The Design and Implementation of an Intelligent Interface for Information Retrieval

Roger Howard Thompson
Ph.D. Thesis
Computer and Information Science Department
University of Massachusetts

COINS Technical Report 88-88

Computer and Information Science

University of Massachusetts at Amherst

The Design and Implementation of an Intelligent Interface for Information Retrieval

A Dissertation Presented

by

ROGER HOWARD THOMPSON

Submitted to the Graduate School of the

University of Massachusetts in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

February 1989

Department of Computer and Information Science

© Copyright by ROGER HOWARD THOMPSON 1989

All Rights Reserved

The Design and Implementation of an Intelligent Interface for Information Retrieval

A Dissertation Presented

by

Roger Howard Thompson

Approved as to style and content by:

[illegible]

W. Bruce Croft, Chairperson of Committee

[illegible]

David W. Stemple, Member

[illegible]

Nicholas J. Belkin, Member

[illegible]

Joseph Sardinas, Outside Member

[illegible]

W. Richards Adrion, Department Head
Department of Computer and Information Science

DEDICATION

This work is dedicated to the memory of

Dr. Victor Paul Wierwille.

ACKNOWLEDGMENTS

I would like to thank the following people, who helped me greatly to accomplish this work. First, I would like to thank my advisor Bruce Croft, whose constant encouragement and thoughtful constructive criticism was instrumental in helping see this project through. Professors Dave Stemple and Nick Belkin provided me with different perspectives that enabled me to think more clearly.about the subject.

I would like to thank some of the residents of the Wombat Research Lab, Larry Lefkowitz, Carol Broverman, Tom Parenty, Norm Carver, and Al Hough for making the time bearable.

I would like to thank my supervisors at Hughes Aircraft, Bill King and Jim Blackburn for their understanding while finishing my writing.

I would like to thank my friend, Andy Zitelli, for his wise counsel throughout my entire undergraduate and graduate academic career.

Finally, I would like to express my gratitude to wife, Darlene, for her unfailing encouragement and support, and my daughter Rebeca for the joy that only a young child can bring.

ABSTRACT

THE DESIGN AND IMPLEMENTATION OF

AN INTELLIGENT INTERFACE FOR

INFORMATION RETRIEVAL

February, 1989

ROGER HOWARD THOMPSON, B.A., UNIVERSITY OF CALIFORNIA AT BERKELEY

M.S., NEW MEXICO STATE UNIVERSITY

Ph.D., UNIVERSITY OF MASSACHUSETTS

Directed by: Professor W. Bruce Croft

Commercial information (text) retrieval systems have been available since the early 1960's. While they have provided a service allowing individuals to find useful documents out of the millions of documents contained in online databases, their are, a number of problems that prevent the user from being more effective. The primary problems are an inadequate means for specifying information needs, a single way of responding to all users and their information needs, and an inadequate user interface.

This thesis describes the design and implementation of I^3R, an intelligent interface for information retrieval the purpose of which is to overcome the limitations of current information retrieval systems by providing multiple ways of assisting the user to precisely specify his information need and to search for information. The system organization is based on a blackboard architecture and consists of a number of "experts" that work cooperatively to assist the user. The operation of the experts is coordinated by a control expert that makes its decisions based on a plan derived from the analysis of human search intermediaries, end user dialogues, and user model. The experts provide multiple formal search strategies, the use and collection of domain knowledge, and browsing assistance. The operation of the system is demonstrated by four scenarios.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

OVERVIEW

1.1 Introduction

In this chapter, an overview of this dissertation is presented. We begin by discussing the problems of traditional information retrieval systems and how they are usually overcome. These problems form the basis for the requirements of a more sophisticated system called I^3R, an Intelligent Interface for Information Retrieval. A design is then outlined that will meet the specified requirements. The design has two major aspects: the first is facilities that should be provided; the second is how these facilities are to be supported in ways that allow easy modification.

1.2 Retrieval Problems

Commercial retrieval systems have been available since the early 1960's. At that time, they were a significant breakthrough in the use of computers for non-numeric applications. They allowed scientists and engineers to sort through the many journals, technical reports, and other written works to find information that might be useful in helping them solve their problems. The utility of these systems has been recognized in other professions such as law and medicine, where major retrieval services are now available.

While developments in storage technology, such as ever increasing densities in disk storage, and developments in communications technology, such as relatively inexpensive 2400 baud modems, have made these systems more widely available, the interface technology has remained for the most part stagnant, reflecting the designs of the original systems. These interfaces were designed to operate with simple input/output devices such as 110 character/second printing terminals. This significantly limits the kind of information that can be displayed. Furthermore, the operation of the system has a command-line orientation, which is reflected in the use of specialized languages for query specification.

These languages are based on Boolean logic and are usually augmented with proximity operators and "don't care" or wildcard characters. The former specify how close words must be in sentences or paragraphs. The latter handle alternative spellings and inflected forms of words. The use of these languages requires specialized training for the user to teach them the semantics for AND, OR, and NOT. While the basic concepts are relatively simple, use of these languages is mastered only after a significant amount of experience. Furthermore, different systems have different query languages and many users do not have the time or the inclination to learn Boolean logic.

Boolean logic cannot precisely specify many relationships between words. For example, AND can be used to describe phrases or words that are required; OR may specify alternative words, synonyms, or components of "higher" level concepts. In addition, AND and OR in some situations in everyday language can be used synonymously. This lack of precision or multiple meaning can be overcome by adding other operators to specify relationships more exactly or adding weights to the AND and OR to give "soft" Boolean operators [Salton 83].

Both solutions, while feasible, simply add to the amount of knowledge that the user must know in order to use a system effectively. This increases the potential for confusion and, hence, frustration on the part of the end user. The casual user or "permanent novice" will, in all likelihood, never bother to learn how to use the advanced features of the query language.

Compounding the problem of using the query language, which is a matter of query form, is the problem of determining precisely what is the content of the query. This is a problem of selecting the proper words to express what the user wants. Two potential problems arise here. The first is that the user may not know exactly what he wants, and the second is that he may not know the precise terminology required to express the need. In some systems, the user has recourse to an online thesaurus, which is a collection of words that is structured to show the relationships between them, to find the proper descriptive terms and to give the structure of the knowledge of a domain. In others, the best that he can do is get an alphabetical list of terms occurring in the database.

The problems of query form and content are manifestations of the inflexible nature of retrieval systems. They have only one way to respond to every type of user and every type of problem.

To overcome this inflexibility, end-users, the persons with the information need, often resort to using the services of a search intermediary. Intermediaries have received specialized training in the use of retrieval systems. They often have a degree in librarianship or have a degree in the field in which they search or by constant use have developed a knowledge of the terminology of a domain. For example, an intermediary that searches Chemical Abstracts might have a Ph. D. in chemistry. This background allows them to concentrate on getting the best possible results from the retrieval system by knowing the correct terminology.

One of the main advantages of using intermediary services is that the intermediary, being a person, can be much more flexible than the current commercial systems. The intermediary can adapt to the needs of different users. If the session is the end-user's first experience, the intermediary can help the user understand the search process by explaining what he is doing as he goes along. The intermediary can adjust his explanations to match the kind of user that he is dealing with. A college freshman with an orientation to the humanities would require a different kind of assistance than a medical doctor with some computing experience. Another advantage of an intermediary is that he can continue to learn about the domains that people consistently search in and he can learn about the needs of the people that consistently use their services.

While the use of intermediary services removes the burden from the end-user of having to deal with the query language, and often provides him with terminological assistance, it adds a new difficulty, since the user is now often removed from participating directly in the search process. The user must now, as before, try to express his information need to the intermediary, but, in general, cannot take advantage of the recognition ability that humans have in the search process. This is due to the fact that often intermediaries will search without the user present. The preferred situation is to have the end-user present with the search intermediary while the search is taking place. This, however, often slows the intermediary down, since he often has to explain his actions to the end user. This situation is not always possible due to considerations such as scheduling, among others. Other factors such as the availability of intermediary services also come into play. These services may not be free; adding further to the cost of using the system. Furthermore, with the advent of extremely high density storage such as CD-ROM (Compact Disk–Read Only Memory), end-users may be searching for information in their own home, where search intermediaries are not available.

1.3 The Intermediary Model

The search intermediary provides a model that can be useful in designing systems that can help overcome the problems of using IR systems. There are two ways that this concept can be used. One way is to simulate the activity of an intermediary, that is to attempt to provide the same services as the intermediary. This has been the basis of a number of expert systems that provide such services as a common command language to multiple retrieval systems [Marcus 81a, Marcus 81b, Marcus 83] and rudimentary query formulation assistance [Yip 79, Pollitt 84]. More sophisticated systems [Brajnik 85, Brajnik 87, Chiaramella 87, Defude 85] that take this approach attempt to implement the strategies and tactics used by intermediaries for searching [Bates 79a, 79b] and attempt to incorporate a natural language dialogue with the user. All of the systems that attempt this kind of simulation have been designed to work with Boolean systems and therefore have the limitations on retrieval effectiveness [Salton 83] that plague Boolean systems.

The approach taken in this thesis is to look at the intermediary concept as an intelligent interface system which is composed of the intermediary and the retrieval system. Analysis can then be made of the kinds of facilities that this system provides or should provide to assist the user in expressing his need and finding information that will meet it. The system designer can then determine how best to implement those facilities, taking advantage of the current research in information retrieval, and not be limited to ineffective, immature, or inappropriately applied technologies in an effort to exactly simulate the human intermediary.

1.4 System Analysis and Requirements

In analyzing the combined intermediary/retrieval system, the four basic elements of a retrieval system are the basis of the analysis. These basic elements are:

1. a representation of the content or meaning of the documents and the queries,
2. a process, usually called indexing, that maps the content of the document and the queries into the content representation,
3. a decision method, usually called a search strategy, that the system uses to determine whether or not a document should be retrieved,
4. a user interface.

The user interface element in the combined intermediary/retrieval system is composed of the services that the search intermediary provides, and the actual method (i.e. how the query is typed in, how results are displayed, etc.) of interacting with the system. The essential services that the intermediary provides are:

1. explanation of system operation,
2. term selection assistance,

3. construction of a model of the information need, which consists of the query and the documents that have been retrieved,
4. execution of the searches,
5. overall control of the course of a session.

To adapt to the different kinds of end-users, the intermediary must make some assessment with regard to the end-user's familiarity with the domain, his familiarity with the search process, and the kind of results that he wants, such as whether he wants a few specific documents or a comprehensive collection. Essentially, the intermediary forms a model of the end-user and adapts the session to that model.

While the intermediary aspect of the system addresses most of the issues of inflexibility, some of them are rooted in the retrieval portion of the system. In the past, systems have been limited to a single decision method (retrieval method) for determining what documents ought to be retrieved. By having different methods for different kinds of queries the effectiveness of a system can be increased substantially [Croft 85]. A system's effectiveness can also be increased by providing direct access to the documents by browsing, a heuristically driven incremental search and evaluation technique [Oddy 77]. Browsing need not be limited to just the examination of documents; it can also be used to find the appropriate concepts to describe the information need.

The preceding high level analysis of the elements of the combined intermediary/retrieval system has pointed out the need for the system to support a number of facilities or functions that either provide services similar to that of an intermediary or support functions that are part of the underlying retrieval system. These functions or services can be summarized in the following modules:

1. Explainer – explains system operation to the user,
2. Domain Knowledge Expert – suggests additional concepts to the user and acquires domain knowledge from the user,
3. Request Model Builder – maintains information about the current state of the session such as relevant concepts and relevant documents,

4. Search Controller – chooses search techniques that are appropriate to the current state of the session and information need,

5. User Model Builder – determines what kind of end-user is currently interacting with the system,

6. Browsing Expert – provides recommendations to the user about information to view that is likely to be relevant when the user is browsing, and remember the path that the user has taken during browsing,

7. Control Module – determines the direction of the "dialogue" that system has with the user.

The representation for the documents must contain all the information necessary to support multiple search strategies and browsing. Traditional systems have usually maintained simple inverted files that would be inadequate in this case. In addition, thesaurus information in most systems has not been integrated into the overall retrieval process.

A number of other factors come into play in determining what the requirements of the retrieval system should be. One important factor of traditional information retrieval systems that is desirable to maintain is their domain independence. This means that the system cannot depend on having a significant amount of domain specific knowledge. However, since domain knowledge is very useful in assisting the user to precisely express his information need, the system should have the ability to use whatever domain specific knowledge that is available, and should be able to acquire this knowledge from the user.

1.5 Architecture

In order to build a system that provides the kinds of facilities that the combined search intermediary/retrieval system does, it must have an architecture that allows it to be flexible. This flexibility is manifested in a number of different ways. First, the system must adapt itself to different kinds of users and different kinds of information needs; this is external flexibility. Second, it must be flexible enough so that it can incorporate new techniques as they are developed; this is internal flexibility.

The first kind of flexibility requires that the system changes the way it interacts with the user as does the intermediary. For a novice user, it should offer more explanation and assistance, and it should limit his choices so that he does not get in to a situation that he cannot handle; for an expert user it should not interfere with his use of the system, and should provide him access to all of the system's functionality. Another aspect of this flexibility is that different kinds of information needs require different kinds of searches. The system must be able to respond appropriately.

The second kind of flexibility requires an architecture that is modular in nature. This modularity should be at two levels. It should be able to support the addition of new large pieces of functionality. This would allow it to take advantage of new developments in information retrieval research. Each large scale function should also be modular, so that it can be adjusted to operate more effectively as the pattern of system usage is established. It also allows for the integration of new developments. For example, if a new search technique is developed that is particularly good at retrieving relevant information for one kind of information need, it can be incorporated into the search function of the system.

The architecture that best supports the requirements of an intelligent IR interface is a modified blackboard architecture [Erman 80, Nii 86a Nii, 86b]. A blackboard architecture, of which Hearsay II is a typical example, consists of a number of independently operating modules, called knowledge sources, that work together to solve a problem. Each works on a particular aspect of a problem. The results of their work is posted on a shared data structure called a blackboard. This blackboard is typically organized as a series of levels that represent abstraction levels of the problem. The operation of the knowledge sources is coordinated by a scheduler.

The basic operation of a blackboard system is as follows. First, each expert examines the state of the blackboard in its area of interest. It then decides if it has any action that it would like to perform based on the current conditions. If it does, it places an action (called an instantiation) on the system agenda. The agenda is examined by the scheduler and is sorted in order of importance based on criteria that are problem dependent. The scheduler then takes the most important action and runs it. The cycle then begins again.

The blackboard architecture is appropriate since supports the easy addition of large scale functions by means of knowledge sources. In addition, the way that knowledge sources are to be implemented is not specified, so they can be implemented in the way that is most appropriate for their specific task. The knowledge sources in I^3R are called experts since they are implemented as individual rule based systems. This provides a means of incrementally developing the experts. These experts correspond to the functions that were derived in the system analysis.

The basic blackboard architecture must be adapted to fit the nature of the information retrieval problem. The first adaptation is to the structure of the blackboard; it is not structured into abstraction levels, since there is no single overall hierarchical representation that can be applied to IR. Instead, the blackboard, called the short term memory, consists of different models built by the experts in the course of the session.

The purpose of the control function in I^3R also differs from that of the scheduler in a typical blackboard system. In a typical system, the scheduler manages the system's resources to come to the solution of the problem in the shortest time possible. In I^3R the control expert manages the dialogue the system has with the user, so that it is consistent and coherent. This difference stems from the fact that information retrieval is better likened to a process than to a problem to be solved. The control expert makes sure that the process is conducted correctly.

The control function uses information provided by the user model builder and the request model to determine the course of a session. The information for the user model builder is based on the stereotypes that the UMB decides apply to the particular user for the particular session. Stereotypes are models of different kinds of typical users. In the current system three general categories are used, with two values for each category. The categories are domain expertise, search system expertise, and search type. The values are novice and expert for the first two categories, and selective or exhaustive for the third category.

The documents, concepts, and user histories are kept in a long term memory. The user histories store information about the user obtained from previous sessions with the system. This includes the original query, concepts that were judged relevant, documents that were judged relevant, and the stereotypes that were in effect at the end of the session. Also included in the user histories is a model of the whatever domain knowledge that the user has contributed in the course of his interaction with the system.

The system also maintains a store of global domain knowledge that is derived from available sources such as thesauri, and domain experts that use the system. This store is organized as semantic net [Quillian 68] with the concepts being the nodes and the links being the relationships. Stored with the concepts nodes is their frequency of occurrence in the document collection. Included with the normal conceptual relationships is a statistical nearest neighbor relationship that reflects the occurrence of concepts together in documents of the collection.

The documents are represented by lists of concepts that occur in them (authors are also considered concepts) and their frequency in the document. The lists are determined using a standard automatic indexing technique [Porter 80]. Additionally, citation information is retained along with the document nearest neighbors, which is a link based on the similarity of the representations of two documents. Other information such as the date and journal is included. The combination of the user domain knowledge model, global domain knowledge model, and document database forms the concept/document knowledge base.

The concept/document knowledge base supports all of the traditional search techniques, as well as providing a structure that the user can browse. Browsing is considered an important alternative method for finding information and incrementally specifying an information need. It allows users to use their recognition abilities to confirm or deny the relevance of items presented for display. By doing this, a model of their information need is built up.

The original Hearsay II system had little in the way of requirements for sophisticated input and output, whereas information retrieval needs a sophisticated user interface. Consequently, a separate interface manager is added to provide a window-based environment. Where the experts are responsible for what is displayed and when, the interface manager is responsible for how information is displayed and collected. The interface manager communicates with the experts by placing messages on and receiving messages from the short term memory.

The overall organization of the system is shown in figure 1.1

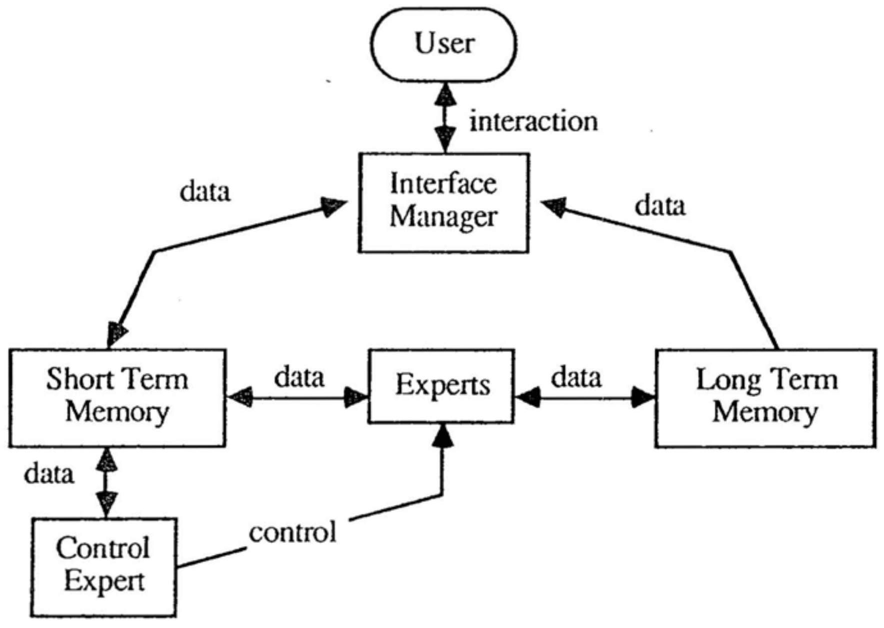

Figure 1.1: System organization of I^3R.

1.6 Organization

This chapter has provided a high level overview of the motivating factors behind and architecture of I^3R, Intelligent Interface for Information Retrieval. The remainder of this thesis is organized as follows. In chapter two an extensive review of the information retrieval field is presented. This covers the basic principles of IR research, the problems of traditional IR systems, and a number of systems that have attempted to attack different problems relating to IR interfaces. Based on this survey and analysis, chapter three sets forth the requirements for an intelligent IR interface. Chapter four presents the design and the implementation of a system, I^3R, that meets these requirements. Chapter five provides more detail on the browsing expert. Chapter six presents a discussion of the difficulties in evaluating a system such as I^3R, and three example scenarios that give the flavor of how I^3R operates. And, finally, chapter seven presents the conclusion and directions for future research.

1.7 Contributions

This thesis, presenting an architecture for an intelligent information retrieval interface, makes the following contributions to the art and science of information retrieval:

- The incorporation of multiple automatic search methods into one system.
- The integration of browsing, a user-directed search method, into a system with automatic search methods for the purpose of both query refinement and information search.
- A concept/document knowledge organization that supports the use of multiple search strategies, browsing and the use of user supplied domain knowledge.
- The first use of user stereotypes in an information retrieval system to control the adaptation of the system's operation to individual users.
- The use and integration of user supplied domain knowledge, which allows the system to retain its domain independent design, but allows it to gain domain knowledge with use.

- A flexible control structure that allows the system to alter its operation based on the user and the progress that the user and the system make in satisfying the user's information need.

- Implementation of the major functions of the system as separate rule based systems that allows each function to be incrementally developed, and allows new major functions to be smoothly integrated.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Introduction

In this chapter the background and problems of traditional information retrieval are first presented. Next, the concept and limitations of "intelligent" retrieval are discussed. This forms the basis for examining research directed at increasing the performance and usability of information retrieval (IR) systems in the fourth section.

2.2 Traditional Information Retrieval

2.2.1 Definition

An information retrieval (IR) system is concerned with providing an individual with access or references to documents or other text that contain information that is likely to be relevant to the user's expressed information need. This differentiates IR systems from fact retrieval systems that attempt to provide discrete pieces of knowledge, for example, the melting point of a substance or the salary of an individual. Although database systems are considered information retrieval systems, for the purposes of this work they are considered as part of an implementation base that would support both text and fact retrieval.

2.2.2 Components

An IR system has four major elements:

1. A representation of the content or meaning of the documents and the queries,

2. A process, usually called indexing, that maps the content of the documents and queries into the content representation,
3. A decision method, usually called a search strategy, that the system uses to determine whether or not a document should be retrieved,
4. A user interface.

Each of these elements will be discussed in turn in the following sections. Additionally, consideration must be made of how the representations are organized and what other information is required for search and indexing.

2.2.2.1 Representation

The representation most often used is a keyword approach, where the text contained in the system is represented by a list of terms or concepts that are indicators of the information contained in the text. In order to conserve storage the text of these terms is replaced with a number, and a dictionary is maintained to map the number to the original text. Along with this, information such as the authors, date of publication, language, and journal issue identification may also be kept.

An alternative to keywords is a full text representation, where the entire text of the document is stored and available for search. Full text systems are typically used in domains such as law where the specific wording of text is of particular importance.

Additional information is often kept to support more sophisticated search techniques. In the document representations, the frequency of the concepts is retained. This information allows search methods to know what terms are important in a document, based on the assumption that the more frequent a term is in a document, the more important it is. Furthermore, some systems store the reference or citation information. This information may be composed of pointers to the documents in the reference list (cit*ed* documents) and, additionally, pointers to documents that reference a document (cit*ing* documents).

2.2.2.2 Indexing

The text must be transformed from its raw state into the representation. This is accomplished by either manual indexing, or automatic indexing.

2.2.2.2.1 Manual Indexing

Manual indexing systems rely on human judgement to decide what are appropriate keywords for describing a document's content. It is accomplished by a person reading the text and assigning keywords. These may be from a controlled vocabulary, like MeSH (Medical Subject Headings) that specifies all the keywords that can be used and their hierarchical relationships, from the text itself, or supplied by the user himself.

2.2.2.2.2 Automatic Indexing

Automatic indexing takes the text and removes stopwords, which are high frequency words like "and," "was," "the," and "it." For a typical list see [Van Rijsbergen 79, pp. 18-19]. The remaining words are stemmed (i.e., have their suffixes removed using a standard algorithm [Porter 80]). The stems are then used in a table lookup in a dictionary that maps stems to term numbers to find the corresponding term number. The primary reason for this lookup dictionary is to reduce the amount of space required to store all of the document representations. This is significant, since some document databases have millions of documents. If there is no corresponding number, a new is one generated and the stem/term number pair is added to the dictionary. This dictionary also contains a count of the occurrences of a term in the entire collection. This information is useful in various retrieval techniques to be discussed in the section on automatic retrieval. The term numbers for all the terms in the document along with their frequency in the document form the document representation.

Queries are processed in a similar manner. The only difference is that if there is a term in the query that is not found in the collection, it is deleted from the query, and not inserted into the dictionary.

To illustrate the automatic indexing process, consider the following document abstract :

```
The problem of the mutual exclusion of several indepen-
dent processes from simultaneous access to a critical
section is discussed for the case where there are two
distinct classes of processes known as "readers" and
"writers." The "readers" may share the section with each
other, but the "writers" must have exclusive access.
Two solutions are presented: one of the cases where we
wish minimum delay for the "readers"; the other for the
case where we wish writing to take place as early as
possible.
```

The first step is to remove the stop words and sort the remaining words, which results in the following list.

```
access, access, case, case, case, classes, critical, de-
lay, discussed, distinct, early, exclusion, exclusive,
independent, known, minimum, mutual, place, possible,
presented, problem, processes, processes, readers, read-
ers, readers, section, section, share, simultaneous,
solutions, take, two, two, wish, wish, writers, writers,
writing
```

Next the list of words is stemmed, and the list is compressed to add up the number of times that a stem is found in the document. The original words in this example are saved since this process is also used on the query, and the original words can be used to find related terms. The result is the following list of forms (in this example, the indexing algorithm is implemented in Lisp, and so it produces Lisp forms):

```
(access 2 (access)) (case 3 (case)) (class 1 (classes))
(critic 1 (critical)) (delay 1 (delay))
```

```
(discuss 1 (discussed)) (distinct 1 (distinct))
(ear 1 (early)) (exclus 2 (exclusion exclusive))
(independ 1 (independent)) (known 1 (known))
(minimum 1 (minimum)) (mutual 1 (mutual))
(place 1 (place)) (possibl 1 (possible))
(present 1 (presented)) (problem 1 (problem))
(process 1 (processes)) (reader 3 (reader))
(section 2 (section)) (share 1 (share))
(simultan 1 (simultaneous)) (solut 1 (solutions))
(take 1 (take)) (two 2 (two)) (wish 2 (wish))
(writer 2 (writers))
```

Next, the term numbers that correspond to the stems are looked up in the stem dictionary resulting in the following.

```
(10191 88 access 2 (access)) (11384 128 case 3 (case))
(36481 146 class 1 (classes))
(38351 26 critic 1 (critical)) (11673 5 delay 1 (delay))
(33740 356 discuss 1 (discussed))
(20610 21 distinct 1 (distinct))
(21787 21 ear 1 (early))
(25518 18 exclus 2 (exclusion exclusive))
(10445 71 independ 1 (independent))
(18391 79 known 1 (known)) (637 1 minimum 1 (minimum))
(3540 18 mutual 1 (mutual)) (25845 23 place 1 (place))
(29239 81 possibl 1 (possible))
(27663 253 present 1 (presented))
(31459 452 problem 1 (problem))
(2012 427 process 1 (processes))
(646 70 reader 3 (reader)) (9112 25 section 2 (section))
(9943 99 share 1 (share))
(12768 50 simultan 1 (simultaneous))
(6723 279 solut 1 (solutions)) (19792 50 take 1 (take))
(20170 8 two 2 (two)) (4183 10 wish 2 (wish))
(6626 126 write 3 (writers writing))
```

If this is a new document, the second number represents the frequency of the term in the collection, including the occurrences of the term in the new document. If this is a query, the number would not include the frequency count in the query. This list is then reduced to the term numbers and frequencies in the document resulting in:

```
(10191 2) (11384 3) (36481 1) (38351 1) (11673 1)
(33740 1) (20610 1) (21787 1) (25518 2 ) (10445 1 )
(18391 1) (637 1) (3540 1) (25845 1) (29239 1) (27663 1)
(31459 1) (2012 1) (646 3) (9112 2) (9943 1) (12768 1)
(6723 1) (19792 1) (20170 2) (4183 2) (6626 3)
```

Depending on what other kinds of information are maintained with the document, this is stored as the document's representation in the document database. If this process is applied to a query the complete information would be saved and forms the request .

The document database consists of all of the representations of the documents and the stem dictionary that maps the stems to the term numbers. Most systems also have an inverted file that has for each term, the documents in which it occurs. The use of an inverted file, while causing as much as a 100% increase in the space needed to store the document database, significantly increases the efficiency of searching.

2.2.2.3 Search Methods

There are two basic methods for deciding what documents to retrieve, user-directed and automatic.

2.2.2.3.1 User-Directed Methods

User-directed methods are characterized by being highly interactive and generally very slow. The first kind of user-directed method, which is found in nearly all commercial systems, is based on Boolean logic. In this case the form of the query and the decision method are the same. The query is essentially a decision rule in which the Boolean expression is the condition part and retrieval is the action part. A document is retrieved if it fulfills the conditions exactly. Most commercial systems also provide proximity operators so that the user can also specify phrases in the query formulation. In addition, these systems use an inverted file of the documents, containing for every term in the collection, the documents it is in.

Search in these systems is generally not performed by submitting a query and waiting for the results. Typically, the searcher will use the query as a guide to plan a series of actions that will retrieve documents. Often it is the case that the searcher will replan as he gets feedback during the search. The process begins by retrieving an initial set or sets of documents by using the terms provided by the user. Then, new terms or constructions made with adjacency operators are included using the AND, OR, and NOT operators as specified by the query expression. The query can be broadened by adding terms using the OR operator causing more documents to be retrieved, or it can be narrowed by using the AND and NOT operators causing fewer documents to be retrieved.

Another kind of user-directed method is *browsing*, which is characterized as an informal search that uses the structural links or connections between items in an organized body of information to look for relevant information. Browsing is often pursued when the user does not have a firm idea of what information exactly he desires, but has a general idea. He may use a classification system to help locate a large group of documents or books that are in the general area of his topic. From there he will pick some initial entry point and start to explore.. He will view and evaluate information in relation to his need, which may cause him to adjust his topic of interest.

2.2.2.3.2 Automatic Methods

Automatic methods are characterized as more batch oriented than the user-directed search methods. The user generally develops some specification of his information need (a query), submits it to the system, and waits for the results. The differences in the methods lie primarily in how they interpret the information provided by the document collection and the query, in the form of term frequencies.

2.2.2.3.2.1 Coordination measures

There are a variety of automatic search methods. The simplest kind of method counts the number of terms that the document and query representations have in common; this is called the coordination level. Retrieval is based on the ranking of the coordination level, but no ordering is done within a level. To take into account the differing sizes of documents and queries, the co-occurrence measure can be normalized. A typical normalized measure is Dice's coefficient [Van Rijsbergen 79] which is:

$$2 \cdot \frac{|D \cap Q|}{|D| + |Q|}, \qquad (2.1)$$

where D is the set of terms representing a document, Q is the set of terms representing the query, and |.| is the number of terms in the representation. For example, consider the following document, D, and query, Q, where

```
D =   ((1, 1) (2, 2) (3, 1) (4, 3) (5, 2) (6, 1)
       (7, 1) (8, 1) (9, 2) (10, 2) (11, 2) (12, 6))
Q =   ((6, 1) (7, 4) (10, 1) (12, 2) (15, 1) (16, 2)
       (20, 1)),
collection =  ((1, 21) (2, 23) (3, 14) (4, 16) (5, 81)
       (6, 13)  (7, 62)  (8, 23)  (9, 17)  (10, 5)
       (11, 14) (12, 18) (13, 23) (14, 31) (15, 6)
       (16, 14) (20, 5)).
```

The first value of the pair is the term number and the second is the frequency. The value for Dice's coefficient then is 2·4 / 10 + 7 = 0.47.

2.2.2.3.2.2 Vector space model

A more sophisticated approach is to consider the document and query representations as vectors in an n-dimensional space, where n is the number of unique terms in the collection. The similarity between a document and a query can then be formalized as the cosine of the angle between their respective vector representations, and is expressed as follows.

$$\frac{\sum_i (d_i \cdot q_i)}{\sqrt{\sum_i d_i^2 \cdot \sum_i q_i^2}} \qquad (2.2)$$

Where d_i is a term in D and q_i is a term in Q. The value for the cosine between the query and the document , from the previous example, if frequencies are not taken into consideration is 4 / 9.16 = 0.43.

Search is performed in these systems in the following way.

1. The query, Q, is indexed, converting it to a list of terms and their frequency in the query.
2. If no more terms, done, else get the next term.
3. Get the documents that this term occurs in.
4. If no more documents, then go to 2, else get the next document, D.
5. If this document has not been seen before, compute cos(D, Q) and place it in the ordered list of documents that have been seen.
6. Goto 2.

Associated with the terms can be weights which reflect the importance of the term in a document and in the entire collection. Based on empirical studies [Sparck Jones 77, Sparck Jones 80, Salton 83], the inverse document frequency (idf) weight has been shown to increase retrieval effectiveness. This weight is based on the observation that the less frequently a term is found in the collection the better discriminator it is. It is expressed as either log(N) - log (n_i) + 1, or N/n_i where N is the number of occurrences the most frequent term in a collection and n_i is number of occurrences of term i. Additionally, this can be multiplied by the term frequency (tf) within a document or a term significance weight (tsw) which is m_j/M, where M is the frequency of the most frequent term in the document and m_j is the frequency of a term in the document, so that the weight for a term in a document (d_i) is tf × idf or tsw × idf. This weight is based on the observation that the importance of a term in describing a document is directly related to the number of times it occurs. In essence, if a term occurs many times in a document,then that is a good indication of what the document is "about."

2.2.2.3.2.3 Probabilistic model

The next group of methods are based on probability theory [Van Rijsbergen 79]. This interpretation means that the system selects documents that have a high probability of being relevant to the query. The documents and queries are represented by the same sets of keywords or terms and use the same frequency information as the previous searches. The decision to retrieve is also based on a ranking with each document being scored by the following function:

$$\sum_i T(x_i)\ W(x_i)\ x_i\ , \qquad (2.3)$$

where x_i is the i th term in the vector of terms describing a document, T(x_i) is the term significance weight (tsw), W(x_i) is a weight related to the frequency of term i, in the collection of documents and to its frequency in relevant and non-relevant documents [Robertson 76, Sparck Jones 80]. The weight for each term is computed as:

$$\log \frac{r\ /\ (R-r)}{(n-r)\ /\ (N-n-R+r)}\ , \qquad (2.4)$$

where r is the frequency of a term in relevant documents, (n-r) is the frequency in non-relevant documents, (R-r) is the absence of a term in relevant documents, and (N-n-R+r) is the absence of a term in non-relevant documents. In practice 0.5 is added to each numerator and denominator to avoid division by zero [Sparck Jones 76], so that the weight is

$$\log \frac{(r + 0.5) / (R - r + 0.5)}{(n - r + 0.5) / (N - n - R + r + 0.5)}. \tag{2.5}$$

Since, in an initial search there is no information with regard to the frequency of a term in relevant or non relevant documents, this weight is estimated using the idf weight [Croft 80].

This computation is motivated by Bayesian decision theory [Van Rijsbergen 79]. The weight is also based on the assumption that terms are independent. Considering the case where terms are dependent requires more information than can be reasonably determined from the collection statistics.

However, the idea of term dependencies is useful and leads to some information that can be reasonably be used. An assumption can be made, called the *Association Hypothesis* [Van Rijsbergen 79], that if a term is a good discriminator of relevance and non-relevance then a closely associated term should also be a good discriminator. This leads to the supposition that the terms in the database can be organized into clusters of terms that are related because they are used together frequently in the collection of documents. This information can then be used to expand queries by adding terms that are closely related to the terms that the user provides. The clusters can be organized prior to search for the entire collection using an efficient nearest neighbor algorithm [Croft 84, Croft 86]. The measure of two terms "closeness" is based on the frequency of their co-occurrence in the collection using Dice's coefficient.

A similar hypothesis, called the *Cluster Hypothesis* [Van Rijsbergen 79] can be made about the documents. It states that closely related documents tend to be relevant to the same queries. Therefore, it would seem worthwhile to determine what documents are closely related in content. This can be accomplished with the same algorithms used to cluster terms. There are a variety of different ways to organize clusters of documents [Salton 83, VanRijsbergen 79]. One way is to compute a *centroid* of a group of documents, which is an "average" of the documents that compose it. Then a centroid is computed for a cluster of centroids, and so forth leading to a hierarchic clustering of the entire collection. Search in this organization starts by examining the top level clusters, and choosing the one that is most similar to the query. Then this cluster is examined in the same way, and so forth until there are a reasonable number of documents, usually around 20, left to retrieve.

Another method is called the *single-link* method. In this method only the the lowest level clusters are formed. The collection is searched by computing similarities with documents as is done in a non-cluster search, but instead of retrieving a single document, the document found and all other documents connected to it in its cluster are retrieved also. The links that are used to form these clusters are nearest neighbor links.

Clustering also can lead to more efficient searching, since only the cluster representatives need to be examined to decide what documents to retrieve rather that all the documents. There are many other considerations with regard to clustering, but they are beyond the scope of this work. The important thing about cluster searches is that they tend to retrieve different documents than non-cluster searches. The reason for this is shown in figure 2.1, where the query may share a great number of terms with the document, but share none of the terms that define the nearest relationship with another document. The nearest neighbor would be retrieved along with the document.

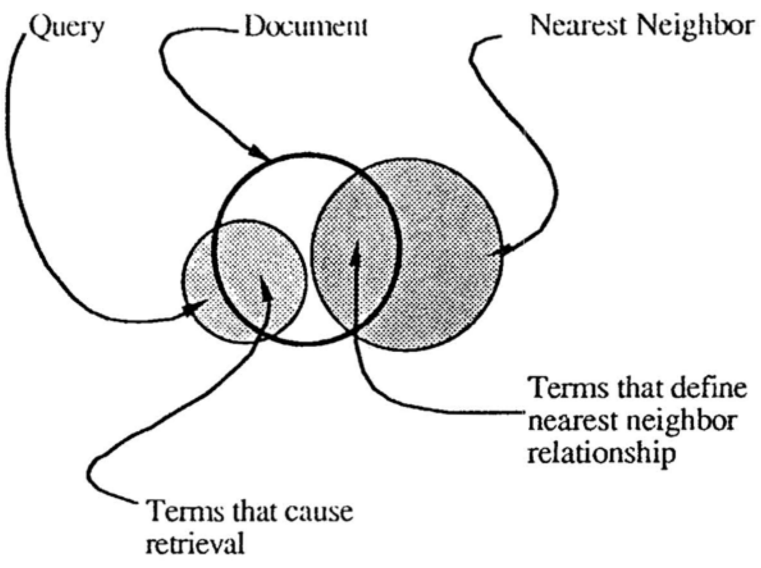
Query
Document
Nearest Neighbor
Terms that define nearest neighbor relationship
Terms that cause retrieval

Figure 2.1: Showing how a cluster search can retrieve different documents.

2.2.2.3.2.4 Determining nearest neighbors

Determining nearest neighbors for both documents and terms is done using the same algorithm. In this discussion, documents will be used to explain the method. The most direct way to find nearest neighbors is to compute the similarity that a document D has with all other documents; this requires N (N-1) / 2 calculations. There are a number of optimizations, however, that can be made to significantly speed up the process . First, the assumption is made that only the closest nearest neighbor is needed (more if there is a tie). The maximum number saved is 5. Second, the documents must share at least one term. This means that a large number of documents will not be checked with a particular document since they have no terms in common. Third, a minimum similarity is placed as a threshold; this implies that documents must be more than weakly related to qualify; this value is 0.30 (This similarity is computed using Dice's coefficient).

Fourth, an upper bound (U_1) [Smeaton 81] can be calculated that will terminate the examination of any more of a document's term list when the remaining documents cannot have a larger similarity than the current largest similarity calculated so far. For example, a document D has 20 terms and 7 terms have been processed so far. For any document not already seen, the largest similarity possible is if it contains the remaining 13 terms and only those terms. If the similarity is smaller than the smallest on the list of similarities calculated so far, processing for the current document is stopped. This bound avoids the calculation of similarities involving very common terms. It requires that the term lists of the documents are sorted in order of increasing frequency.

Another upper bound (U_2) [Murtagh 82] can be calculated while looking at documents not already seen. It is a variation of U_1 applied to the calculation of a single document. It is interpreted to say that a document,D′, not already seen, can have either the remaining terms on the current documents term list or fewer terms, whichever is smaller. If this similarity is smaller than the smallest on the list of similarities calculated so far, no similarity is calculated. This bound avoids the retrieval of many term lists, which is the most expensive part of determining nearest neighbors.

Fifth, since the documents are processed in increasing order of their unique identifier, save the highest value found so far for a document. For example, if in processing document 50 the largest similarity was with document 234, save that value with document 234 also. When processing comes to document 234, any similarity has to be greater than or equal to this value. And no document with a number less than 234 need be examined.

The algorithm for determining nearest neighbors is the following

```
1.  Get the next document, D.
2.  Get the term list for that document.
3.  Sort the list in order of increasing frequency, if not already done.
4.  Calculate the bound $U_1$.
5.  If  $U_1$ < the current neighbor similarity on the list or
        $U_1 < 0.30$ (the minimum similarity)
    Goto 1.
6.  Get the next term from D's term list.
```

```
7.  Get the list of documents for that term.

8.  Get the next document, D′, from the list where the D′ > D.

9.  Calculate the bound $U_2$

10. If  $U_2$ < the current neighbor similarity on the list OR
        $U_2 < 0.30$ (the minimum similarity)
    Goto 8.

11. Calculate similarity = Dice(D, D′)

12. If similarity > previous largest similarity, for the pair (D, D′) replace it.

13. If similarity >= current neighbor similarity the replace it if > or add it if equal

14. Goto 8.
```

2.2.2.3.3 Network Representation

By combining the association and the cluster hypothesis and performing the nearest neighbor calculations on both the terms and the documents, a network representation of the document collection can be formed. This representation supports both cluster searches and normal search techniques efficiently [Croft 83, Croft 85]. The bounds U_1 and U_2 used in the calculation of nearest neighbors can also be used in the searches to make them more efficient. The only alteration is in the size of the number of documents saved, which determines the value of these bounds. In the nearest neighbor calculations, only the most similar document or term was sought, so the bounds were relatively high. In the normal search environment, this restriction can be adjusted to find a reasonable number of documents, usually about 20, for retrieval.

2.2.2.3.4 Relevance Feedback

After a search has been evaluated, the information provided by the user can be used to adjust the query so that it more accurately reflects his interest. This process can be applied to both the vector and probabilistic searches. In vector space model searches, terms that are in relevant documents are given additional positive weights and terms that are in non-relevant documents are given negative weights. Of course, some terms may be in both categories in which case the weights will tend to cancel each other out. In probabilistic model searches, the occurrence of a term in relevant or non-relevant documents is already factored into the weight, so all that needs to be done is to fill in the values.

Relevance feedback can also provide additional terms for the query. A simple approach is to add all of the terms to the query from the documents judged relevant. This approach in systems that use automatic indexing tends to expand the query too much, adding many terms that are of little use. A more sophisticated approach is to have the user select terms from the relevant documents that are particularly interesting and add only those terms to the query. In this way, terms such as `interesting` from phrases such as "`An interesting approach is...`" are not added to the query.

2.2.2.4 User Interface

The user interface aspect has largely been ignored in the design of traditional retrieval systems. Most commercial interfaces are designed to work with hardcopy terminals, forcing the style of interaction, generally, to a command line orientation. Online help facilities are limited to command explanation and, if available, simple alphabetical listings of terms. To get help in selecting the proper terms to use, a user must generally refer to a printed thesaurus. More sophisticated interfaces have been developed, some based on the concept of menu selection, and others based on the use of a high resolution bit-mapped screen. A number of these kinds of interfaces will be discussed in relation to specific systems in section 2.5.

2.2.3 Evaluation

There are a variety of factors that can be evaluated with regard to information retrieval systems. Cleverdon [1966] identified the following six significant factors that can be measured.

1. **Coverage** — the extent to which all relevant material is included in the system.
2. **Time** — how long it takes from when the query is submitted to when the system responds.
3. **Presentation** — the form in which the system's output is displayed.
4. **Effort** — the labor on the user's part, either mental or physical, to use the system.
5. **Recall** (R) — the proportion of the material relevant to the request that was retrieved.
6. **Precision** (P) — the proportion of the material retrieved that is relevant to the user's request.

In the usual discussion of IR system evaluation [Salton 83, Van Rijsbergen 79], items 1–4 are usually passed over as being easy to measure, and the emphasis is placed on examining the last two items and measures related to them. Coverage is simply a matter of the content of the documents in the database in relation to the subject of a query. Time is often a matter of the kind of hardware, the system load, and the efficiency of the file organization or database system, as well as other factors.

Items 3 and 4 are related since the form of the presentation can cause the user to expend more or less effort depending on its effectiveness at displaying and capturing information. Suffice it to say that effective presentation of information will reduce the amount of effort that the user must expend while interacting with the system. The most effective system would have multiple ways of displaying information so that the user can use the form that he prefers. For example, many people find point-and-click menu selection interfaces very easy to use, while those that are accomplished typists find that using a mouse or trackball is a hindrance and prefer control key combinations to perform the same tasks. Novice users often find menu-based systems easy to use, since they provide a structure that helps them avoid making mistakes and learn the function of the system. But as they become proficient with the system and know where information of interest is in the menu structure, the menus become an annoyance, especially if they have to go several layers deep to get to a particular selection.

Some systems have been evaluated with the idea of the user's effort in mind. One way this is done is to count the number of tokens that the user must enter. An example [Oddy 74] of this type of comparison is between THOMAS and MEDUSA, a medical information retrieval system. The tokens for MEDUSA are command names, concepts, system-assigned codes, and logical connectives The tokens for THOMAS are concepts, special words (YES, NO, and NOT), numbers from the user displays, and null messages (no comment). The average effort, over 32 queries, to use THOMAS required the user to enter 9.5 tokens; MEDUSA required 33.25 tokens. Both systems had approximately the same level of effectiveness. This kind of comparison gives a rough idea how easy one system or the other is to use.

Factors 5 and 6, are the two primary measures of IR system effectiveness. These measures as well as others are computed from the "contingency" table in figure 2.2.

	Relevant	Not Relevant	
Retrieved	r	n-r	n
Not Retrieved	R-r	N-n-R+r	N-n
	R	N-R	N

Figure 2.2: "Contingency" table for computing evaluation measures.

Precision is defined to be r / n and recall is r / R. Recall is calculated in an experimental situation when standard test collections that have known relevance judgements for the test