creating bullets. SAGE gave him help for this subtask and he completed it. However, he could not solve the indenting problem after searching on his own and he did not choose to ask for SAGE's help.

It is interesting that of the 16 failed attempts, ten were interactions in which SAGE provided the best help available. These failures are more reasonably attributed to users' unwillingness to ask for help or deficiencies in the help text and interface.

8.2.3 Postexperiment Interview Following the experiment, subjects were interviewed. They were asked the same interview questions used in the first experiment, with the exceptions of questions five and six. These two questions asked about the users' expectations of the system. They did not provide any useful information during the first set of interviews and were eliminated for this interview.

Subjects' responses during the interview were remarkably similar to those received in the first experiment. The results are briefly reviewed here and notable additions or differences from the first set of interviews are described. Additional information from the interviews is included in Section 8.3 where a set of lessons learned during the experiment is given. As a reminder, the interview questions are repeated here.

(1) How did the help system's usefulness compare with other on-line help systems you have used?
(2) How did the help system's responses compare with those you would have expected from a person (e.g., the facilitator)?
(3) How much English do you think the help system understands?
(4) Did you feel you had to be careful about how you spoke or what you said? If so, in what ways?
(5) What bothered you most about the help system or what was most frustrating about it?
(6) What did you like best about the help system?
(7) Do you think such a help system is useful? Would you use it?

Summaries of the responses to these questions are given below.

(1) Thirteen subjects thought SAGE was better or much better than other help systems they had used. One subject did not comment because she had no prior experience with on-line help. Another's only comment was that the system needed to be faster. The final subject thought SAGE was similar to Microsoft's new AnswerWizard help system. Though SAGE was not as close to perfect as the system simulated in the first experiment, subjects thought it was a very useful help system.

(2) The primary difference expected when asking a person for help was the human's ability to make inferences about the source of trouble. The additional language capabilities of a person were also cited by three subjects as a strength. They had trouble articulating their question and they wanted to be able to describe their problem. One subject thought he would prefer the verbal responses given by a person because reading the help text took too long. In contrast, two subjects thought

they learned best by reading and preferred that interaction style. An advantage of interacting with SAGE rather than a person mentioned by three subjects was that the computer provided would not forget answers and it could easily present various related choices for the users to browse.

(3&4) Subjects' responses to these two questions about the system's command of English closely matched the responses given by subjects in the previous study. Most said that they tried to speak clearly and restrict their questions to use words likely to be found in keyword help systems. Five subjects said they were cautious at first, but then they began to ask questions like they would if asking a person. From the responses gathered in these questions, it did not appear that the less accurate answers given by SAGE affected the subjects' perception of its language abilities.

(5) As in the first interview, the most prevalent complaint of the system was its speed. Still, five of the nine subjects with this complaint noticed that they adapted to the system by using the time to continue searching or they anticipated a future need for help and asked well in advance. The most frustrating thing about the system for five subjects was when they had trouble formulating a question to ask. Three subjects with this problem doubted the system had very extensive language abilities and so they had trouble putting their information need into words the system could understand. One subject simply did not care for voice activation, particularly in help systems. He felt that each time he used the system he was announcing to anyone within earshot that he needed help with the application.

(6) Subjects' choice of their favorite feature varied widely. One enjoyed being able to get help without being interrupted from the task. Speaking their questions was preferred by four subjects. Six subjects liked the list of relevant document titles. They felt these gave them additional options to examine when they had a difficult time asking a question that adequately expressed their information need. One subject cited the novelty of the system as the most fun thing about it.

(7) Fourteen subjects said they thought they would use such a system. They thought it would be particularly useful for novel tasks. The subject who did not like asking for help aloud would not use the system. The final subject remained uncommitted. However, he thought the system would be more useful for someone who does not use computers as often as he does.

Overall, the responses given during the interview were positive in favor of SAGE. Substitution of SAGE for the human question-answerer in the initial WOZ study did not diminish subjects' enthusiasm for the system and the improvement it offers over traditional help facilities. Subjects' behavior and interactions with the system did not differ much when using SAGE compared with the more "intelligent" system.

8.3 Lessons Learned

Certainly, many if not all of the observations that follow have been made by others. The literature undoubtedly had an effect on the things noticed during the experiment and the interpretations of those observations. However, inclusion of this material is valuable as supporting evidence for the work of others and simply as

a repository of lessons learned about users and on-line help.

(1) On-line help cannot help those who do not ask.

This observation seems obvious, but this experiment gives some additional insight about why people may not ask for help. Clearly, there are limits to the help that can be obtained by a system such as SAGE, but some users never tested those limits. As (Rieman, 1994) found, some users may have failed to obtain help effectively with previous help systems and so now they simply resist using help systems. Also, some participants who did not succeed with a few questions appeared a little more hesitant to try again. However, this was not always the case and no direct correlations of this behavior were found in the data.

A small group of users had success the few times they used SAGE, but they often spent minutes at a time hunting through menus and browsing dialog boxes without turning to SAGE for help. There appeared to be two reasons for their reluctance. One group thought they knew where to find the feature they were looking for and become so engrossed in the search that they did not turn to the system for help. For example, features related to tables were usually found within the table menu. This consistency gave users a starting point for their search. Although their first few attempts to find a desired feature were unsuccessful, they felt they were on the right track and continued hunting. One subject described the behavior in response to a question about the system's usefulness as follows, "It was helpful, but ... I know how to create a general table and once I get there, I get so busy trying to figure it out ..." This subject suspected that a solution was close to being found and asking SAGe for help was not necessary.

The other users reluctant to use help are those that I call the independent, silent types. Their attitude may be that asking for help would be a source of embarrassment or be seen as a sign of weakness. One user commented that he probably would not use a spoken-language help system because he did not want to "broadcast to the world" that he needed help. A second characterized himself as someone who "won't ask [for] directions until we've been lost for an hour."

(2) Users do not always ask the right questions.

When users ask the wrong question for the task they want to complete, it is frequently caused by a mismatch between their vocabulary and the system's. Sometimes, they can recognize the difference and learn the system's vocabulary (see observation (6) below for an exception), but when they do not recognize the mismatch, they can be lead astray. One subject asked how to make a chart. WordPerfect uses the term chart for plots and bar graphs. The first document returned described how to create a chart from a table. This information helped the subject recognize that table was the name for the desired object. However, a different subject also asked how to create charts. This subject did not recognize the vocabulary mismatch and proceeded to follow the instructions. The chart had a display option to format it as a table, but the table border formatting options are not available for charts. The desired look could not be achieved because the subject had been led to use the wrong object.

Another common occurrence of vocabulary mismatch occurred when users asked about specific items and the system only described them in more general terms. For example, users wanted to create "foreign characters," "Spanish characters," or

even a "tilde over an 'n'." These are specific names for a particular character. However, the help text never addresses such a specific topic. Instead the correct answer talks about using "WordPerfect Characters" to create characters not found on the keyboard. Another example of this problem is when users asked about adding and modifying **titles** rather than using the more generic term **text**.

(3) On-line help does not answer all the right questions.

This observation is not new or surprising. Many others have commented on the limitations of on-line help as described in Chapter 2. When users have a basic understanding of how to complete a task, but lack some way to apply it in their particular context, they need help adapted to their situation. Receiving the same static text to a context specific question becomes very frustrating because it fails to provide new insight into a problem's solution.

One task required subjects to add page numbers to the document. This task was given after they have been working on page two. Subjects who asked about adding page numbers received a correct answer on that topic. However, following the prescribed actions adds page numbers to pages beginning with the current page, not from the beginning of the document. Some subjects noticed a page number was missing on the first page. They asked for help adding page numbers to the **first** page received the same response as before. SAGE's responses are limited to the available help text so it can do no better. However, the response does not fully answer the question because it is not adapted to the users' particular situation. In these instances, users expressed a desire to be able to describe their problem to someone who could infer the cause of the trouble and help them work through a solution.

(4) Users sometimes need information options.

As a corollary to observation (2), when users know they may not be asking the right question, they appreciate the availability of extra options for getting help. Users are aware when they are struggling to phrase their questions or articulate their information needs and recognize that they may not find exactly what they need with the first answer given. When this happens, they do not seem to mind searching a little for information that may be helpful.

Chapter 5 discussed the choice of a document cutoff level based on the expected number of document titles users were expected to examine. The expectation was that 20 was a generous number and users were only likely to look at five to ten documents. This assumption was both true and false. It was true that users only looked at the contents of five or fewer documents. However, the false assumption was that users would only want the top five documents available to them.

Seven of the 16 subjects stated that they appreciated the opportunity to scan the titles of a larger return set. They believed that the return list "found more relevant stuff" and "made allowances for me not understanding exactly what I wanted." Another thought the system "anticipated what else I might need." None of the subjects commented on whether the size of the list (20 documents) should be larger or if it was too large. Further research is necessary to determine an appropriate return set size.

(5) Help texts must be explicit and anticipate problems.

When users are focused on a particular task solution, often they are blind to other possibilities. They fail to infer possible solutions to their problems and need to be told explicitly. Two examples will help clarify this point.

The first example illustrates how users fail to use peripheral information in solving their problem. They need something to bring that information to their direct attention. In trying to modify the borders of a table, four subjects found their way to the table styles dialog box. This dialog box contained the list of all predefined table styles and a small picture to show what they looked like. The 40 predefined styles have names such as, "Single No Border," "Double Border Header," and "Ledger Open Header." In fact, almost 20% of them contain the word "border" in their name. The assigned task could not be completed with a predefined style and it was surprising that none of the subjects recognized "border" as a relevant term to include subsequent questions to SAGE.

The second example comes from the task of creating an indented, bulleted list. The document on indenting text mentions that text is moved one tab stop to the right. The default configuration of WordPerfect has only a single tab stop defined. Indenting a bullet causes it to move to the first tab stop and the text following it also moves to the next tab stop available, which happens to be on the line below. This appearance is probably not what the user had in mind. Readers of the help text do not infer that the document's tab stops must also be set properly to use indenting effectively. An explicit mention of this fact would help draw the users' attention to this detail.

A fine line exists between too much help text and not enough. If too much text is given, users complain about being required to read so much to find just the information they want. When not enough is provided, users cannot obtain the help they need to complete some tasks. In choosing where to draw this line, help text authors should strongly consider providing explicit information about preconditions and how to resolve certain problems. All the information does not need to appear on a single page, but it should be obvious how to find it easily.

(6) A few early successes can create very trusting users.
The first few experimental tasks were designed to be fairly easy and later tasks were expected to be more difficult. Novice application users who had success completing early tasks using the first returned help text seemed hesitant to examine other texts while completing later tasks. They appeared to have such confidence in the help system's abilities that they trusted all its answers to be accurate.

Two subjects who received accurate help with earlier tasks became stuck on the sixth task. The sixth task involved creating a bulleted list indented from the margin. Both subjects asked similar questions about how to indent bullets. The top-ranked document was about creating bullets and the second document described how to indent. However, both subjects focused their attention entirely on the text about bullets. They read the text, followed the directions, and added bullets. The indentation was incorrect so they deleted their work, reread the text more slowly, and repeated the same actions with the same results. It never seemed to occur to either of them that they had exhausted the possibilities afforded by the top-ranked document and that they needed to look elsewhere.

A different subject was trying to modify the borders in a table. The question, **"I need to make one line change in the table."** was asked. The first text returned by SAGE was about modifying a table of authorities, WordPerfect's name for a bibliography or reference section. The subject struggled to figure out how this information could help with her problem. The correct document, titled **Table Lines/Fill**, was listed in position four, well within view on the display. However, the subject gave up on the task before examining any additional documents.

(7) Users want help systems to be unintrusive.

This observation is made based on this experiment and the initial data collection experiment. Users enjoyed being able to ask for help verbally because it permitted them to continue to focus on their tasks. Two subjects anticipated future information needs and asked questions while continuing to modify the document.

Similarly, subjects did not want the system's responses to disrupt them either. Particularly when responses are slower than real-time, they must not interrupt the user's work. Subjects praised the system for not occluding their work with the help text. This observation poses difficult challenges for the user interface designer. Computer display real estate is a precious commodity and must be used wisely. When help systems get in the way of task completion they become intrusive.

(8) Users adapt quickly to the system's performance.

A major concern with SAGE is the length of time it takes to get a response. Of course, we would all like instant answers to our problems, but SAGE fails to satisfy this wish. The response time experienced with SAGE is usually between 15 and 20 seconds. Part of the delay is due to network communication and a nonoptimized implementation of SAGE. However, much of it is the result of the extended processing SAGE does.

Users quickly modified their problem-solving behaviors to adapt to this deficiency of SAGE. They tried to anticipate problems they might need help with, asked for help early, and continued to work while SAGE searched for an answer. Part of this behavior was made possible by the nonintrusive nature of the help system. Users commented that they would prefer the system to be faster, but they did not mind waiting if they could continue working and provided the answers given were worth the wait. One user specifically said that waiting for SAGE to respond was still faster than searching through the keyword help system. Whether this attitude would persist after extended use of the system is another question for future exploration.

(9) A little lightheartedness can relieve users' frustrations.

I admit that this observation may not have much practical use in a real work situation, but the incident was too enjoyable not to include somewhere. A particular user who vocalized many of her thoughts and goals during the experiment was becoming frustrated while trying to indent a bulleted list. After nearly eleven minutes of unsuccessful attempts to complete the task, she finally said to herself, "All right computer, you've got to help me out here." Since subjects had been instructed to address the computer with the word **computer** before asking a question, the wizard dutifully entered "you've got to help me out here" as a question. What answer did SAGE give? The first document returned contained instructions for uninstalling WordPerfect from the computer! We both had a good laugh at this and the subject's

mood lightened considerably.

8.4 Conclusions

The results of this experiment are encouraging. SAGE answered questions correctly at a high rate and users were able to accomplish most of their work using it. However, a limitation of this research is that the application's full range of features could not be tested. The tasks were restricted to those that could easily be conveyed using a picture and those that did not take an excessive time to complete. These included creating, deleting, and changing the appearance of some objects in the document. More complex tasks might include constructing macros to execute common tasks or using formulas in a table to update its contents automatically. However, the user would have a difficult time interpreting what the true task was intended to be based on a picture alone. The picture can only display the results of the underlying macro or formula.

Based on this experiment and with these limitations in mind, SAGE succeeds at the task it was designed to do. It gives users an effective means of finding help texts relevant to their task-specific questions. Furthermore, the interaction style it promotes is well received by most users (of course, this assumes that the speech processing of the system is adequate). Because its goal is to be application independent, it cannot make inferences about users goals within the application. Consequently, the help it provides is not suitable for highly context dependent problems. Also, the help it provides is limited to the quality of the texts it is given to index. In spite of these limitations, SAGE is well received by most of its users and it is heralded as a more useful help system than traditional keyword systems.

CHAPTER 9

SUMMARY AND FINAL DISCUSSION

This research was motivated by personal and observed difficulties encountered when using traditional help systems. Typically, they are user unfriendly. They require users to think in terms of keywords and finding answers with them is often clumsy and time-consuming. SAGE was developed to address these problems.

9.1 Research Review

The research presented here is based on an empirical approach to intelligent system development. Data from real users was collected by giving them real tasks. A simulated intelligent help system listened and responded to their spoken, task-specific questions. These questions were recorded and used to guide the development of SAGE, an automated system to replace the simulated one. A final test of SAGE, again using real users with real tasks, verified that the implemented techniques worked as well with new questions as they did with the initial set.

The development of SAGE began with an information retrieval engine. This engine was built around Latent Semantic Indexing (LSI), a variant of the vector space model of information retrieval. An index of the WordPerfect help collection was built and a baseline performance standard was established. Performance of this index was measured as the number of questions from the test corpus answered correctly with the top one, three, five, ten, and twenty documents returned by LSI. Additional information retrieval techniques were tested with the system and performance improved only by including a morphological stemmer.

The architecture of SAGE was designed to permit multiple, lightweight components to be added for manipulating documents and queries. Preprocessing components alter the document collection in some way before it is indexed by LSI. Postprocessing components manipulate the list of ranked documents produced by LSI and then pass the (possibly) modified list to the next component or to the user. Three components were written to test the integration of weak natural language processing and machine learning techniques with LSI.

Two components tested were postprocessing components. The first one tags words in the questions with their parts of speech. The tagged words are matched against a set of patterns to eliminate those unrelated to the expressed information need. The tagged words are also used to identify nouns, verbs, and their modifiers. From these, a new query vector is created and the top documents are reranked by their similarity to this vector. The second postprocessing component filters out documents of the wrong type based on the expectations dictated by a question's content.

The final component is a preprocessor that teaches LSI new mappings of questions to documents. A sensitivity analysis helped determine that questions

should be collected from eight to twelve sample users. The analysis also showed that this learning method does not significantly inhibit SAGE's ability to answer novel questions. Instead, it has a small positive effect at higher document cutoff levels.

Throughout the development of SAGE, results for each technique were presented as an incremental improvement over prior, successful techniques. Before the results and contributions of this research are reviewed, this section concludes with a graph (see Figure 9.1) showing the total question-answering performance of SAGE compared with a basic vector space model. The vector model shown here uses the traditional $tf{\cdot}idf$ term weighting formulas. The number of questions answered by

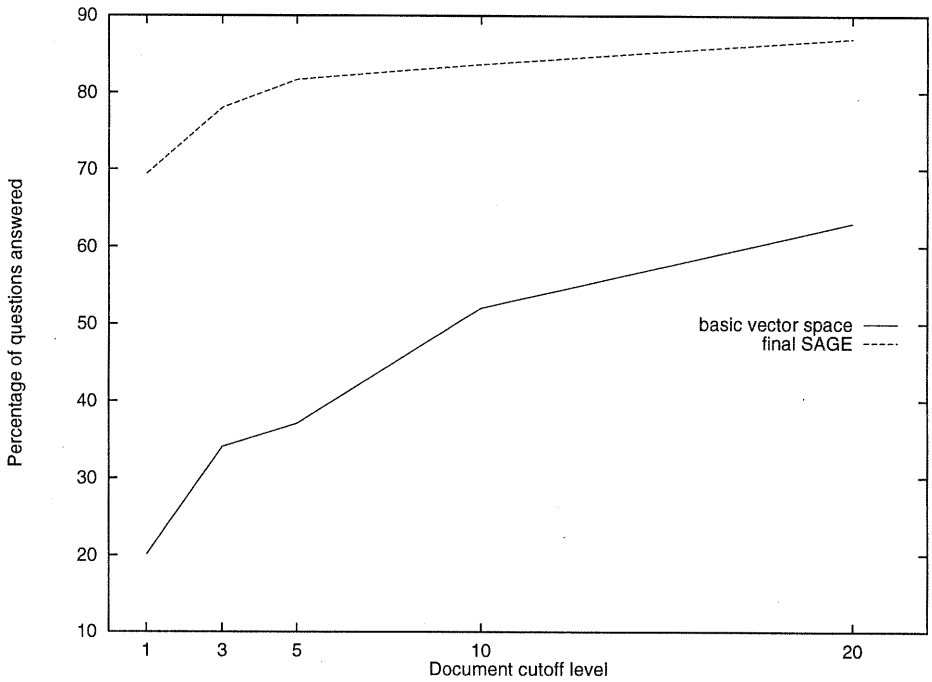


Figure 9.1. Total performance of SAGE compared with a basic vector space model.

SAGE compared with the vector space model is nearly 50 percentage points larger if only the first document is taken into account. Of the 262 questions in the test corpus, the vector model answers only 52, whereas SAGE answers 181 of them. When the top five ranked documents are searched for answers, 97 questions are answered by the vector space model, while SAGE answers 214.

9.2 Summary of Results

This research effort began with two primary questions:

- **How will users interact with an intelligent help system?**
- **How can submitted questions be answered intelligently?**

The two subject experiments addressed the first question. Subjects assumed the system was not very intelligent, but they still preferred it to traditional

help systems almost unanimously. Their interaction was intentionally limited to accommodate the perceived or expected limitations of the system. Most questions were short, grammatical sentences or phrases. The linguistic devices employed in their speech were also very limited. A significant result from the study was the indication that users are willing to avoid engaging the system in dialogue. Their questions are each completely self-contained. This result frees system designers from incorporating discourse management facilities in intelligent help systems.

Another important result from the second user study was the observation that subjects' behavior changed very little from the first user study. The first set of users had access to a near-perfect help system, while the second set used the SAGE system. Although SAGE was not as accurate as the wizard in the first experiment, the interaction with the system was not noticeably different. Furthermore, postexperiment interviews revealed that subjects had similar likes, dislikes, and impressions of the two systems. They were satisfied with the system's performance even though it was not perfect.

The second question explored by this research was investigated through the development of the question-answering system, SAGE. Several contributions came from this effort. Chapter 6 explored the application of information retrieval techniques to the problem of question answering. The observation from these experiments was that question answering is different from traditional information retrieval. Because the goal of each differs somewhat from the other, most techniques developed by the information retrieval community were not effective at increasing SAGE's ability to return correct answers.

Another interesting contribution is the architecture of SAGE. The system design treats a help collection and user question as two streams of data that are manipulated and merged to produce answers. It is built around a simple pipeline model of data flow in which lightweight components may be plugged into the flow to manipulate the contents and properties (i.e., term weights, document ranks, etc.) of different data elements. This architecture permits new techniques and ideas to be tested easily without disrupting the rest of the system.

The three components added in Chapter 7 validated the theory that combining weak AI techniques with an IR engine would improve the system's ability to answer questions. The components tested were not intended to be an exhaustive exploration of AI techniques. Instead, the three techniques selected were chosen because they were quite different in nature and each made a contribution to improved system performance. The part of speech processing component manipulates the questions as part of an answer reranking strategy. Document filtering focuses more on the contents of the documents while reranking them. Finally, the learning technique is a sample preprocessing technique that modifies the documents' contents before they are indexed.

The work done with each of these techniques also provides contributions to the field. The idea of producing a new query vector and executing another document look up is not new; however, I believe the idea presented in this work is novel in two ways. First, the approach presented here constructs a new query based entirely on the original in which the only goal is to identify nouns and verbs and equalize

their importance in the query. Second, the new query is used to rerank the top documents, rather than performing a completely new retrieval as is typically done in the information retrieval community. The document filtering work proposed a rule-based method for identifying document types. Although these rules are currently hand-coded, eventually machine learning techniques could be applied to acquire them automatically. Finally, the section on learning from users was unique because of the sensitivity analysis performed on the results. Furnas' adaptive indexing (Furnas, 1985) showed that learning new mappings can help with later searches. The work presented here extends this result with an analysis of the effect on performance using sample questions from different numbers of users. Furthermore, it shows that learning on some tasks does not negatively affect the performance when users ask about new tasks.

9.3 Future Work

The work presented here is a first attempt at answering task-specific questions for high-functionality applications. The techniques employed in the development of SAGE raise questions for further research. Three of these are suggested here.

One goal of this work was to develop application independent techniques that could easily be used in other domains. One area in which this goal was not fully realized was the document classification component. As mentioned earlier, machine learning techniques should be used to generalize the technique so it can be applied to other collections and domains.

In the part of speech tagging technique, the term weighting scheme seeks to equalize the objects and the actions performed on them. This practice should be validated in additional domains. Furthermore, a more sophisticated approach would be interesting to develop and test. For example, nominalized verbs probably ought to be treated as verbs rather than nouns. The request, **"Tell me about table justification."**, would receive a more accurate response if the nominalized verb **justification** were treated as the verb **justify**. Additionally, this technique could be expanded to evaluate the relationships of terms in the documents too. Many top-ranked documents that are not relevant to a question contain all the right terms, but the terms do not modify each other in the way necessary to answer the question. SAGE currently identifies simple relationships among terms in the question. However, the documents are still treated as bags of words. Therefore, highly ranked documents could be checked to verify that the same object-action relationships hold within them and then they could be reranked accordingly.

Finally, a frustrating thing about SAGE is the very static nature of its answers. Subjects in the second experiment who failed to complete a task often did so because they missed a key part of the instructions. The claim from this research is that SAGE can function adequately without discourse management. However, by remembering the previous few questions, SAGE could be equipped to recognize when a subject was asking about the same topic repeatedly. Then, further research could be done to give SAGE the ability to identify and focus the users' attention on various preconditions and steps required to complete the task.

BIBLIOGRAPHY

Aaronson, Amy and John M. Carroll. 1987a. The answer is in the question: A protocol study of intelligent help. **Behavior and Information Technology**, 6(4):393–402.

Aaronson, Amy and John M. Carroll. 1987b. Intelligent help in a one-shot dialog: A protocol study. In **Proceedings of the CHI + GI '87 Conference**, pages 163–168. Association for Computing Machinery.

Allen, James. 1995. **Natural Language Understanding**. The Benjamin/Cummings Publishing Company, Redwood City, CA.

Allen, James F. 1991. Discourse structure in the TRAINS project. In **Proceedings of the Speech and Natural Language Workshop**, pages 325–330, CA, February. DARPA, Morgan Kaufmann Publishers.

Allen, James F. and Lenhart K. Schubert. 1991. The TRAINS project. Technical Report TRAINS Technical Note 91-1, Computer Science Department, University of Rochester.

Barr, Avron and Edward A. Feigenbaum. 1982. **The Handbook of Artificial Intelligence**, volume 1. HeurisTech, Stanford, California.

Bates, Madeleine, Robert Bobrow, Pascale Fung, Robert Ingria, Francis Kubala, John Makhoul, Long Nguyen, Richard Schwartz, and David Stallard. 1993. The bbn/harc spoken language understanding system. In **Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)**, volume 2, pages 111–114. IEEE, April.

Berry, Michael W. and Susan T. Dumais. 1994. Using linear algebra for intelligent information retrieval. Technical Report CS-94-270, Computer Science Department, University of Tennessee, Knoxville, December.

Borenstein, Nathaniel S. 1985. **The Design and Evaluation of On-Line Help Systems**. Ph.D. thesis, Computer Science Department, Carnegie Mellon University. Technical Report CMU-CS-85-151.

Brill, Eric. 1993. **A Corpus-Based Approach to Language Learning**. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.

Broglio, John, James P. Callan, W. Bruce Croft, and Daniel W. Nachbar. 1995. Document retrieval and routing using the INQUERY system. In Donna Harman, editor, **Proceedings of the Third Text REtrieval Conference (TREC-3)**, Gaithersburg, MD. National Institute of Standards and Technology Special Publication.

Brunner, Hans, Greg Whittemore, Kathleen Ferrara, and Jiamiene Hsu. 1992. An assessment of written/interactive dialogue for information retrieval applications. **Human-Computer Interaction**, 7(2):197–249.

Buckley, Chris. 1985. Implementation of the SMART information retrieval system. Technical report, Cornell University.

Buckley, Chris, Gerard Salton, James Allen, and Amit Singhal. 1995. Automatic query expansion using SMART: TREC 3. In Donna Harman, editor, **Proceedings of the Third Text REtrieval Conference (TREC-3)**, Gaithersburg, MD. National Institute of Standards and Technology Special Publication.

Burgin, Robert. 1994. Variations in relevance judgments and the evaluation of retrieval performance. **Information Processing & Management**, 28(5):619–627.

Carberrry, Sandra. 1989. A pragmatics-based approach to ellipsis resolutions. **Computational Linguistics**, 15:75–96.

Carroll, John M. and Amy P. Aaronson. 1988. Learning by doing with simulated intelligent help. **Communications of the ACM**, 31(9):1064–1079, September.

Chapanis, Alphonse. 1981. Interactive human communication: Some lessons learned from laboratory experiments. In B. Shackel, editor, **Man-Computer Interaction: Human Factors Aspects of Computers & People**. Sijthoff & Noordhoff, Rockville, MD, pages 65–114.

Charniak, Eugene. 1993. **Statistical Language Learning**. The MIT Press, Cambridge, MA.

Chen, Hsinchun. 1994. Collaborative systems: Solving the vocabulary problem. **IEEE Computer**, 27(5):58–66, May.

Chu-Carroll, Jennifer and Sandra Carberry. 1994. A plan-based model for response generation in collaborative task-oriented dialogues. In **Proceedings of the Twelfth Conference on Artificial Intelligence**, pages 799–805. American Association for Artificial Intelligence.

Cohen, Michael, Ze'ev Rivlin, and Harry Bratt. 1995. Speech recognition in the ATIS domain using multiple knowledge sources. In **Proceedings of the Spoken**

Language Systems Technology Workshop, pages 257–260. ARPA, Morgan Kaufmann, January.

Cohen, Paul R. 1995. **Empirical Methods for Artificial Intelligence**. The MIT Press, Cambridge, MA.

Cohen, Paul R. and Sharon L. Oviatt. 1995. The role of voice input for human-machine communication. In **Proceedings of the National Academy of Sciences**, volume 92.

Cooper, William S., Aitao Chen, and Frederic C. Gey. 1995. Experiments in the probabilistic retrieval of full text documents. In Donna Harman, editor, **Proceedings of the Third Text REtrieval Conference (TREC-3)**, Gaithersburg, MD. National Institute of Standards and Technology Special Publication.

Dahlback, N., A. Jonsson, and L. Ahrenberg. 1993. Wizard of Oz studies – Why and how. In **Proceedings of the ACM International Workshop on Intelligent Interfaces (cited by MaulsbyEtal93) (or??) Third Conference on Applied Natural Language Processing, Trento, Italy 1992 (cited by Salber93a)**. Association for Computing Machinery.

Deerwester, Scott, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard A. Harshman. 1990. Indexing by Latent Semantic Analysis. **Journal of the American Society for Information Science**, 41(6):391–407, September.

Dumais, Susan T. 1992. Enhancing performance in Latent Semantic Indexing (LSI) retrieval. Technical report, Bellcore, September.

Dumais, Susan T. 1995. Latent semantic indexing (LSI): TREC-3 report. In Donna Harman, editor, **Proceedings of the Third Text REtrieval Conference (TREC-3)**, Gaithersburg, MD. National Institute of Standards and Technology Special Publication.

Dumais, Susan T., George W. Furnas, Thomas K. Landauer, Scott Deerwester, and Richard Harshman. 1988. Using Latent Semantic Analysis to improve access to textual information. In **Human Factors in Computing Systems, CHI'88 Conference Proceedings (Washington, D.C.)**, pages 281–285, New York, May. ACM.

Eckart, Carl and Gale Young. 1939. A principle axis transformation for non-hermitian matrices. **American Mathematical Society Bulletin**, 45:118–121.

Fach, Peter W., Maria Bannert, and Klaus Kunkel. 1993. From conflict to dialogue: On attack and defense in on-line assistance. In Thomas Grechenig and Manfred Tscheligi, editors, **Human Computer Interaction**. Springer-Verlag, September, pages 103–113.

Finin, Timothy W., Aravind K. Joshi, and Bonnie Lynn Webber. 1986. Natural language interactions with artificial experts. **Proceedings of the IEEE**, 74(7):921–938.

Fischer, Gerhard and H. Nieper-Lemke. 1989. Helgon: Extending the retrieval by reformulation paradigm. In **Human Factors in Computing Systems, CHI'89 Conference Proceedings (Austin, TX)**, pages 357–362, New York, May. Association for Computing Machinery.

Foltz, Peter W. 1995. Improving human-proceedings interaction: Indexing the CHI index. In **Human Factors in Computing Systems: CHI'95 Conference Companion**, pages 101–102. Associations for Computing Machinery (ACM), May.

Ford, Nigel. 1991. Knowledge-based information retrieval. **Journal of the American Society for Information Science**, 42(1), January.

Fox, Barbara A. 1988. Robust learning environments: The issues of canned text. Technical report, University of Colorado at Boulder.

Fox, Barbara A. 1993. **The Human Tutorial Dialogue Project**. Computers, Cognition, and Work Series. Lawrence Erlbaum Associates, Inc., New Jersey.

Frakes, William B. and Ricardo Baeza-Yates, editors. 1992. **Information Retrieval - Data Structures and Algorithms**. Prentice Hall, New Jersey.

Franzke, Marita and John Rieman. 1993. Natural training wheels: Learning and transfer between two versions of a computer application. In Thomas Grechenig and Manfred Tscheligi, editors, **Human Computer Interaction**. Springer-Verlag, pages 317–328.

Fraser, Norman M. and G. Nigel Gilbert. 1991. Simulating speech systems. **Computer Speech and Language**, 5:81–99.

Fuhr, Norbert. 1992. Probabilistic models in information retrieval. **The Computer Journal**, 35(3):243–255.

Fukumoto, Fumiyo, Yoshimi Suzuki, and Jun'ichi Fukumoto. 1997. An automatic extraction of key paragraphs based on context dependency. In **Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP'97)**, pages 291–298, Washington, DC.

Furnas, George W. 1985. Experience with an adaptive indexing scheme. In **CHI'82 Proceedings**, pages 131–135. Association for Computing Machinery, April.

Furnas, George W., Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. 1984. Statistical semantics: Analysis of the potential performance of key-word information systems. In **Human Factors in Computer Systems**. Ablex Publishing Company, New Jersey, chapter 8, pages 187–242. (Also in **Bell System Technical Journal**, 62:1753–1806.).

Furnas, George W., Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. 1987. The vocabulary problem in human-system communication. **Communications of the ACM**, 30(11):964–971, November.

Gauch, Susan. 1992. Intelligent information retrieval: An introduction. **Journal of the American Society for Information Science**, 43(2), March.

Glass, James, David Goddeau, Lee Hetherington, Michael McCandless, Christine Pao, Michael Phillips, Joseph Polifroni, Stephanie Seneff, and Victor Zue. 1995. The MIT ATIS system: December 1994 progress report. In **Proceedings of the Spoken Language Systems Technology Workshop**, pages 252–256. ARPA, Morgan Kaufmann, January.

Gould, J. D., J. Conti, and T. Hovanyecz. 1982. Composing letters with a simulated listening typewriter. In **CHI'82 Conference Proceedings**, pages 367–370. Association for Computing Machinery.

Graesser, Arthur C., Natalie Person, and John Huber. 1992. Mechanisms that generate questions. In Thomas W. Lauer, Eileen Peacock, and Arthur C. Graesser, editors, **Questions and Information Systems**. Lawrence Erlbaum Associates, chapter 9, pages 167–187.

Grefenstette, Gregory. 1994. **Explorations in Automatic Thesaurus Discovery**. Kluwer Academic Publishers, United States.

Guindon, Raymonde. 1988. How to interface to advisory systems? Users request help with a very simple language. In **Human Factors in Computing Systems, CHI'88 Conference Proceedings (Washington, D.C.)**, pages 191–196, New York, May. Association for Computing Machinery.

Guindon, Raymonde. 1991. Users request help from advisory systems with simple language: Effects of real-time constraints and limited shared context. **Human-Computer Interaction**, 6:47–75.

Haas, Stephanie and Robert M. Losee Jr. 1994. Looking in text windows: Their size and composition. **Information Processing & Management**, 30(5):619–629.

Harman, Donna. 1986. An experimental study of factors important in document ranking. In **Proceedings of the 1986 ACM Conference on Research and Development in Information Retrieval**, Pisa, Italy.

Harman, Donna. 1991. How effective is suffixing? **Journal of the American Society for Information Science**, 42(1):7–15, January.

Harman, Donna. 1995a. Overview of the third Text REtrieval Conference (TREC-3). In Donna Harman, editor, **Proceedings of the Third Text REtrieval Conference (TREC-3)**, Gaithersburg, MD. National Institute of Standards and Technology Special Publication.

Harman, Donna, editor. 1995b. **The Third Text REtrieval Conference (TREC-3)**, Washington, DC. Government Printing Office (GPO). (NIST SP 500-225).

Harman, Donna, editor. 1996. **The Fourth Text REtrieval Conference (TREC-4)**, Springfield, VA. National Technical Information Services (NTIS). (NIST SP 500-236).

Hauptmann, Alexander G. and Alexander I. Rudnicky. 1988. Talking to computers: an empirical investigation. **International Journal of Man-Machine Studies**, 28:583–604.

Hill, William C. 1989. How some advice fails. In **Human Factors in Computing Systems, CHI'89 Conference Proceedings (Austin, TX)**, pages 85–90, New York, May. Association for Computing Machinery.

Hill, William C. and James R. Miller. 1988. Justified advice: A semi-naturalistic study of advisory strategies. In **Human Factors in Computing Systems, CHI'88 Conference Proceedings (Washington, D.C.)**, pages 185–190, New York, May. Association for Computing Machinery.

Hirshman, Lynette. 1994. The roles of language processing in a spoken language interface. In David B. Roe and Jay G. Wilpon, editors, **Voice Communications Between Humans and Machines**. National Academy of Sciences, Washington, D.C., pages 217–237.

Issar, Sunil and Wayne Ward. 1994. Unanswerable queries in a spontaneous speech task. In **Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)**, volume 1, pages 341–345. IEEE, April.

Jing, Y. and W. Bruce Croft. 1994. An association thesaurus for information retrieval. In **RIAO Conference Proceedings**, New York, October.

Jones, Michael P. and James H. Martin. 1997. Contextual spelling correction using latent semantic analysis. In **Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP'97)**, pages 166–173, Washington, DC.

Jones, William P. and George W. Furnas. 1987. Pictures of relevance: A geometric analysis of similarity measures. **Journal of the American Society for Information Science**, 38:420–442.

Jurafsky, Daniel, Chuck Wooters, Gary Tajchman, Jonathan Segal, Andreas Stolke, and Nelson Morgan. 1994. Integrating experimental models of syntax, phonology, and accent/dialect in a speech recognizer. In **Proceedings of the AAAI-94 Workshop on Integration of Natural Language and Speech Processing**, pages 107–115. American Association for Artificial Intelligence.

Keen, E. Michael. 1992. Presenting results of experimental retrieval comparisons. **Information Processing & Management**, 28(4):491–502.

Keene, Carol A. 1987. Retrieval of unstructured natural language text. In **Proceedings of the 1987 Rocky Mountain Conference on Artificial Intelligence**, pages 73–83, Boulder, Colorado, June. University of Colorado.

Keene, Carol A. 1991. The Chaucer on-line document retrieval system. Technical Report TR-2, Department of Computer Science, University of Colorado at Denver.

Kelly, M. J. and A. Chapanis. 1977. Limited vocabulary natural language dialogue. **International Journal of Man-Machine Studies**, 9(4):479–501.

Korycinski, C. and Alan F. Newell. 1990. Natural-language processing and automatic indexing. **The Indexer**, 17(1):21–29.

Kowtko, Jacqueline C. and Patti J. Price. 1989. Data collection and analysis in the air travel planning domain. In **DARPA Speech and Natural Language Workshop**, pages 119–125, Los Altos, CA, October. DARPA, Morgan Kaufmann Publishers.

Kupiec, Julian. 1993. MURAX: A robust linguistic approach for question answering using an on-line encyclopedia. In **Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval**, pages 181–190, Pittsburgh, PA. Association for Computing Machinery.

Kwok, K.L., L. Grunfield, and D. D. Lewis. 1995. TREC-3 ad-hoc, routing retrieval and thresholding experiments using PIRCS. In Donna Harman, editor, **Proceedings of the Third Text REtrieval Conference (TREC-3)**, Gaithersburg, MD. National Institute of Standards and Technology Special Publication.

Lambert, Lynn and Sandra Carberry. 1991. A tripartite plan-based model of dialogue. In **Proceedings of the 29th Annual Meeting for the Association for Computational Linguistics**, pages 47–54. Association for Computational Linguistics.

Landauer, Thomas, Dennis Egan, Joel Remde, Michael Lesk, Carol Lochbaum, and Daniel Ketchum. 1993. Enhancing the usability of text through computer delivery and formative evaluation: The SuperBook project. In **Hypertext: A Psychological Perspective**. Ellis Horwood, New York, chapter 5.

Landauer, Thomas K. 1988. Research methods in human-computer interaction. In Martin Helander, editor, **Handbook of Human-Computer Interaction**. Elsevier Science Publishing Company Inc., New York, pages 905–928.

Landauer, Thomas K. and Susan T. Dumais. 1997. A solution to Plato's problem: The Latent Semantic Analysis theory of acquisition, induction, and representation of knowledge. **Psychological Review**, 104:211–240.

Lang, K. L., A. C. Graesser, and D. D. Hemphill. 1990. The role of questioning in knowledge engineering and the interface of expert systems. **Poetics**, 19:143–166.

Lehnert, Wendy G. 1978. **The Process of Question Answering: A Computer Simulation of Cognition**. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Lewis, David D. 1992. **Representation and Learning in Information Retrieval**. Ph.D. thesis, Department of Computer and Information Science, University of Massachusetts.

Lin, Chin-Yew and Eduard Hovy. 1997. Identifying topics by position. In **Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP'97)**, pages 283–290, Washington, DC.

Lochbaum, Karen E. and Lynn A. Streeter. 1989. Comparing and combining the effectiveness of Latent Semantic Indexing and the ordinary vector space model for information retrieval. **Information Processing & Management**, 25(6):665–676.

Lovins, J. B. 1968. Development of a stemming algorithm. **Mechanical Translation and Computational Linguistics**, 11:22–31.

Lu, X. Allan and Robert B. Keefer. 1995. Query expansion/reduction and its impact on retrieval effectiveness. In Donna Harman, editor, **Proceedings of the Third Text REtrieval Conference (TREC-3)**, Gaithersburg, MD. National Institute of Standards and Technology Special Publication.

Mack, Robert L., Clayton H. Lewis, and John M. Carroll. 1983. Learning to use word processors: Problems and perspectives. **ACM Transactions on Office Information Systems**, 1(3):254–271, July.

Magerman, David M. 1994. **Natural Language Parsing as Statistical Pattern Recognition**. Ph.D. thesis, Department of Computer Science, Stanford University.

Mane, Amir, Susan Boyce, Demetrios Karis, and Nicole Yankelovich. 1996. Designing the user interface for speech recognition applications. **SIGCHI Bulletin**, 28(4):29–34, October.

Mann, W. C. and S. A. Thompson. 1987. Rhetorical structure theory: A theory of text organization. In **Discourse Structure**. Ablex, Norwood, New Jersey.

Martin, Gale L. 1989. The utility of speech input in user-computer interfaces. **International Journal of Man-Machine Studies**, 30:355–375.

Mauldin, Michael L. 1991. **Conceptual Information Retrieval: A Case Study in Adaptive Partial Parsing**. Kluwer Academic Publishers, New York.

Maulsby, David, Saul Greenberg, and Richard Mander. 1993. Prototyping an intelligent agent through Wizard of Oz. In **INTERCHI'93 Conference Proceedings**, pages 277–284. Association for Computing Machinery.

McKevitt, Paul. 1990. Data acquisition for natural language interfaces. Technical Report MCCS-90-178, Computing Research Laboratory, New Mexico State University.

McKevitt, Paul and William C. Ogden. 1990. OZWIZ II: Wizard-of-Oz dialogues in the computer operating systems domain. Technical Report MCCS-90-181, Computing Research Laboratory, New Mexico State University.

Meisel, William S. 1993. Talk to your computer. **Byte**, 18(11):113–120, October.

Miller, Scott, Madeleine Bates, Robert Bobrow, Robert Ingria, John Makhoul, and Richard Schwartz. 1995. Recent progress in hidden understanding models. In **Proceedings of the Spoken Language Systems Technology Workshop**, pages 276–280. ARPA, Morgan Kaufmann, January.

Moore, Johanna D. 1993. What makes human explanations effective? In **Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society**, pages 131–136. Cognitive Science Society, June.

Moore, Robert, Douglas Appelt, John Dowding, J. Mark Gawron, and Douglas Moran. 1995. Combining linguistic and statistical knowledge sources in natural-language processing for ATIS. In **Proceedings of the Spoken Language Systems Technology Workshop**, pages 261–264. ARPA, Morgan Kaufmann, January.

Oddy, Robert N., Elizabeth Duross Liddy, Bhaskaran Balakrishnan, Ann Bishop Joseph Elewononi, and Eileen Martin. 1992. Towards the use of situational information in information retrieval. **The Journal of Documentation**, 48(2):123–171, June.

Oviatt, Sharon L. 1995. Predicting spoken disfluencies during human-computer interaction. **Computer Speech and Language**, 9:19–35.

Oviatt, Sharon L. 1996. User-centered modeling for spoken language and multimodal interfaces. **IEEE Multimedia**, 3(4):26–35, winter.

Oviatt, Sharon L. and Robert VanGent. 1996. Error resolution during multimodal human-computer interaction. In **Proceedings of the 1996 International Conference on Spoken Language Processing**, volume 1, pages 204–207, Philadelphia.

Pilkington, Rachel M. 1992a. **Intelligent Help: Communicating with Knowledge-Based Systems**. Oxford University Press, New York, New York.

Pilkington, Rachel M. 1992b. Question-answering for intelligent on-line help: The process for intelligent responding. **Cognitive Science**, 16:455–489.

Pollack, Martha E. 1985. Information sought and information given: An empirical study of user/expert dialogues. In **Proceedings of the CHI'85 Conference**, San Francisco, California.

Porter, M. F. 1980. An algorithm for suffix stripping. **Program**, 14(3):130–137, July.

Resnick, Phillip S. 1993. **Selection and Information: A Class-Based Approach to Lexical Relations**. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.

Rieman, John Franklin. 1994. **Learning Strategies and Exploratory Behavior of Interactive Computer Users**. Ph.D. thesis, Department of Computer Science, University of Colorado at Boulder. Technical Report CU-CS-723-94.

Riloff, Ellen. 1996. NLP class lecture notes. Available on the Internet at http://www.cs.utah.edu/~cs534/slides/lecture14–ir_nlp.ps.

Robertson, S. E. 1969a. The parametric description of retrieval tests, part I: The basic parameters. **Journal of Documentation**, 25(1):1–27, March.

Robertson, S. E. 1969b. The parametric description of retrieval tests, part II: Overall measures. **Journal of Documentation**, 25(2):93–107, June.

Robertson, S. E. and M. M. Hancock-Beaulieu. 1992. On the evaluation of ir systems. **Information Processing & Management**, 28(4):457–466, July.

Robertson, S. E. and K. Sparck Jones. 1976. Relevance weighting of search terms. **Journal of the American Society for Information Science**, 27:129–146, May-June.

Russell, Stuart and Peter Norvig. 1995. **Artificial Intelligence: A Modern Approach**. Prentice Hall, Englewood Cliffs, NJ.

Salber, Daniel and Joelle Coutaz. 1993. Applying the Wizard of Oz technique to the study of multimodal systems. In **East-West HCI'93 Conference**, August.

Salton, Gerard. 1971. **The SMART Retrieval System; Experiments in Automatic Document Processing**. Prentice Hall, Englewood Cliffs, NJ.

Salton, Gerard. 1989. **Automatic Text Processing—The Transformation, Analysis and Retrieval of Information by Computer**. Addison-Wesley Publishing Co., Reading, MA.

Salton, Gerard. 1992. The state of retrieval system evaluation. **Information Processing & Management**, 28(4):441–450, July.

Salton, Gerard and Chris Buckley. 1988. Term weighting approaches in automatic text retrieval. **Information Processing & Management**, 24(5):513–523.

Salton, Gerard and Chris Buckley. 1990. Improving retrieval performance by relevance feedback. **Journal of the American Society for Information Science**, 41(4):288–297, June.

Salton, Gerard and Michael J. McGill. 1983a. **Introduction to Modern Information Retrieval**. McGraw-Hill Book Co., New York.

Salton, Gerard and Michael J. McGill. 1983b. The SMART and SIRE experimental retrieval systems. In **Introduction to Modern Information Retrieval**. McGraw-Hill Book Company, United States, chapter 4.

Sanderson, Mark. 1994. Word sense disambiguation and information retrieval. In **Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval**, pages 142–151, Dublin, Ireland, July. Association for Computing Machinery.

Santhanam, Radhika and Susan Wiedenbeck. 1991. Modeling the intermittent user of word processing technology. **Journal of the American Society for Information Science**, 42(3):185–196, April.

Sap, Mohd Noor Md and D. R. McGregor. 1992. Natural language interfaces to databases: State of the art. Technical report, Department of Computer Science, University of Strathelyde, May.

Schank, Roger C. and Robert P. Abelson. 1977. **Scripts, Plans, Goals, and Understanding : An Inquiry into Human Knowledge Structures.** Lawrence Erlbaum Associates, Hillsdale, N.J.

Seneff, Stephanie. 1992. Tina: A natural-language system for spoken-language applications. **Computational Linguistics**, 18(1):61–86, March.

Smadja, Frank. 1993. Retrieving collocations from text: Xtract. **Computational Linguistics**, 19(1):143–177.

Smeaton, Alan F. 1992. Progress in the application of natural language processing to information retrieval tasks. **The Computer Journal**, 35(3):268–278.

Smeaton, Alan F., Ruairi O'Donnell, and Fergus Kelledy. 1995. Indexing structures derived from syntax in TREC-3: System description. In Donna Harman, editor, **Proceedings of the Third Text REtrieval Conference (TREC-3)**, Gaithersburg, MD. National Institute of Standards and Technology Special Publication.

Smithson, Steve. 1994. Information retrieval evaluation in practice: A case study approach. **Information Processing & Management**, 30(2):205–221.

Sparck Jones, Karen. 1971. **Automatic Keyword Classification for Information Retrieval**. Buttersworth, London.

Sparck Jones, Karen. 1995. Appendix C - comparison between TREC2 and TREC3. In **Proceedings of the Third Text REtrieval Conference (TREC-3)**. National Institute of Standards and Technology (NIST), NIST special publication.

Streeter, Lynn A. and Karen E. Lochbaum. 1988. An expert/expert-locating system based on automatic representation of semantic structure. In **Proceedings of the Fourth Conference on Artificial Intelligence Applications**, pages 345–350, San Diego, CA, March. IEEE.

Strzalkowski, Tomek. 1994. Document representation in natural language text retrieval. In **Proceedings of the Human Language Technology Workshop**, pages 364–369, New Jersey, March 8-11. ARPA, Morgan Kaufmann.

Strzalkowski, Tomek, Jose Perez Carballo, and Mihnea Marinescu. 1995. Natural language information retrieval: TREC-3 report. In Donna Harman, editor, **Proceedings of the Third Text REtrieval Conference (TREC-3)**, Gaithersburg, MD. National Institute of Standards and Technology Special Publication 500-225.

Strzalkowski, Tomek, Fang Lin, Jose Perez-Carballo, and Jin Wang. 1997. Building effective queries in natural language information retrieval. In **Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP'97)**, pages 299–306, Washington, DC.

Su, Louise T. 1992. Evaluation measures for interactive information retrieval. **Information Processing & Management**, 28(4):503–516.

Tague-Sutcliffe, Jean. 1992. The pragmatics of information retrieval experimentation, revisited. **Information Processing & Management**, 28(4):467–490, July.

Turtle, Howard and W. Bruce Croft. 1990. Inference networks for document retrieval. In Jean-Luc Vidick, editor, **Proceedings of the 13^{th} International Conference on Research and Development in Information Retrieval**, pages 1–24. ACM, September.

Turtle, Howard R. and W. Bruce Croft. 1992. A comparison of text retrieval models. **The Computer Journal**, 35(3):279–290.

van Rijsbergen, C. J. 1979. **Information Retrieval, Second Edition**. Butterworth, London.

van Rijsbergen, C. J. 1992. Probabilistic retrieval revisited. **The Computer Journal**, 35(3):291–298.

Voorhees, Ellen M. 1993. Using WordNet to disambiguate word senses for text retrieval. In **Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval**, pages 171–180, Pittsburgh, PA, June. Association for Computing Machinery.

Ward, Wayne and Sunil Issar. 1995. The CMU ATIS system. In **Proceedings of the Spoken Language Systems Technology Workshop**, pages 249–251. ARPA, Morgan Kaufmann, January.

Wilensky, Robert, Yigal Arens, and David Chin. 1984. Talking to UNIX in English: An overview of UC. **Communications of the ACM**, 27(6):574–593.

Wilensky, Robert, David N. Chin, Marc Luria, James Martin, James Mayfield, and Dekai Wu. 1988. The Berkeley UNIX Consultant project. **Computational Linguistics**, 14(4):35–84.

Woods, William A. 1970. Transition network grammars for natural language analysis. **Communications of the ACM**, 13:457–471.

Yetim, Fahri. 1993. User-adapted hypertext explanations. In Thomas Grechenig and Manfred Tscheligi, editors, **Human Computer Interaction**. Springer-Verlag, September, pages 91–102.

Zhai, Chengziang. 1997. Fast statistical parsing of noun phrases for document indexing. In **Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP'97)**, pages 312–319, Washington, DC.

Zue, Victor, James Glass, David Goodine, Hong Leung, Michael Phillips, Joseph Polifroni, and Stephanie Seneff. 1991. Integration of speech recognition and natural language processing in the MIT VOYAGER system. In **Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)**, pages 713–716. IEEE, May.

Zue, Victor, James Glass, David Goodine, Hong Leung, Michael Phillips, Joseph Polifroni, and Stephanie Seneff. 1990. The VOYAGER speech understanding system: Preliminary development and evaluation. In **Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)**, pages 73–76. IEEE, April.

Zue, Victor W. 1994. ARPA strategic computing initiative: Toward systems that understand spoken language. **IEEE Expert**, pages 51–59, February.

APPENDIX A

USER EXPERIMENT TASKS

This appendix shows the beginning and final state of the document used in the second user experiment (see Chapter 8). The start state of the document is also the same as the one used in the first user experiment (see Chapter 3). The final document shown here contains only minor variations from the final document of the first experiment.

The following two pages show the document given to subjects. The next two pages after those show the document as it would look after all tasks had been completed. The dotted lines represent the page borders and were not part of the document given to the subjects. The individual steps performed to reach the final state are described in Section 3.3.2.

This memo represents an effort to keep you informed about the news and events from around the company. This is an exciting time for the company since we are almost ready to release gadgets to the market. As a company, we've been working on this project for a long time and I would like to thank each of you for helping to achieve this goal.

Marketing

The marketing department has been making plans to roll out our new product line of gadgets. The resulting plan calls for an aggressive sales pitch to our existing widget customers. Our marketing literature will demonstrate how seamlessly gadgets integrate with widgets while boosting productivity. We'll also begin to focus some of our marketing efforts on larger corporate accounts. The increased flexibility and power provided by gadgets should allow our sales force to compete for the Fortune 1000 accounts.

The marketing department will be sending out a press release at the end of the month which announces our new product. We are running a full-page advertisement in The American Journal of Widget Users beginning next month. Additionally, Dave Benson will be attending two trade shows next month to demonstrate gadgets.

Sales

The past quarter was a successful one for our sales staff. They met their department goal for the month. We'll show a small profit for the third quarter in a row.

Engineering

The engineering and quality assurance staffs have been busy trying to get gadgets ready for release. I would like to thank them for the many late hours and a few weekends they have spent readying this product for release. Last week we brought in some of our existing customers to try gadgets and give us some feedback. Without exception, the customers couldn't stop talking about how wonderful gadgets are and how they can't wait to get their own. The marketing department will be including some of their comments in the forthcoming sales collateral.

The research staff is currently designing and prototyping ideas for our next product. We expect this effort to continue for the next 3 or 4 months. At the end of that time, the prototypes will be shown to the senior staff and marketing and sales managers for review. I'll share more information about their efforts after the first review.

Finance

Finance has been busy trying to keep up with the orders booked by the sales staff. Our

ability to collect our accounts receivable in a timely manner has allowed us to maintain a positive cash flow. However, Jerry has expressed some concern to me about our increasing expenditures. Please don't request unnecessary equipment or supplies so that we can keep our expenses under control.

As the following financial chart shows, sales during the quarter were strong, but increasing expenses have begun to cut into our expected profits.

ACME Widgets

This memo represents an effort to keep you informed about the news and events from around the company. This is an exciting time for the company since we are almost ready to release gadgets to the market. As a company, we've been working on this project for a long time and I would like to thank each of you for helping to achieve this goal.

Engineering

The engineering and quality assurance staffs have been busy trying to get gadgets ready for release. I would like to thank them for the many late hours and a few weekends they have spent readying this product for release. Last week we brought in some of our existing customers to try gadgets and give us some feedback. Without exception, the customers couldn't stop talking about how wonderful gadgets are and how they can't wait to get their own. The marketing department will be including some of their comments in the forthcoming sales collateral.

The research staff is currently designing and prototyping ideas for our next product. We expect this effort to continue for the next 3 or 4 months. At the end of that time, the prototypes will be shown to the senior staff and marketing and sales managers for review. I'll share more information about their efforts after the first review.

Engineering has a number of open positions. Please submit a résumé if you wish to apply for one of the following jobs:

- Senior Research Engineer
- Quality Assurance Engineer
- Engineering Project Coordinator

Marketing

The marketing department has been making plans to roll out our new product line of gadgets. The resulting plan[1] calls for an aggressive sales pitch to our existing widget customers. Our marketing literature will demonstrate how seamlessly gadgets integrate with

1. See Sarah Brown if you would like a copy of the marketing plan.

widgets while boosting productivity. We'll also begin to focus some of our marketing efforts on larger corporate accounts. The increased flexibility and power provided by gadgets should allow our sales force to compete for the Fortune 1000 accounts.

The marketing department will be sending out a press release at the end of the month which announces our new product. We are running a full-page advertisement in *The American Journal of Widget Users* beginning next month. Additionally, Dave Benson will be attending two trade shows next month to demonstrate gadgets.

Sales

The past quarter was a successful one for our sales staff. They met their department goal for the month. We'll show a small profit for the third quarter in a row. Congratulations to Ann Miller and Peter Nuñez for exceeding their personal sales goal by nearly 20%.

Finance

Finance has been busy trying to keep up with the orders booked by the sales staff. Our ability to collect our accounts receivable in a timely manner has allowed us to maintain a positive cash flow. However, Jerry has expressed some concern to me about our increasing expenditures. Please don't request unnecessary equipment or supplies so that we can keep our expenses under control.

As the following financial chart shows, sales during the quarter were strong, but increasing expenses have begun to cut into our expected profits.

	April	May	June
Sales (in $1000)	347	322	365
Expenses (in $1000)	$92	207	236
# Employees	36	36	38
Profit/Employee	$4300	$3200	$3400

APPENDIX B

QUESTION CORPUS I

This appendix contains the corpus of questions collected during the first user experiment (see Chapter 3). The questions are sorted by task and by subject within each task. Each question has a six digit encoding preceding it. The first two digits represent the task number. The next two digits are the subject number. The final two digits encode the question type (see Section 4.3).

(1) [01-01-01] How do I use italics?
(2) [01-02-02] How do you highlight text?
(3) [01-03-01] I'd like help on italics.
(4) [01-03-01] I'd like help on italics again.
(5) [01-04-01] How do I change font style?
(6) [01-04-01] I'd like to change this to umm italic, how do I do that?
(7) [01-05-01] I need some italics, er italization directions.
(8) [01-06-01] How do I italicize?
(9) [01-07-01] I need to know how to put a text in italics.
(10) [01-07-02] I need to know how to highlight a portion of text.
(11) [01-08-01] How do I italicize text?
(12) [01-09-01] Italics. Help with italics.
(13) [01-10-01] How do I italicize?
(14) [01-11-01] How do I italicize?
(15) [01-12-02] How do I place cursor?
(16) [01-12-01] How do I italicize?
(17) [01-13-01] How do you italicize?
(18) [01-14-01] How do you change fonts?
(19) [01-14-01] Which font is italics?
(20) [01-15-01] Explain type face changes.
(21) [01-15-01] How do I change a font?
(22) [01-15-01] I'd like to change this font to italics.
(23) [01-16-01] Hi, I want to know about italics.
(24) [01-17-01] How do I italicize text?
(25) [02-01-01] How do I bold?
(26) [02-01-01] What happens if I don't highlight first?
(27) [02-02-01] How do you change to bold?
(28) [02-03-01] I'd like to know if you can repeat a format for different ranges.
(29) [02-04-01] How do I make this bold?
(30) [02-06-01] How do you do the bold feature?
(31) [02-07-03] I need to make a text larger and in bold letters.
(32) [02-07-01] I need to know how to do different size letters or different fonts.

(33) [02-08-01] How do I bold characters?
(34) [02-09-01] Help with bold-face.
(35) [02-10-01] How do I use bold?
(36) [02-10-01] How do I get rid of block?
(37) [02-11-01] How do I make something bold-faced?
(38) [02-12-01] How do I increase point size?
(39) [02-12-01] Can I repeat the last command?
(40) [02-14-01] How do you change the size of print?
(41) [02-14-01] How do I make the letters bold?
(42) [02-16-03] I want to know about enlarging and highlighting characters.
(43) [02-16-02] Highlighting characters this time. How do I highlight?
(44) [02-16-01] I want to know if I can do character changes in more than one place at one time that are not connected.
(45) [02-17-01] How do I make things bold?
(46) [02-17-01] How can I repeat commands?
(47) [03-01-01] How do I insert text?
(48) [03-01-01] How do you add a accent over a letter?
(49) [03-02-01] How do you insert text?
(50) [03-02-01] How do you insert a tilde over a letter?
(51) [03-03-01] I'd like some help on foreign characters.
(52) [03-03-01] I'd like help on characters that are not English characters.
(53) [03-03-01] I'd like to know how to make an 'n' tilde.
(54) [03-03-02] I'd like help on spelling.
(55) [03-03-02] I'd like help on spell checking a document.
(56) [03-04-01] How do I insert a uh sentence?
(57) [03-05-01] Information please on special characters.
(58) [03-05-01] Information please on enyeahs.
(59) [03-06-01] How do you add a character not on the keyboard?
(60) [03-06-01] How do you make the tilde?
(61) [03-07-01] I need to insert some text.
(62) [03-07-01] I need to add additional text into the text that I already have.
(63) [03-07-01] I need to know how to do a Spanish character.
(64) [03-07-01] I need to know how to do Spanish characters.
(65) [03-07-01] I need a Spanish enyeah or tilde.
(66) [03-08-01] How do I insert text?
(67) [03-08-01] How do I insert a an extra space?
(68) [03-08-01] How do I insert a character over a letter as a foreign word? For example, 'Nunez'?
(69) [03-09-01] Help with tilde. Help with enyeah or help with special characters.
(70) [03-10-01] How do I get special characters?
(71) [03-10-01] I need a tilde.
(72) [03-11-01] How do I add text?
(73) [03-11-01] Is there an enyeah in the system?
(74) [03-12-01] How do I begin adding text?
(75) [03-12-01] Why is system typing in bold?

(76) [03-12-02] How do I undo bold typing?
(77) [03-12-01] Um a tilde above the 'n' above a letter?
(78) [03-13-02] How do you find the 'n' character with the wavy line over the top?
(79) [03-14-01] How do I add text?
(80) [03-14-01] How do I make a Spanish 'n'?
(81) [03-15-01] Um, I need a special Spanish character.
(82) [03-15-01] Uh, I need a Spanish 'n'.
(83) [03-16-01] How do I add something to the text?
(84) [03-16-01] How do I add the tilde to the 'n'?
(85) [03-16-02] I need to spell check one word.
(86) [03-16-01] One word only on the spell check.
(87) [03-17-01] How do I put uh tildes over letters?
(88) [04-01-01] How do I center?
(89) [04-02-01] How do you insert text at the top of a document?
(90) [04-02-01] How do you center?
(91) [04-03-01] I'd like to know how to center text.
(92) [04-04-02] How do I center something?
(93) [04-05-01] Assistance on centering please.
(94) [04-06-01] How do you center a title?
(95) [04-07-01] I need to center some text.
(96) [04-08-02] How do I insert a title at the beginning of a document?
(97) [04-08-01] How do I center a title?
(98) [04-09-03] Help with title. Help with centering.
(99) [04-09-01] Help with centering.
(100) [04-09-01] Help with font size.
(101) [04-09-01] Help with centering.
(102) [04-10-01] I need center.
(103) [04-11-02] How do I put on a title?
(104) [04-11-02] How do I center something?
(105) [04-12-01] How do I go to top of the page?
(106) [04-12-01] How do I go to top of file?
(107) [04-12-01] How do I center text?
(108) [04-12-01] How do I insert blank lines?
(109) [04-12-01] How do I create blank space between text?
(110) [04-12-01] How do I create a blank line?
(111) [04-13-01] How do you center text?
(112) [04-13-01] How do you change font size?
(113) [04-14-01] How would I center words?
(114) [04-15-01] Explain to me, centering or alignment.
(115) [04-16-02] How do I put a title on the first page?
(116) [04-16-01] Centering text. How do I center text?
(117) [04-17-01] How do I center text?
(118) [05-01-02] How do I do superscript?
(119) [05-01-01] How do I do footnotes?
(120) [05-02-01] How do you insert a footnote?

(121) [05-03-01] I'd like help on footnotes.
(122) [05-03-01] How do I make footnotes?
(123) [05-04-01] How do I insert a footnote?
(124) [05-04-02] How do I cancel a footnote?
(125) [05-05-01] Information on footnotes please.
(126) [05-06-01] How do you do a footnote?
(127) [05-07-01] I need to know how to make a footnote.
(128) [05-08-01] How do I create footnotes?
(129) [05-09-01] Help with footnotes.
(130) [05-09-01] Help with notes.
(131) [05-09-01] Footnotes.
(132) [05-10-01] I need footnote.
(133) [05-11-01] How do I do a footnote?
(134) [05-12-01] How do I create a footnote?
(135) [05-13-01] How do you add footnotes?
(136) [05-13-01] How do you save?
(137) [05-14-01] How would I add a footnote?
(138) [05-15-01] Uh, I need help with footnoting.
(139) [05-16-01] Tell me how to do footnotes.
(140) [06-01-01] How do I cut and paste?
(141) [06-02-01] How do you move a block of text?
(142) [06-04-01] How do I move a doc uh a paragraph?
(143) [06-05-01] Information on cutting and pasting please.
(144) [06-06-01] How do you move a paragraph?
(145) [06-08-01] How do I move blocks of text to different positions in the document?
(146) [06-10-03] How do I block and move?
(147) [06-11-01] How do I move text up to a different place?
(148) [06-12-01] How do I move text?
(149) [06-12-01] How do I see second page of text?
(150) [06-12-01] How do I delete a paragraph marker?
(151) [06-14-01] How do you move text?
(152) [06-16-01] How do I move a block?
(153) [07-01-01] How do I do bullets?
(154) [07-02-01] How do you put an accent mark over a letter?
(155) [07-02-01] How do you indent a line of text?
(156) [07-02-01] How do you insert bullets?
(157) [07-03-01] I'd like to know about French characters.
(158) [07-03-01] I'd like to know how to bullet a list.
(159) [07-03-01] I'd like to know about bullets, how to insert bullets.
(160) [07-03-01] I'd like to know how to indent.
(161) [07-03-01] I'd like to know how to bullet again.
(162) [07-04-01] How do I put in bullets?
(163) [07-04-01] How do I insert a tab?
(164) [07-05-01] Information on bullets please.

(165) [07-05-01] Information on indentation please.
(166) [07-05-01] Special character information.
(167) [07-06-01] How do you offset a subset of data?
(168) [07-06-02] How do you outline?
(169) [07-06-01] What's a bullet?
(170) [07-06-01] How do you indent?
(171) [07-07-01] I need to know how to set tabs, I guess.
(172) [07-07-01] I need some kind of a dot to indicate items of a list.
(173) [07-08-01] How do I create bullets?
(174) [07-08-01] How do I place tildes?
(175) [07-08-01] How do I place accent marks?
(176) [07-08-01] How do I how do I place tabs? How do I create a tab?
(177) [07-08-01] How do I create bullets?
(178) [07-09-01] Accent.
(179) [07-09-02] Lists.
(180) [07-09-01] Bullets.
(181) [07-10-01] I need an accent over an 'e'.
(182) [07-10-01] How do I use tab?
(183) [07-11-01] How do I place an accent over a letter?
(184) [07-11-01] How do you set a tab?
(185) [07-12-01] How do I place an accent mark above a character?
(186) [07-12-01] How do I indent text?
(187) [07-12-01] How do I indent just one line of text?
(188) [07-12-01] How do I create a bullet?
(189) [07-12-01] How do I delete lines?
(190) [07-13-01] How do you place a heavy black dot as a marker at the beginning of a line?
(191) [07-14-01] How do I add an accent?
(192) [07-14-01] How do I create a tab?
(193) [07-14-01] How would I mark umm text by using a round circle dot?
(194) [07-15-01] I need some help with setting tabs or indenting.
(195) [07-15-01] I'd like to put a bullet in front of these.
(196) [07-15-01] Uh, I'd like to find out how to put an accent.
(197) [07-16-01] Accents on vowel sounds.
(198) [07-16-03] How do I indent and make bullets?
(199) [07-17-01] How do I make accents over letters?
(200) [07-17-01] How do you tab?
(201) [07-17-01] How do I set tabs?
(202) [07-17-01] How can I edit tab stops using the ruler icons?
(203) [07-17-01] How do you make bullets?
(204) [08-02-01] How do you page down?
(205) [08-02-01] How do you make a table?
(206) [08-02-02] How do you insert text within a table?
(207) [08-03-01] I'd like to be able to create a table.
(208) [08-04-01] How do I make a table?

(209) [08-05-01] Table information please.
(210) [08-05-01] Information on boxing a table.
(211) [08-09-01] Table.
(212) [08-11-01] How do I insert a table?
(213) [08-13-01] How do you create a table?
(214) [08-15-01] Um, I need to change the size of the table.
(215) [08-15-01] Um, help with table row size.
(216) [08-15-01] Uhmm, I'd like to find out about editing a cell in a table.
(217) [08-15-03] I need to know how to change a cell in this table. Change the information in it, correct it.
(218) [08-15-01] I need to correct information in my table.
(219) [08-16-01] I need to make a table. Show me how to make a table.
(220) [09-01-01] How do I center a table?
(221) [09-02-01] How do you center a table?
(222) [09-04-01] How do I center this table?
(223) [09-11-01] How do I center a table?
(224) [09-13-01] How do you center a table?
(225) [09-16-01] I need to center a table.
(226) [09-17-01] How do I center a table?
(227) [09-17-01] How do I select a table?
(228) [10-09-01] Resize columns in table.
(229) [10-17-01] How do I adjust column width in a table?
(230) [11-09-01] Right justify margins.
(231) [11-17-01] How do I align text in a table?
(232) [12-03-03] I'd like to place text above a table and center it hopefully.
(233) [12-15-01] Fix table heading.
(234) [12-15-01] How do I add a title to this table?
(235) [12-17-01] How do I delete the title from a table?
(236) [14-01-01] How do I do numbering?
(237) [14-01-01] How do I do page numbering for a document?
(238) [14-01-01] How do I center a page number?
(239) [14-03-01] I'd like to find out how to put in page numbers.
(240) [14-04-01] How do I put in page numbers?
(241) [14-04-01] How do I center my page numbers?
(242) [14-05-01] Page numbering information.
(243) [14-06-01] How do you insert page numbers?
(244) [14-06-01] Page numbers.
(245) [14-07-01] I need to know how to do page numbers.
(246) [14-08-01] How do I place page numbers at the bottom of the page?
(247) [14-09-01] Page numbers.
(248) [14-09-01] Centering page number.
(249) [14-10-01] I want page numbers.
(250) [14-10-01] I want to delete a page number.
(251) [14-10-01] I want to center.
(252) [14-12-03] How do I create page numbers? How do I create footers?

(253) [14-12-01] How do I move a page number to the center of a footer?
(254) [14-13-01] How do you add page numbers?
(255) [14-13-01] How do you center a page number?
(256) [14-14-01] How do I add page numbers?
(257) [14-14-01] How do I center the page number?
(258) [14-14-01] How do I center a page number?
(259) [14-16-01] I need to add page numbers. Page numbers.
(260) [14-16-01] How do put the page numbers so they show up?
(261) [14-16-01] How do I center the page number?
(262) [14-17-01] How do I insert page numbers?

APPENDIX C

QUESTION CORPUS II

This appendix contains the corpus of questions collected during the second user experiment (see Chapter 8). The questions are sorted by task and by subject within each task. Each question has a six digit encoding preceding it. The first two digits represent the task number. The next two digits are the subject number. The final two digits encode the question type (see Section 4.3).

(1) [01-02-01] How do I italicize?
(2) [01-02-02] How do I highlight?
(3) [01-03-01] Will you put the American journal of widget users in italics?
(4) [01-04-01] How do I italicize?
(5) [01-12-01] How do I italicize?
(6) [01-13-01] How do I make a character over another character?
(7) [01-14-01] Change words to italic form, not sure how to go about it.
(8) [01-14-01] I need to change words into italic form and need help.
(9) [01-16-01] How do I italicize a phrase?
(10) [02-01-01] I need help bolding multiple entries.
(11) [02-02-02] How do I emphasize words?
(12) [02-03-01] I need to change font to bold.
(13) [02-14-01] I need to know how to take the dark shade off the screen.
(14) [02-14-01] Need to find the paragraph that disappeared.
(15) [03-01-01] Help Spanish alphabet.
(16) [03-01-01] Help concerning foreign alphabets.
(17) [03-03-01] I need to add a sentence.
(18) [03-04-01] How do I use foreign characters?
(19) [03-05-01] How do you use a tilde?
(20) [03-05-01] How do you use accents?
(21) [03-05-01] What does overstrike mean?
(22) [03-07-01] Find foreign language characters.
(23) [03-07-01] How do I access international characters?
(24) [03-09-01] How do I do a tilde on top of this n.?
(25) [03-09-01] How do I undo what I just did?

(26) [03-10-01] Show special characters.
(27) [03-11-01] How do I do international characters?
(28) [03-12-01] How do I accent a letter?
(29) [03-12-01] How do I put a symbol on top of a letter?
(30) [03-12-01] How do I put in a symbol?
(31) [03-13-01] How do I put an enyeah over an n?
(32) [03-15-01] Reveal codes.
(33) [03-15-01] Tilde.
(34) [03-15-02] Superscript.
(35) [03-16-01] How do I put an accent mark over a letter?
(36) [03-16-01] How do I put an accent mark over a letter?
(37) [03-16-01] I need to create an overstrike. The directions are not clear. How do I put an easy accent mark over a letter?
(38) [03-17-01] How do you make accent marks in foreign language words?
(39) [03-17-01] What are those little squiggly lines above the letter?
(40) [04-01-01] Help headers.
(41) [04-03-01] Can I center one line?
(42) [04-03-02] I need to add a title.
(43) [04-05-01] How do you center?
(44) [04-07-01] Font size.
(45) [04-09-01] How do I center a title?
(46) [04-10-01] Show centering.
(47) [04-13-01] How do I number my pages?
(48) [04-14-01] I need to make the size of the letters larger.
(49) [05-02-01] How do I designate an endnote in my paragraph?
(50) [05-03-01] I need to add a footnote.
(51) [05-05-01] How do you create footnotes?
(52) [05-06-01] How do I delete footnotes?
(53) [05-07-01] Footnotes.
(54) [05-09-02] How do I a superscript?
(55) [05-13-01] How do I add a footnote?
(56) [05-14-01] I need to know how to make footnotes.
(57) [05-16-01] How do I put in a footnote?
(58) [06-03-01] I need to add text.
(59) [06-03-01] I need to move a paragraph.
(60) [07-01-01] Help bullet.
(61) [07-03-01] I need to tab over.
(62) [07-03-01] I need to type text after my bullet point.
(63) [07-03-01] I need to use a bullet point.
(64) [07-04-01] How do I increase indent?
(65) [07-04-01] How do I indent bullets?
(66) [07-04-01] How do I make bullets?
(67) [07-05-01] How do you indent bullets?
(68) [07-05-01] How do you insert bullets?
(69) [07-06-01] How would I make bullets?

(70) [07-07-01] Bullets.
(71) [07-09-01] How do I get that accent mark over the 'e'?
(72) [07-09-01] How do I indent?
(73) [07-11-01] How do I indent a bullet?
(74) [07-13-01] How do I put a bullet before a character?
(75) [07-13-01] How do I put an accent mark over a character?
(76) [07-15-01] Bullets.
(77) [07-15-01] Indent.
(78) [07-16-01] How do I create a bullet-ed list?
(79) [07-16-01] How do I tab a series of text?
(80) [07-16-01] How do I tab in a bullet-ed list?
(81) [07-16-01] You've got to help me out here.
(82) [07-17-01] How do I get the bullet right next to the letter?
(83) [07-17-01] How do I indent bullets?
(84) [08-01-01] Help charts.
(85) [08-01-01] Help remove chart.
(86) [08-02-01] How do I enter a box?
(87) [08-03-01] I need to add a table.
(88) [08-03-01] I need to edit in the table.
(89) [08-05-01] How do you make a chart?
(90) [08-05-02] How do you create a spreadsheet?
(91) [08-09-01] How do I make a table?
(92) [08-16-01] How do I create a table with numbers and text?
(93) [08-17-01] How do I edit tables?
(94) [09-03-02] I need to make one line change in the table.
(95) [09-03-02] I need to make one line double.
(96) [09-04-01] How do I apply a table style?
(97) [09-10-01] Show table styles.
(98) [09-12-01] How do I double lines in a table?
(99) [09-15-01] Table double underline.
(100) [09-17-02] How do you add graphics to tables?
(101) [11-07-01] How do I center a table?
(102) [11-12-01] How do I center a table?
(103) [11-13-01] How do I center a table?
(104) [14-04-01] How do I insert page numbers?
(105) [14-05-01] How do you number pages?
(106) [14-10-01] Number pages.
(107) [14-11-01] Show me footers.
(108) [14-12-02] How do I enter a footnote a footer?
(109) [15-04-01] How do I change margins?
(110) [16-04-01] How do I insert a line division?