Interactive Query Formulation and Feedback Experiments in Information Retrieval

Jose E. Araya
Ph.D Thesis

TR 90-1115
April 1990

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

INTERACTIVE QUERY FORMULATION AND FEEDBACK EXPERIMENTS IN INFORMATION RETRIEVAL

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Jose Enrique Araya

May 1990

INTERACTIVE QUERY FORMULATION AND FEEDBACK EXPERIMENTS IN INFORMATION RETRIEVAL

Jose Enrique Araya, Ph.D.
Cornell University 1990

The effective use of information retrieval systems by end-users has been limited by their lack of knowledge on the particular organization of the databases searched and by their limited experience on how to formulate and modify search statements. This thesis explores and evaluates two mechanisms to improve retrieval performance by end-users.

The first mechanism complements the formulation of a query by allowing users to interactively add term phrases. These phrases are generated either from the query text or from known relevant documents. This addition of term phrases to a query is suggested by the term discrimination model as a precision enhancement device. An interactive front-end for the SMART information retrieval system was developed to perform the interactive experiments needed to evaluate different phrase addition strategies.

The second aspect of retrieval improvement studied is the evaluation of two database organizations that can be used to obtain new relevant documents by looking in the neighborhood of known relevant documents, *browsing*.

Browsing in cluster hierarchies and nearest-neighbor networks is compared to relevance feedback in non-interactive experiments.

The results obtained for the phrase addition methodology showed that simple non-interactive addition of phrases can perform as well as interactive addition. Even an optimal selection of the phrases based on the relevant documents not yet retrieved, did not significantly improve performance over simply adding all the phrases generated. Many useful phrases are not selected by users because they look like random association of terms. The usefulness of these phrases comes from the fact that either they are pieces of larger (semantically meaningful) phrases, or they are made up of local synonyms specific to the document collection used.

The browsing experiments in cluster hierarchies and nearest-neighbor networks showed that the second organization consistently performs better than relevance feedback in different collections. Cluster browsing is more dependent on the characteristics of the collections; but when the circumstances are favorable, cluster browsing can produce larger improvements on retrieval than network browsing. Retrieval in both structures is much faster than relevance feedback since only a small portion of the database needs to be inspected.

Biographical Sketch

José Enrique Araya-Monge was born July 6, 1958 in San José, Costa Rica. He is the ninth child of a family of ten. In February, 1980 he got his B.Sc. in Computer Science from the Universidad de Costa Rica; a year later he got his Licenciado degree from the same university. He started his studies at Cornell in the Fall of 1983, got a M.Sc. from Cornell in August of 1986. Finally, in May of 1990 he got his Ph.D. in Computer Science from Cornell University.

To my family and friends

Acknowledgements

First of all, I want to express my sincere gratitude to my thesis advisor, Prof. Gerald Salton, for all the years I have the privilege to work with him. I also want to thank Prof. R. Constable and Prof. N. Prabhu for kindly serving in my Special Committee.

I am very pleased of having worked with the other members of the SMART group, Maria Smith, Chris Buckley and Joel Fagan. I will also remember with pleasure my conversations with Prof. Michael Wong about life, information retrieval and ice skating.

I will always have fond memories of my years as a member of the Dewitt Place Coop. There, I was forced to learn how to cook and fortunately nobody was ever poisoned. Many thanks to Alan Denton, Mark Gomez, Julia Kim, John Johnstone and Scott Smith.

During my years at Cornell, many good friends have come and left before me. Among them, I want to mention Roxana Salazar and Eduardo Scorza, for sharing with me our first years at Cornell. Also special is Gonzalo Castañeda for our Fridays of movies, discussion and music.

I was lucky enough to find many good new friends at the end of my road in Cornell. Among them, I am specially in debt with Maro Villalobos and

Juvenal Fernández for their kind help.

Finally, I want to thank Cornell University and the Department of Computer Science for the opportunity they gave me further my studies. Their support facilities made my work easier.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The task of an information retrieval system is to answer a user request for information by providing a set of documents relevant to that request. The system is expected to fulfill this task as efficiently as possible. At the same time the system must be effective at distinguishing which documents are relevant to the request from those which are not. End-users of information retrieval systems find it difficult to make effective use of those systems. Even though they may be experts in the area covered by the given collection, two factors limit their ability to elaborate effective search requests. First, few of them have good experience in transforming their information need into a specific query suitable to be submitted to a computer. Usually, queries take the form of Boolean expressions whose mastering requires an effort that many users are not willing to put. The other factor limiting users is lack of knowledge about the specific vocabularies used to describe the documents in the different collections. The document indexing and the presence of many competing terminologies may produce many alternative ways to describe a specific topic. To ask a user to come up, unassisted, with all the different alternatives is clearly unacceptable.

These problems are partially solved by the intervention of a search intermediary who is an expert in both boolean query formulation, and in the content of the collection of documents. The task of this intermediary then consists in interacting with both the user and the retrieval system so that even though the user is isolated from the specific details of the system a satisfactory answer to the request can still be found. The communication between the user and the intermediary is the crucial step in this arrangement. The intermediary needs to understand what the user is looking for in order to make effective use of the retrieval system. When the user has a clear idea of what he wants and can convey this idea to the intermediary, this arrangement works well. In many instances, however, the user only has a fuzzy idea about what he is looking for, making the task of the intermediary much harder than in the previous case.

The need to involve users in the search process has motivated the emergence of sophisticated interfaces for information retrieval systems. Users must actively participate because in many cases, see [Odd77], even though some users would have a hard time trying to clearly specify their requests, they can easily recognize what they are looking for. Furthermore, the search intermediary may misunderstand the user request and waste effort trying to solve a slightly different problem.

If an effective participation of the users is wanted, an information retrieval system must help users overcome the two limitations discussed before. Some systems guide users during the query formulation process by providing mechanisms to build normalized queries. One example of such a system is OAK [MCBC89]. This system guides users in the formulation of boolean queries in disjunctive normal form. Many systems also compensate the user lack of familiarity with the vocabulary of the collection by providing tools to inspect the set of terms used. Along with terms, information such as document frequency is also provided to help users to choose among the terms. Besides these two facilities, some systems act as gateways or front-ends between the users and many different collections, CONIT [MR81], IIDA [MHA82]. In this case, the system must be able to present a consistent set of operations across the different retrieval systems so that the users are not bothered with many different ways to do the same thing.

The query formulation mechanisms mentioned in the previous paragraph usually involves some form of query enhancement. Many of these query enhancement techniques have been described in the literature [Sal86]. These techniques are presented in more detail in section 1.3; for the moment it is important to know that these enhancements improve the performance of a query by either expanding or contracting the scope query terms. This query modification process may be either fully automatic or user-controlled.

Search performance can be improved not only by changing the scope of the query terms, but by taking into account known relevant documents. The second alternative may take many forms, among which relevance feedback is the most studied, [RJ71], [Ide71], [IS71] and [SB88a]. In its general form, relevance feedback changes the query by adding terms from retrieved relevant documents and subtracting terms from retrieved non relevant documents; the intention of the changes is to make the query more similar to the relevant documents and less similar to the non relevant documents. The feedback modification is usually done automatically but as evaluated in this thesis, it is possible to allow users to choose which feedback terms will modify the query.

Browsing is an alternative to relevance feedback. In this case, documents closely associated to retrieved relevant documents are also retrieved and presented to the user. Structures such as cluster hierarchies and database networks can be used for browsing. Later in the thesis these structures will be analyzed in more detail.

1.1 Overview

The general goal of this thesis is to explore and evaluate mechanisms for the improvement of retrieval performance. More specifically, two objectives are pursued. In the first objective, we want to evaluate different strategies for interactive modification of queries; while for some strategies the users complement their initial queries by adding phrases selected from a set of phrases generated from the queries; for other strategies the users control the relevance feedback modification process by selecting some phrases extracted from relevant documents. The second objective is the evaluation of different structures used to obtain new relevant documents from known relevant ones. In this case, the database organization is used to expand the search but the query is not modified in any way.

Several interactive experiments were performed to evaluate different query modification strategies. In these experiments, users modify some standard queries by choosing some term phrases to be added to the query. Section 2.4 gives a detailed description of these experiments. In addition to the interactive experiments just mentioned,

The remainder of this chapter presents the general background needed to understand the experiments done. First, section 1.2 reviews the vector space model. A discussion of the different query enhancement devices proposed in the literature is done is section 1.3. Section 1.3.4 reviews query enhancement using relevance feedback. The concept of retrieving documents through browsing is discussed in section 1.4. And finally, section 1.5 gives an overview of interactive systems and the different facilities provided by them to help users elaborate their searches.

Chapter two presents the retrieval environment used in the interactive query modification experiments. There, a description is done on the way the phrases used to modified the queries were generated and selected for presentation to the user. The user interface employed in the experiments is also presented.

In Chapter three the results of the interactive experiments are analyzed. Besides the results obtained by the subjects of the experiments, many automatic strategies are also presented and compared against the interactive strategies used by the subjects.

Chapter four deals with browsing around a cluster hierarchy after retrieving some relevant documents. Document clustering is introduced and an algorithm that simulates browsing in a cluster hierarchy is presented. Experiments using this algorithm are evaluated and presented.

Chapter five also deals with browsing but in this case the underlying structure is a network of terms and documents. Experiments on how to use this network to look around relevant documents in search of new relevant documents are performed, evaluated and presented.

Finally in chapter six, a recapitulation and discussion of the results obtained is done.

1.2 Vector Space Model

This thesis uses the vector space model (VSM) for the representation of documents and queries. In the VSM the documents and queries are represented by vectors of descriptors. If n is the number of descriptors in the collection then document D_i is represented in the following way:

$$D_i = < d_{i1}, d_{i2}, \cdots, d_{in} >$$

where d_{ij} is the weight or importance of descriptor T_j in document D_i. Queries are represented also by vectors:

$$Q = < q_1, q_2, \cdots, q_n >$$

A detailed presentation of the VSM can be found [SWY75].

The VSM was developed to solve several problems of the Boolean model which is the principal model used in commercial applications. Salton [Sal75a, pp. 121-123] lists some of the problems with the Boolean model:

1. It is hard to control the amount of material retrieved. It has to be done by manipulating the formulation of the query which requires knowledge of the vocabulary.

2. There is no partial matching. The search involves a series of "all or nothing" decisions without a continuous spectrum of similarity between the documents and queries.

3. No ranking of the output by decreasing similarity.

Weights can be added to the boolean model so that a ranking can now be obtained. This however generates problem in the interpretation of the results, see Bookstein [Boo78]. In an interactive environment these problems of the boolean system have the effect of discouraging end-users. On one side they have to express their information need in the highly structured form of a boolean query when it would have been much easier for them to write a paragraph describing what they need. On the other hand, their limited knowledge of the specific vocabulary used makes it very hard for them to control the size of the material retrieved. According to Hildreth [Hil83] boolean logic appears to be one of the most difficult aspects of information retrieval for end-users. The strict meaning of the operators does not correspond to their every day use which causes confusions like using AND instead of OR.

The vector space model has several important advantages:

1. No need for boolean expressions. The vector query can be automatically constructed from a natural language statement of the information need of the user.

2. Terms can be easily weighted to reflect their relative importance.

3. The similarity function allows the ranking of the output, controlling its size and retrieving documents that only partially match the query.

The vector space model has some disadvantages:

1. Some parameters of the model like the similarity function are not explicitly derivable from the model but instead are chosen a priori by the system designer.

2. Term independence is assumed even though it does not occur in practice.

3. Term relationships are not expressible within the model.

1.2.1 Vector Similarity and Term Weights

Any retrieval process requires a mechanism to determine how well a given document matches a query. For the vector representation there are many functions that can be used to measure the degree of similarity between two vectors. Salton and McGill [SM83] present many of the functions that have been used in the literature. In this study the cosine similarity measure will be used. This function is generally regarded as a good similarity measure. It is defined by the formula:

$$\text{Sim}(D_i, D_j) = \frac{\sum_{k=1}^{k=n} d_{ik} d_{jk}}{\sqrt{\sum_{k=1}^{k=n} d_{ik}^2 \sum_{k=1}^{k=n} d_{jk}^2}}.$$

The cosine similarity is an inverse function of the angle between the pair of vectors so that the larger the similarity the smaller the angle between the vectors and the closer they are to point to the same direction.

The assignment of term weights can be done following many different methodologies. [SM83] presents several of them. Most of the weighting schemes are based on the observation that there is some relation between the frequency a term occurs and the importance of that term as a descriptor of the document. This leads to the *term frequency* weighting where the term frequencies are used as the term weights. Usually the terms are normalized using the maximum term frequency of the document vector so that the term weights are in the interval [0,1]. The following formula shows the normalized term frequency weight for a term **t** in a document **d**:

$$norm\_tf_t = \frac{\text{frequency of t in d}}{\text{maximum term frequency in d}}$$

Term frequency weights give too much importance to very frequent terms. On the other hand, the inverse document frequency weighting (*idf*), [SJ72], [SY73], stresses those terms that do not appear in a large proportion of the documents in the collection. That is, the more documents a term appears, the less useful that term is to identify a particular document. In this thesis the idf is used as a factor modifying the *norm_tf* weights presented above. The following formula shows the inverse document frequency weighting:

$$tf\_idf_t = norm\_tf_t \cdot \log \frac{\text{number of documents in collection}}{\text{document frequency of t}}$$

Finally in order to compensate for the differences in weights caused by different vector lengths the *tf_idf* weights are normalized using the cosine normalization as it is shown in the next formula:

$$w_t = \frac{tf\_idf_t}{\sqrt{\sum_{i=1}^{i=k} tf\_idf_i^2}}$$

where k is the length of document vector $\mathbf{d}$.

1.3 Query Enhancement Mechanisms

The elaboration of effective queries is in general a hard task. To do this users must select terms that can distinguish between relevant documents and non relevant ones. Because of the inherently subjectivity of relevancy it is very hard to come up with a set of terms that will perfectly separate relevant and non relevant documents. For this reason some models, like the term discrimination model, only try to measure the ability of terms to distinguish documents between each other.

In this section several mechanisms for query enhancement are reviewed. All these mechanisms modify the query by replacing some terms by other terms considered better and by introducing new terms into the query. Before describing the enhancement mechanisms in more detail, it is convenient to briefly present the two measures normally used to evaluate the performance of a retrieval methodology. The first measure, *precision*, gives the proportion of retrieved documents that are relevant. The second measure, *recall*, gives the proportion of relevant documents that are retrieved. Some of these enhancements to be presented improve the precision of the query, while other enhancements improve the recall of the query.

1.3.1 Term Discrimination

Originally the term discrimination model was presented as a technique for assigning weights to the terms [SY73]. It is used here to justify the query modification strategies that will be tested interactively. A general review of the model can be found in [SM83], [Sal75a] and [Sal75b].

In the term discrimination model, a term is considered good if its removal renders the documents of the collection more similar to each other. That is, a good term is one able to increase the separation between documents. On the other hand, a term is considered bad if its removal makes the documents less similar, since in that case the term is reducing the separation between documents.

The effect of the removal of a term is measured by the change in the space density which is the mean similarity between documents in the collection. In other words, for all pairs of documents, D_i and D_j, their similarities $\mathrm{Sim}(D_i, D_j)$ are computed, and their average give the space density. The *discrimination value* of a term k is then defined as:

$$\mathrm{DV}_k = \mathrm{density(VS)} - \mathrm{density}(\mathrm{VS}_k).$$

Where VS is the original vector space and VS_k is the vector space after the term k is removed.

One very important relation was found between the frequency of a term and its discrimination value [SY73]. Bad discriminators are general terms with a high document frequency. These terms occur in a large proportion of the documents of the collection and are not useful at distinguishing one document from another. Good discriminators, terms with positive discrimination values, were found to be terms with moderate document frequency: neither too frequent nor too rare. Finally, discrimination values closer to zero were found associated with terms having very low document frequency. Those terms are so specific that they occur in only a handful of documents so they have little impact on the space density. One problem with these unfrequent terms is that they are so rare that it will be very hard for the users to come up with them during the query formulation process. One initial query modification technique removes bad and indifferent discriminators. The elimination of the former, however, will degrade recall in a high-recall search since in that case general terms are most likely to be useful. On the other hand the elimination of very specific terms may cause a loss in the description of certain documents with the corresponding deterioration in effectiveness.

These observations have motivated different strategies that enhance the initial formulation of a query by taking terms with negative or indifferent discrimination values and respectively turning them into more specific or more general terms. As it will be presented in the next section, better recall can be obtained by collecting low frequency terms (indifferent discriminators) into classes which will have a higher frequency and a greater discrimination value. Correspondingly, precision can be improved by transforming high frequency terms (bad discriminators) into phrases. The next two sections describe in more detail different mechanisms for query enhancement.

1.3.2 Recall Enhancement

Stemming

The most basic recall enhancement device is term stemming. In this technique, terms are stripped of their suffixes so that many related forms are reduced to the same stem. The basic assumption of this transformation is that words with the same stem are close in meaning. While this assumption may be true most of the time, there are situations where words have the the same stem but very different meaning; take for example *NEUTRON* and *NEUTRALIZE.* Because the stems represent several terms combined together their document frequency is higher than the original terms making it more likely to increase recall since they match more documents.

Many stemming algorithms have been presented in the literature. Most of them use suffix dictionaries to reduce the terms. Stemming algorithms like [Lov68] and [Por80] aim at *conflation*; that is, after some suffixes have been removed some stem endings are changed to create a single proper form. [Sal68] on the other hand, aims at *truncation* so that no endings are added after the removal of the suffixes. Table 1.1 illustrates the difference between conflation and truncation.

Some of the stemming algorithms mentioned before ([Sal68], [Lov68] and [Por80]) are not specialized on the suffixes they can recognize and remove. Most of the recent stemming algorithms, however, are oriented towards specific subject areas. This is particularly clear in the area of medicine literature where considerable research has been done to handle medical suffixes; see [UD83] and [PD79].

Terms	Truncation	Conflation
conflation	confla	conflate
relational	rela	relate
absorption	absorpt	absorb
absorbing	absorb	absorb

Figure 1.1: Examples of conflation and truncation

Thesaurus

Another way to enhance the recall performance of a query is by using a thesaurus. In a thesaurus the terms used in a particular subject area are organized into many categories or term classes. Each category consists of several terms considered to be associated with each other. The most common association between terms is by the synonym relation where all the terms in a category refer more or less to the same concept. Usually the thesaurus also provides hierarchical relations between classes. *Narrower* and *broader* relations link term classes with other classes dealing with a more specific or more general concept. Rada [RM87] considers in general that thesauri are poor in other non-hierarchical relations.

The term class associations of a thesaurus can be used to improve the recall of a query. Indifferent discriminators, terms with very low frequency, can be replaced by the thesaurus classes containing them. One alternative is to have the classes added to the query without deleting the original terms [SM83]. As classes have greater frequency than the terms they contain, their

- class 2718
  - dyspnea
  - shortness of breath
  - breathlessness
- class 3197
  - lever enlargement
  - hepatic enlargement
  - hepatomelagy

Figure 1.2: Two thesaurus classes with their terms

discrimination value should be better. If the classes containing the terms are still too specific then the *broader* relation of the thesaurus can be used as necessary to expand the class into a larger set. Figure 1.3 presents an example of the use of the thesaurus classes of figure 1.2 to broaden a query [1]. Assume that *dyspnea* is a rare term. Even though it is very specific and it can be very useful for precision many documents may be missing if alternative terms are used. As the figures show, recall can be improved by replacing *dyspnea* by its class identifier so that now the query will also retrieve documents containing *shortness of breath* or *breathlessness*. Another way to modify the query is by

[1]This example was prepared from a figure in [RM87].

Original query:	< dyspnea, hepatomelagy >
Replace terms:	< 2718, 3197 >
Add classes:	< dyspnea, hepatomelagy, 2718, 3197 >

Figure 1.3: Recall enhancement by thesaurus classes

adding the class identifier to the query without removing the original term. Figure 1.3 also shows a similar expansion process done for *hepatomelagy.*

[SM83, pages 77–78] has pointed out that when a term is replaced by its class, the components of that class should have frequencies that are not excessively disimilar. Otherwise the elements with the larger frequencies will have a strong influence in the class.

Thesauri have several disadvantages. First of all their manual construction is very hard and expensive. They require the participation of experts in the subject area. Even small thesaurus involve a lot of work. Automatic thesaurus construction has been investigated, [Sal71a], [SJ71] as an alternative way to provide term associations for query enhancement. Whereas the manual thesaurus are built taking into account the semantic relation between terms, automatic thesaurus are constructed by statistical analysis of the cooccurrence patterns of the terms.

The second disadvantage of thesauri is the need to update them as the collections grow and new terms are introduced into the subject area. For automatic thesaurus creation from scratch is expensive but continuous use of the old thesaurus can lead to loss in performance, [JW72]. For manual thesaurus, the inclusion of new terms will again involve the participation of experts on the field which is rather expensive.

1.3.3 Precision Enhancement

In order to increase the precision of a given query, those terms with negative discrimination value must be made more specific. Since these bad discriminators are characterized by their high document frequency the reduction of that frequency will improve the discrimination power of the terms. One way to make a general term more specific is by using the *narrower* relation available in a thesaurus so that a term that happens to be too general is replaced by a thesaurus class with a narrower scope.

Another way to handle general terms is to transform them into phrases. That process not only reduces the occurrence frequency of the terms but also introduces a more specific concept. As an example, the terms *information* and *retrieval* used in a query may produce false drops since they will match a statement like:

> ... the second chapter gives *information* on the *retrieval* of underwater samples.

Clearly the problem is caused the lack of association between the terms in the query formulation. By replacing both terms by the phrase *information retrieval* a strong association between both terms is introduced. It makes the query more specific and reduces the number of false drops. From the point of view of the discrimination value the general term *information* is replaced by the phrase *information retrieval* which is more specific and has better discrimination value. [SM83, pages 85–86] recommends that one of the phrase components must be a very frequent term whose discrimination value needs to be increased, while the other components should not be too rare, otherwise the frequency of the phrase will be too low.

Several methodologies have been developed to generate phrases as content descriptors. Basically, two approaches are available: statistical generation and syntactical generation. In the statistical approach, the phrases are generated by considering the statistical associations between the terms, like their individual frequencies, the frequency the terms cooccur in different documents, the context and their relative positions. In the syntactical approach, the document sentences are analyzed and phrases are generated according to the syntactic role of each term and their relationship with other terms.

The early work in statistical phrase generation is characterized by the use of word cooccurrence frequencies to determine the presence of a phrase, [Doy62], and [GJ63]. In [Sal68], a dictionary containing a standard set of phrases was available. A phrase in this dictionary is detected when all its components occurred a specific number of times in a given context (document or sentence). The construction of the dictionary is one disadvantage of this alternative since it will have to be done either manually or by extracting the phrases from other sources like thesauri. Furthermore if the subject area is dynamic the introduction of new terminology requires updates in the dictionary. In [SYY75] a phrase generation procedure is presented which does generate phrases from the queries based on the term frequencies and the distance, number of non stopwords, between the components of the phrase. The procedure takes the query terms then removes the stopwords and stems the remaining terms. Once this is done, pairs of terms are tested to see whether or not their distance is smaller than a threshold. The procedure also requires one of the terms to have high frequency. [Fag87] extended this procedure and included several parameters to control the conditions that a phrase must fulfill before being accepted. Section 2.1.2 presents Fagan's procedure in more detail.

Statistical phrases are not considered totally appropriate to represent the content of a document. In figure 1.4 two cases, taken from [Fag87], are shown where the proximity criterion is not enough to avoid associating two document terms that should not be put together in a phrase. The need of a better representation of the document content than the representation provided by the statistical associations has lead many authors to work on content analysis using some measure of syntax. A syntactical analysis can determine that the document terms in figure 1.4 should not form a phrase.

The simplest syntactical technique consists in attaching syntactic tags like noun, verb or preposition to the non stopwords of the sentences. After the tags have been assigned, some sequences conforming potential phrases can be recognized and the phrases can then be generated. One possibility of a tag sequence leading to a phrase is a noun phrase. For specific uses of this technique see [Kli73] and [DG83]. More complex systems like [Hil73] and [Vla83] do a partial syntactic analysis while others, [Sal66], [MNRH84] perform a full syntactic analysis of the text.

query:	...operating system ...
phrase:	oper system
document:	...a fully automatic document retrieval *system operating* ...
query:	performance evaluation and modeling of computer systems
phrase:	comput system
document:	...of this type of *system* for *computer* programming ...

Figure 1.4: Examples of statistically generated inappropriate phrases

1.3.4 Relevance Feedback

After a query has been run, users determine which of the retrieved documents are relevant to their request, and which are not relevant. This classification of the retrieved documents into relevant and not relevant is called *relevance assessments*. In the relevance feedback methodology, the query is modified by taking into account the relevance assessments given by the users on documents retrieved in a previous run. That is, the query is adjusted to reflect the choices of relevant documents made by the users.

Relevance feedback is an example of what Lesk and Salton [LS71] called a *postsearch* query formulation; in contrast, the formulations previously seen like the recall and precision enhancement techniques are cases of *presearch* query formulations. Whil the *presearch* formulations adjust the query before running it by reviewing the vocabulary available, the *postsearch* formulations modify the query using the information obtained after running the query.

Relevance feedback has been shown to be a very effective technique. It was introduced in the mid 1960's as a way to automatically expand vector queries [RJ71], [Ide71] and [IS71]. Because the expansion is automatic, relevance feedback can be very helpful for users with limited knowledge of the vocabulary used. The query expansion is done by adding to the vector query those terms appearing in relevant documents retrieved, and subtracting from the vector query those terms appearing in non-relevant documents retrieved. After this transformation, the vector query will be more similar to the relevant documents retrieved and less similar the non relevant ones.

The dominant popularity of boolean queries in the operation environments has stimulated studies on applying relevance feedback to boolean queries; on this regard, see [DD80], [SVF84], and [SFV85]. A different methodology for feedback is given in [WYB88] and [WYSB89] where the users are not asked to give relevance assessments on the documents retrieved but a partial order of document preferences. This partial order is then used in an iterative process that modifies the query until the query-document similarities correspond to the partial order. In other words if d_1 is preferred to d_2 then the iterative process modifies query vector q until the new vector query q' satisfies $Sim(q', d_1) > Sim(q', d_2)$. Wong [WYSB89] found this approach comparable to the best relevance feedback strategies studied in [SB88a].

The feedback experiments performed for this thesis follow the original relevance feedback approach. In its general form the relevance feedback modifies the query using formula 1.1.

$$Q_{\text{new}} = Q_{\text{old}} + \alpha \cdot \sum_{R_i \epsilon RelRet} \frac{R_i}{|R_i|} + \beta \cdot \sum_{N_j \epsilon NRelRet} \frac{N_i}{|N_i|}. \qquad (1.1)$$

Where *RelRet* is the set of relevant documents retrieved and *NRelRet* is the set of non relevant documents retrieved. Formula 1.1 is a restriction of formula 1.2 to the set of retrieved documents.

$$Q_{\text{new}} = Q_{\text{old}} + \alpha \cdot \sum_{R_i \epsilon Rel} \frac{R_i}{|R_i|} + \beta \cdot \sum_{N_j \epsilon NRel} \frac{N_i}{|N_i|}. \qquad (1.2)$$

Where *Rel* is the set of relevant documents and *NRel* is the set of non relevant documents. Rocchio [RJ71] showed that for $\alpha = 1/|Rel|$ and $\beta = 1/|NRel|$ formula 1.2 gives the best ranking for the relevant documents.

Several parameters must be set in order to use formula 1.1. First, the values of α and β usually take the form $\alpha = \alpha'/|RelRet|$ and $\beta = \beta'/|NRelRet|$. Then it has to be decided which of the relevant and non relevant documents are used in the modification of the query. For example, all the documents may be used, or only the relevant documents or all the relevant and the top ranked non relevant.

On important point about relevance feedback is its evaluation. What must be reflected in the evaluation is the ability to bring *new* information to the users; it should not be affected by improvements in the ranking of previously retrieved relevant documents since those changes do not bring anything new to the consideration of the users. Several feedback evaluation methodologies has been studied [CCR71]. For the runs done in this thesis, the *residual collection* evaluation is used. In this methodology the feedback query is run and evaluated in a collection similar to the one used for the original run but which has all the previously retrieved documents deleted.

Some of the interactive experiments performed for this study allow users to refine the feedback process by individually selecting which terms are used to modify the query. These strategies are described in section 2.4.

1.4 Browsing

One of the two goals of this thesis is to evaluate how two different organizations of a collection of documents support non-systematic search, *browsing*. During a browsing process documents or terms are accessed because they are closely related to previously retrieved material. Browsing differs from the query enhancement mechanisms presented before because users are active during the retrieval process itself. In other words they do not limit themselves to formulate a query and let the system run it, but they also intervene to direct the search.

Motro [Mot86] lists several reasons why a non-systematic search of a collection may be necessary in certain circumstances. Although he is referring to non-textual databases, his comments are valid for the information retrieval case:

1. lack of familiarity with the way the collection is organized,
2. lack of familiarity with the content of the collection,
3. lack of familiarity with the search language,
4. vague retrieval target,
5. clear retrieval target but user can not describe it.

Section 1.5 deals with the first three points. It details how users can be helped in those cases. Belkin [BOB82] refers to the fourth point as an *anomalous* *state of knowledge*, since users have to describe what they do not know The fifth point correspond to the situation where users can recognize what they are looking for but can not describe it: recognition vs. recall, [Odd77].

1.4.1 Browsing in General

The need for browsing arises in many different areas and it is now considered a fundamental part of content-oriented access [Bat86]. Browsing is done by users when they are looking around the shelves in a library. Once users have been initially attracted to a specific part of the library where they have found a relevant book, they may check the books in the neighborhood to see whether or not some of those books are also relevant. For this process to work, books with similar topics must be stored together. Unfortunately, in many cases the books are distributed following a hierarchical organization which splits some topics and scatters them across the hierarchy. A similar situation occurs in online catalogs where the subject headings follow a hierarchical structure like the Library of Congress Subject Headings which splits many topics. Massicotte [Mas88] presents a browser that compensates for this problem by agglomerating some subject headings into topics.

Motro [Mot86] presents a browser, BAROQUE, for a relational database. He argues that the relational model is not convenient for browsing because information about a specific entity may be scattered among several relations. BAROQUE superimpose a network view of the database which brings together all information available on different entities. The links of the network are automatically constructed using functional dependencies of the attributes of the different relations.

For hypertext systems [Con87], browsing is the major access way. Halasz [Hal88] calls it navigational access and considers it the defining feature of hypermedia. Some hypertext systems are directed exclusively for browsing, in some cases by naive users. Examples of this are, the EMACS Info, the ZOG/KMS project of Carnegie-Mellon [AMY88], the Hyperties project of Maryland [SM86] or the Superbook [RGL87].

1.4.2 Browsing in Information Retrieval

In the area of information retrieval, Oddy [Odd77] proposes a system, THOMAS, where users perform their searches by just browsing without actually formulating a query. THOMAS uses a network associating documents, terms and authors. At any given point of the search there is a set of selected nodes representing the search terms; the documents connected to those nodes are ranked and the top ranked document is presented to the user who not only determines its relevancy but can also change the set of search terms by adding some of the terms found in the top document. The system then reranks the documents according to the new set of search terms and the cycle is repeated when a new top document is presented to the user. The author claims that this approach performs as well as conventional online retrieval systems and it has the advantage of not forcing the user to formulate a query when he only has a fuzzy idea of what he is looking for.

Another browser in information retrieval is presented by Cove and Walsh [CW88] associate words in a network of nearest neighbors to allow the exploration of the contents of some text files.

Browsing can not be considered a replacement of systematic search; furthermore, even in hypertext some authors are calling for the addition of a query processing facility to complement the navigational nature of hypertext [Hal88]. The browsing approach is clearly not feasible for exhaustive searches; but it can provide the searcher of a mechanism to *feel* the collection to formulate a better search [Odd77,Mot86]. Another problem of browsing is that users may get lost. Not only in terms of where they are but what they were trying to do [Fos88],

In order to be efficient, browsing must be supported by structures added to the collection to link different parts of it. This thesis evaluates the support that two different structures give to browsing. Chapter 4 presents and evaluates browsing in a collection whose documents are organized following a cluster hierarchy; while Chapter 5 presents and evaluates browsing in a collection where the documents are organized in a network of nearest neighbors.

1.5 Interactive Retrieval

Because this thesis deals with methodologies to help end-users formulate and reformulate queries, it is appropriate to put those methodologies in context with respect to all the problems faced by end-users when they perform their own searches.

End-users basically have two problems when they run their queries:

- inexperience in the formulation of search statements, and
- inexperience with the organization of the databases to be searched.

Not surprisingly, Fenichel [Fen81] found that the best performance results are obtained by users with experience in search formulation and knowledge of the particular organization of the collections searched. Any interface designed to help end-users must provide mechanisms to compensate for these two factors.

1.5.1 Search Experience and Front-Ends

The search formulation inexperience can be further subdivided into two aspects: first, users may have little knowledge of the operations available in the retrieval systems they are using; and second, they may lack a coherent search methodology to guide them throughout the retrieval process.

The proliferation of retrieval services exacerbates the problems users have when dealing with the mechanical aspects of a specific retrieval systems. These services are usually command-driven and not user-friendly. Even though they share many common operations the particular syntax of the commands is different enough to make learning of each one a non-trivial effort. This problem is addressed by *front-ends*, which are interface programs that allow users to access many different retrieval services, called databanks by some authors [HL85], using a virtual set of operations. These virtual operations are translated by the front-ends into the specific commands of each retrieval service. By providing these virtual operations, the front-ends relieves users of the peculiarities of each individual system.

There are too many of these front-ends to mention all of them here. Hawkins and Levy [HL86] list many of them. Two front-ends will be discussed here. The first one is the CONIT system of Marcus and Reintjes [MR81]. This system uses a table to translate and interpret the messages generated by the different retrieval services. The translation table can be easily updated to accommodate changes in the any of the retrieval services as well as the removal or introduction of new services. The CONIT system sometimes provide a virtual operation not available in some retrieval systems. Marcus and Reintjes considered virtual operations convenient only when the effort required to provide them is not excessive. Another front-end is the IIDA system [MHA82] which not only has a virtual command language but it also monitors the activity of the users. Whenever a problem is detected the system give some advice to the user. Several problems are monitored by the system, among them: creation of too many empty document sets, combining the document sets in too many different ways (*thrashing*), and combining document sets with little change in the results (*dwelling*).

One area where the basic front-ends do not help users is in establishing a successful search strategy which is defined here as the overall approach used to satisfy the user request. It involves the adjustment of the query in order to control the size of the output, its precision and its recall. In a boolean retrieval environment this can only be done by somebody with good knowledge of the database vocabulary which is seldom the case for end-users.

Specific search strategies have been described in many books and articles; look for example Belkin [BV85], and Bates [Bat79b,Bat79a]. Belkin lists the most frequently used search strategies:

- building blocks: run several subsearches and combine their results,
- citation pearl growing: build up the result from a known relevant document,
- successive fractions: starting with a broad search, refine the result,

- most specific facet first: start with the most specific concept,

- lowest postings first: begin with the concept with fewest postings.

Some end-users intuitively follow some of these strategies. Most of them, however, will be greatly helped if the interface can suggest them some course of action. Section 1.5.3 presents some systems that try to address this problem.

1.5.2 Database Experience

User experience with the database where the search is performed is the other important factor for the success of the search request. Even though users may be experts in the subject area covered by a database, the particular way documents are indexed may prevent users from expressing their queries using the most convenient terms. Differences in terminology make difficult for users to come up with all the terms that may be used to describe a concept. For this reason some interactive systems include mechanisms that allow users to browse the vocabulary of the databases to find the way concepts are described. Burgess [BS86] presents a system which can be used by casual users to browse a hierarchy using a graphical interface. This browsing process is done until appropriate terms are found to be included in the query. The OAK system [MCBC89] organizes the search requests as a conjunction of facets; where each facet is a disjunction of several terms. In order to fill up the facets the system allows the user to use windows to search for the different kinds of terms: author names, free vocabulary, and controlled vocabulary.

Some work has been done to use expert systems to simulate the choice of search terms as it is done by human intermediaries. In [Fid86] the actions taken by human intermediaries were coded into rules; some of which were susceptible to automation. This rule-driven system was found to have problems whenever an unsatisfactory term has to be improved since the creativity used by the human intermediaries to solve that situation is hard to simulate.

1.5.3 Intelligent Intermediaries

Before finishing this section, some words must be said about systems that try to automate the job of the human search intermediaries. The objective of these systems is to go beyond the services provided by the front-ends and give user-assistance during the search process and provide advice whenever trouble is encountered. These systems usually are implemented using expert system methodology which approximates the knowledge of the human search intermediaries. In the IR-NLI [GT83] the system represents the knowledge by frames whose slots must be filled up. The process of filling up these frames simulates the interview that the human intermediary has with the user before proceeding to formulate the query. The IIDA system mentioned before [MHA82] is oriented towards helping the users find a few relevant documents; it is not intended to be used for difficult searches which are better served by human intermediaries. Some systems like OAK [MCBC89] only give limited advice like pointing out terms retrieving too many documents or telling users to review the documents retrieved before modifying the query. Despite its simplicity this type of advice was found to be effective and users appreciated it. Fidel [Fid86] examines the possibility of using expert systems in the selection of search keys.

The wisdom of approaching the search intermediary role as an expert system problem is analyzed by Brooks [Bro87]. As pointed out there, the most successful expert systems deals with narrow, homogeneous subject areas; whereas intelligent intermediaries in information retrieval have to deal with a large and heterogeneous domain. For these reasons the possibility of developing a complete expert system to replace search intermediaries does not look bright.

Chapter 2

Experimental Environment

One of the objectives of this thesis is the evaluation of interactive mechanisms to help users improve the formulation of their queries. The goal is to find out whether the user knowledge can be used effectively to direct the expansion of the queries by phrase addition. Because of the well known limitations of computers in natural language understanding, many automatic query enhancement techniques tend to introduce elements that are not semantically related to the query. In relevance feedback, for example, some of the terms added have little to do with the query. This happens because some terms from relevant documents refer to aspects of the documents that are not relevant to the search.

This chapter describes the environment where experiments on interactive query modification are performed. The query modification process essentially consists in choosing term phrases to be added to the queries in order to complement the single terms normally used in retrieval. At the beginning of the chapter there is a discussion on the techniques used for the generation of the phrases. Then there is an analysis on how to select certain phrases to be displayed to the users. Phrase selection is needed to reduce the burden to the users who should not be forced to choose from the large number of phrases potentially generated by the relevance feedback process. After this analysis comes a detailed description of the query modification strategies to be evaluated. Following the strategies, the user interface used in the experiments is described in detail. Finally, the experimental design and some other considerations are discussed.

2.1 Phrase Generation

The generation of term phrases was introduced in section 1.3.3 as a precision enhancement technique to change broad terms into more specific phrases with lower document frequency. Basically two methodologies are available for the generation of term phrases,

- statistical associations between single terms can be used such as location or distance between the different components of the phrases, or term frequency, or frequency of co-occurrence of the phrase components;
- syntactical parsing of the sentences of a document can be used to create phrases with syntactical connections between their components.

The phrases used in this study are generated using the procedures introduced by Fagan [Fag87] in his performance comparison of statistically generated phrases versus syntactically generated phrases. Fagan made several runs to determine the best values for the different parameters controlling the phrase generation process. With one or two exceptions these same values are used here. Some parameters, however, do not correspond to the best values found because of the interactive nature of the experiments. Time and space restrictions make unrealistic the use of some configuration of parameters that perform well but generate an excessive number of phrases.

2.1.1 Phrase Weighting and Similarity Function

Before describing the phrase generation processes, it is important to present how term phrases are incorporated into the vector model representation of documents and queries. As mentioned in section 1.2, documents and queries are represented as vectors of terms. In order to include phrases, these vectors are organized into two subvectors; one of the subvectors contains the single terms, while the other one contains the term phrases:

$$V = << st_1, st_2, \cdots, st_n >, < ph_1, ph_2, \cdots, ph_m >>;$$

st_i is the weight or importance of single term T_i in the document or query vector V, and ph_i is the weight or importance of term phrase Ph_i in V.

The single term subvector is weighted using the *tf * idf* weights. This subvector is also normalized using the cosine normalization function. The weights of the term phrases are the average of the weights of their components. Thus if a phrase *ph* has as components single terms *a* and *b*, their weights are related by the formula:

$$w_{ph} = \frac{w_a + w_b}{2}$$

This definition of phrase weights has two advantages. First, the importance of the single terms is also reflected in the phrases. Second, the magnitudes of the phrase weights are similar to the magnitudes of the single terms.

The similarity function used in this study takes two vectors, each one containing two subvectors, and computes the weighted sum of the inner product similarities of the two pairs of subvectors. In other words, given two vectors, D_1 and D_2, with single term subvectors St_1 and St_2, and phrase subvectors Ph_1 and Ph_2 respectively, the similarity between two vectors is computed as follows:

$$\mathrm{sim}(D_1, D_2) = c_s \cdot inner(St_1, St_2) + c_p \cdot inner(Ph_1, Ph_2) \qquad (2.1)$$

Where c_s and c_p are the weights assigned to the single term and phrase subvectors. Unless mentioned, both, c_s and c_p, will be equal to 1.0.

2.1.2 Statistical Phrases

The statistical phrase generation process in this study is a generalization used by Fagan [Fag87] of a method presented by Salton et al [SYY75]. Seven parameters control the conditions to be satisfied by a potential phrase before it is accepted. These parameters will be described in more detail in the next paragraphs.

The following is an abbreviated description of the process used to generate phrases. A more detailed description of this process can be found in [Fag87, pp. 37–38]. Note that in order to use phrase generation as a precision enhancement device, one of the phrase components must be a poor discriminator. This term is called the *phrase head*, while the other elements of the phrase are called *phrase components*.

- Create dictionary of valid phrases.

- Identify terms that can become part of phrases either as *phrase heads* or as *phrase components*.

- For each *phrase head* construct phrase candidates using co-occurring *phrase components* different from the head. The candidate phrases must respect the restrictions on domain, phrase length, proximity.

- Accept candidate phrases if they appear in dictionary.

- Query phrases have the additional restriction that they must appear in at least one document.

- Single terms are accepted according to their particular restrictions. These restrictions may be different for phrase terms than for terms not appearing in phrases.

- The single terms identified in the queries have the additional restriction that they must appear in at least one document.

The process described above is controlled by the following parameters. For each parameter, a comparison is made between the best values found by Fagan and the values used in this study.

domain: unit of text where the elements of a phrase must co-occur. The *domain* may be as unrestricted as a document or as specific as a sentence. Even though it is expected that restricted domains are more likely to generate meaningful phrases Fagan found that the most unrestricted domain gives the best performance. Because of that result this study uses a complete document as domain.

proximity: maximum distance between phrase components. A proximity of one for example, requires the phrase components to be adjacent. Even though the smaller the proximity the more likely it is the phrases include related components, Fagan found that the best results are obtained by not setting up restrictions in the proximity. In this study, unrestricted proximity parameter is not convenient because too large a number of phrases will then be generated. For this reason, the number of phrases presented to the user is reduced to a small value (20 phrases). A proximity value of 1 is then used since this reduces the number of phrases and is more likely to generate phrases that are semantically meaningful to the users.

df_phrase: maximum and minimum document frequency of the phrases. This parameter is used to remove phrases which are either too frequent or too rare. Fagan found that retrieval is only marginally affected when phrases are excluded because of their frequency. Since this parameters has such a small influence, no restrictions are placed on the frequency of the phrases.

df_head: minimum frequency of the *phrase head.* The best value for this parameter only had negligible effect on retrieval, so no lower limit is imposed on document frequency of the *phrase head.*

df_comp: minimum document frequency of the *phrase components* other than the head. This parameter can be used to avoid generating phrases with very low frequency due to the low frequency of one component.

As with *df_head*, this parameter has little effect on retrieval, so that no lower limit is imposed on the document frequency of the components.

df_st: maximum document frequency for single terms. The objective is to eliminate the bad discriminators which are characterized by high document frequency. This parameter also has only a small effect on retrieval so that no upper limit is imposed on the single term frequency.

length: number of components of the phrases. Only phrases of length two are considered because of there is a combinatorial explosion on the number of phrases with more elements.

2.1.3 Syntactical Phrases

The generation of syntactical phrases is done by using a syntactical analysis of the sentences of a document to create phrases whose components are associated by syntactical relations.

The syntactical phrases used in this study are those used by Fagan [Fag87] in his dissertation. This set of phrases was chosen because its generation has been optimized. Fagan used a full syntactic analysis system, the PLNLP English Grammar from IBM Research Laboratory [HJM+82], [JHMR83]. For each sentence the PLNLP system produces a parse tree which describes the structure of the sentence by tagging each tree node with a syntactical role indicator. Table 2.1 gives the different syntactical tags used. Figure 2.1 shows an example of a parse tree generated by PLNLP. Node 1 is the root of the tree; this node has the NP tag which means that the entire tree represents a noun phrase. The different subtrees represent syntactical constructions that

Table 2.1: Labels of syntactical tags

Node	Syntactical tag
NP	noun phrase
PP	prepositional phrase
NOUN	noun
PREP	preposition
AJP	adjetival phrase
ADJ	adjetive

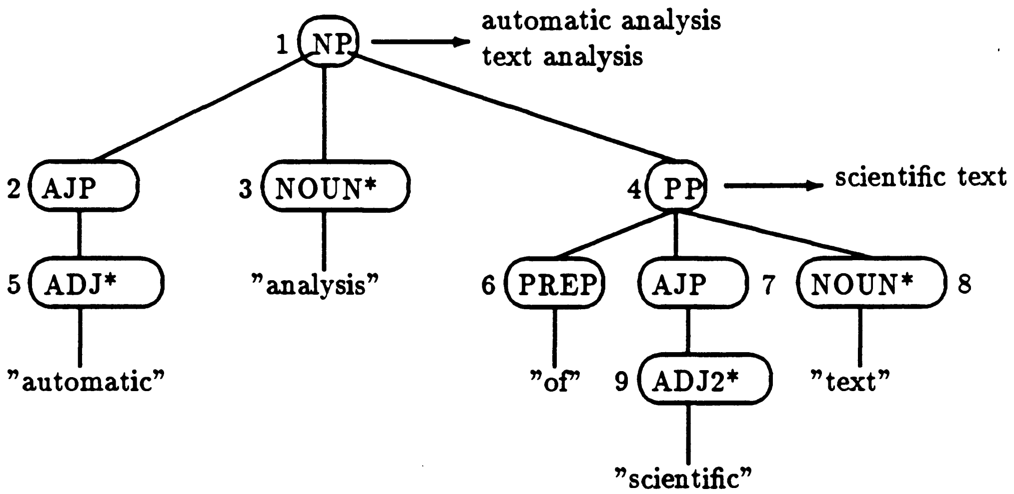


Figure 2.1: Parse tree.

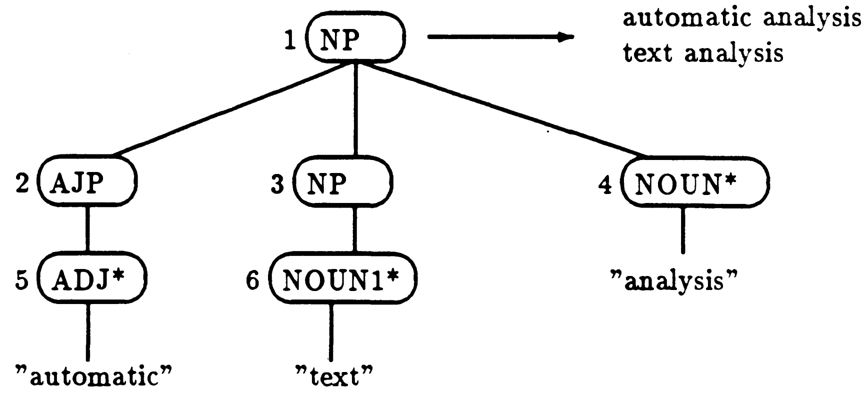


Figure 2.2: Examples of phrases extracted from a parse tree.

are part of that noun phrase. For example, the subtree whose root is node 4 (tag PP) represents a prepositional phrase that is a part of the entire noun phrase.

Each node includes among its children a distinguished member called the *head*, which is the main syntactical part of the construction represented by that node; the other children are the *modifiers* of the head. In Figures 2.1 and 2.2, the different heads are marked by a star, "*". More specifically, in Figure 2.1, a NOUN (node 3) is the head of the noun phrase represented by node 1, while another NOUN (node 8) is the head of the prepositional phrase represented by node 4. Other heads are the ADJ in node 5 and the ADJ in node 9.

The phrases produced by PLNLP are generated from subtrees extracted from the parse tree. Basically, given a subtree representing a syntactical construction, the head of that construction is combined with the heads of its modifiers to generate the phrases. For example, in Figure 2.1, the phrases *automatic analysis* and *text analysis* are generated when node 3 (*analysis*), which is the head of the whole tree, is combined with node 5 (*automatic*) and node 8 (*text*) which are the heads of the subtrees with roots 2 and 4. Another phrase is generated when the same methodology, combine head of construction with heads of modifiers, is applied to the subtree whose root is node 4; in that case the phrase *scientific text* is generated.

This phrase generation process provides a *normalization* of the phrases since the same phrases can be generated from syntactically different but semantically close constructions as is shown in Figures 2.1 and 2.2 where the phrases, *automatic analysis* and *text analysis* are obtained from two different parse trees.

The most important rule used for phrase generation will be presented as an illustration. Additional rules are included in the Appendix A. For a complete presentation of the rules see [Fag87].

Noun phrases are syntactical constructions considered to be good sources of phrases for document indexing. Some restrictions, however, are usually applied to reduce the number of phrases generated. In the present application, the phrases are constructed by combining the head of the noun phrase with the heads of the modifiers. For example, in Figure 2.2 $NOUN_2$, the head of NP_1, is combined with the heads of the modifiers AJP and NP_2. The result of these combinations are the phrases *automatic analysis* and *text analysis*.

Four parameters were used by Fagan to control the content and influence of the phrase subvectors. As for the statistical phrases, the best values found by Fagan are not always used because of the interactive nature of the experiments.

Parse threshold: Some phrases are ambiguous and produce more than one parse tree. In those cases, a ranked list of possible parses is provided. The *parse threshold* parameter specify how many of those parses (up to ten) can be used to generate phrases. For this study only one parse is used.

Parsing mode: this parameter tells the parser whether the queries should be considered full sentences or noun phrases. Because queries are descriptions of topics rather than actions, queries are better parsed as noun phrases than as sentences. For this reason, this parameter is set so that queries will be parsed as noun phrases.

Phrase subvector weight: this is the weight of the inner product similarity of two phrase subvectors; in other words, the value of c_p in equation 2.1. The best value found by Fagan and the one used in this study is 1.25.

Document frequency of phrases: this parameter represents the upper bound on the document frequency of the phrases. Fagan found that restrictions in the frequency of the phrases has a small positive effect on retrieval. In the present study, however, no restrictions are imposed on the phrase frequencies because it is believed that users need many phrases to choose from and that relatively few syntactical phrases are generated in any case.

2.2 Collections

In order to perform the query enhancement experiments, two versions of the CACM collection were prepared. The CACM collection is a standard database used in experimental studies for information retrieval. It contains 3204 articles on computer science published in the journal *Communications of the CACM* and 64 queries. Also available are the relevance assessments for those queries, that is, for each query a list of those documents judged to be relevant to the topic defined in the query is available. Following the Vector Space Model (see section 1.2), the two versions represent the documents as vectors of pairs <term, weight>. These term vectors are subdivided into two subvectors; one for the single terms and the other for the phrases. In one of the versions the phrases were generated using the statistical association techniques described in section 2.1.2, while in the other the phrases were generated using the syntactic analysis presented in section 2.1.3.

Only thirty of the sixty-four queries provided by the CACM collection are used in the experiments. Thirty-three queries were removed because they either have no relevant documents in the collection, or have no relevant documents retrieved (at least one such item is needed for feedback), or have all their relevant documents retrieved in the first run (not useful for feedback), or have less than two phrases (too few phrases to choose from). To make the experimental design easier, one of the remaining thirty-one queries was randomly discarded so that thirty queries could be used.

During the indexing of the collections no stemming was performed on the terms. Stemming, which usually has a positive effect on retrieval, presents the problem that in many cases the stem left after the stemming process is so small that users have difficulty recognizing the original word. Since this study is evaluating the effect of the choices made by the users, it is necessary that the users understand the terms displayed to them. For this reason only removal of "s" (plural) is performed on the terms. An analysis of the impact of stemming on the experimental runs is done in Section 3.3.

The choice of this collection was motivated by two considerations. First of all the need to recruit searchers knowledgeable in the subject matter made a collection in computer science very convenient to use. In the second place, the collection must be sensitive to improvements on retrieval by the addition of phrases; otherwise the selection of phrases by the users will have little effect. For this reasons the CACM collection was chosen: it was relatively easy to find the searchers and Fagan [Fag87] reports that the use of phrases produces significant improvements in effectiveness.

2.3 Phrase Ranking

Some of the feedback strategies to be described in the next section may overwhelm the users with very long lists of concepts. [1] Since the relevance feedback process may generate hundreds of concepts it is unrealistic to expect that the users will want to check them all in order to select a few. Some mechanism is needed to reduce the number of concepts to be presented. One way to get this reduction is by ranking the concepts and then selecting the top ranked ones. Two issues are involved in this phrase reduction process:

[1] From now on, the term *concept* will refer to either a single term or a phrase.

how many concepts will be presented to the user, and which criteria can be used to rank the concepts. Harman [Har88] deals with the first issue and after some experiments she concludes that only twenty documents are needed if the selection is done carefully. In other words, with a good ranking of concepts there is little to gain by presenting more than twenty phrases. This study follows her and the concepts are ranked so that only the top twenty will be displayed.

To settle the second issue, how to rank the concepts, several feedback experiments on phrase ranking were performed on the CACM collection. The phrases extracted from retrieved relevant documents were ranked following five different criteria. The top twenty phrases were then used to modify the query. The results of these feedback runs are shown in Table 2.2 for statistical and syntactical phrases. The values shown are the average precision at recall points .25, .50 and .75, evaluated by using the reduced collection methodology [CCR71]. The precision values were obtained by selecting from the top twenty ranked phrases those appearing in relevant documents not yet retrieved.

The following information is used in this study to rank the phrases: the number of retrieved relevant documents where the concept appears, *postings*; the number of documents where the concept appears, *term_freq*; and the average weight of the concept in the relevant documents retrieved, *avg_rel_wt*. These three factors are combined together in five phrase ranking functions:

1. The phrases are sorted by decreasing *postings* and then by decreasing *term_freq* * *avg_rel_wt*;

Table 2.2: Phrase ranking results

	Ranking Function				
	1	2	3	4	5
statistical	0.1976	0.1950	0.1892	0.1964	0.1885
syntactical	0.1884	0.1695	0.1847	0.1904	0.1754
average	0.1930	0.1822	0.1869	0.1934	0.1819

2. The phrases are sorted by decreasing *postings*, by decreasing *term_freq* and by decreasing *avg_rel_wt*;

3. The phrases are sorted by decreasing *postings* and by decreasing *term_freq*;

4. The phrases are sorted by decreasing *postings* * *term_freq* * *avg_rel_wt*;

5. sort phrases by decreasing *postings* and by decreasing *avg_rel_wt*.

The *postings* is a very important indicator of the usefulness of phrases, since a phrase appearing in more than one relevant document is probably describing some aspect common to several relevant documents and to the topic of the search. For this reason, *postings* is used as the main sort key in most of the ranking functions. On the other hand, the *term_freq* factor stresses those phrases appearing in more documents. In general, the phrases used in this study usually have low to moderate document frequency, so this factor can be used to eliminate low frequency phrases without risking the inclusion of very high frequency phrases, which are normally useless. The third factor, *avg_rel_wt*, indicates how important is a given phrase in the retrieved relevant documents.

The results from Table 2.2 show that ranking functions 1 and 4 exhibit the best precision. Because the precision is almost identical, the choice between these two ranking functions is quite arbitrary. Ranking function 4 was chosen as the function used in this study to rank the phrases.

The same ranking methodology, ranking function 4, will be used when dealing with single terms. In this case however, there is the problem of very high frequency terms being highly ranked because of their large *term_freq* factors. This situation is compensated by the factor *avg_rel_wt* which is rather small for very high frequency terms. This balance was considered good enough to use the same ranking function for both, phrases and single terms.

2.4 Query Enhancement Strategies

The query enhancement experiments to be described here are designed to evaluate the user impact on the refinement of the search statement. Six strategies will be tested in total; four are *pre-search* enhancements and the remaining two are *post-search* enhancements.

The pre-search experiments are designed to compare the effectiveness of the statistical phrases vs. the syntactical phrases. They also compare two different ways of choosing the phrases; either by explicitly selecting the phrases to be included in the query or by removing from the query those phrases which should not to be included. The combination of these two factors defines four strategies:

$$\left\{\begin{array}{l}\text{statistical phrases}\\ \text{syntactical phrases}\end{array}\right\} \text{X} \left\{\begin{array}{l}\text{selection}\\ \text{removal}\end{array}\right\}.$$

The pre-search strategies are:

1. *Selection of statistical phrases*: the users complement the query by selecting among the phrases presented those considered useful. The phrases used in this strategy are created using statistical associations between the terms.

2. *Removal of statistical phrases*: variation of strategy 1 where all the phrases presented to the users are initially selected to be included in the query; the users remove those phrases considered not useful.

3. *Selection of syntactical phrases*: the users complement the query by selecting among the syntactical phrases presented those considered useful. The phrases used in this strategy are created using a natural language parser.

4. *Removal of syntactical phrases*: variation of strategy 3 where all the phrases presented to the users are initially selected to be included in the query; the users remove those phrases considered not useful.

The post-search experiments consist of two feedback strategies. In those strategies, the user chooses concepts to be added to the query before the query is processed again. The comparisons are done between feedback using user-selected statistical phrases and feedback using user-selected single terms. No syntactical phrases are used in the feedback strategies. This reduces the number of feedback strategies since the comparison between statistical phrases and syntactical phrases is already included in the pre-search strategies.

The post-search strategies are:

1. *Selection of feedback phrases*: in this strategy the queries are first run using only single terms from the original query. The users, then complement the query by adding some of the phrases displayed. These phrases are obtained from the relevant documents retrieved in the initial run.

2. *Selection of feedback terms*: variation of strategy 1 where instead of phrases, single terms are presented for selection.

Besides these six interactive strategies many non-interactive strategies are run as comparisons against the interactive strategies. These comparisons help determine whether or not the improvements on retrieval obtained by the interactive strategies justify the extra burden that is placed on the users.

The following are the non-interactive strategies:

1. *No phrases*: Base case. Query is run using only its single terms.

2. *Best statistical phrases*: The statistical phrases added to the query are those presented to the users which also appear in relevant documents not yet retrieved.

3. *All statistical phrases*: All the statistical phrases presented to the users are included in the query.

4. *Best syntactical phrases*: The syntactical phrases added to the query are

those presented to the users which also appear in relevant documents not yet retrieved.

5. *All syntactical phrases*: All the syntactical phrases presented to the users are included in the query.

6. *No feedback change*: Base case for the feedback runs. The original query is run again without any alteration.

7. *Best single term feedback*: The single terms chosen for feedback are those presented to the users which also appear in relevant documents not yet retrieved.

8. *All single term feedback*: All the single terms presented to the users for feedback are included in the query.

9. *Best phrase feedback*: The phrases chosen for feedback are those presented to the users which also appear in relevant documents not yet retrieved.

10. *All phrase feedback*: All the phrases presented to the users for feedback are included in the query.

11. *Best full single term feedback*: The single terms added to the query are those generated by the relevance feedback process which also appear in relevant documents not yet retrieved.

12. *Full single term feedback*: All the single terms generated by the relevance feedback process are included in the query.

13. *Best full phrase feedback*: The phrases added to the query are those generated by the relevance feedback process which also appear in relevant documents not yet retrieved.

14. *Full phrase feedback*: All the phrases generated by the relevance feedback process are included in the query.

The *all* strategies (items 3, 5, 8 and 10 in the previous list) add to the query all the concepts presented to the users. On the other hand, the *full* feedback strategies (items 11 to 14) use all concepts generated by the feedback process. That is, there is no reduction in the number of concepts. Finally, the *best* strategies (items 2, 4, 7, 9, 11 and 13) simulate the optimal selection by adding to the query those concepts that appear in relevant documents not yet retrieved. Even though users can not make this kind of selection, the *best* strategies are nevertheless useful because these strategies simulate the best choices the users can make. This type of strategy was proposed in [Har88].

2.5 User Interface for Interactive Query Enhancement

The query enhancement strategies described in section 2.4 are implemented by an interactive program which allows the users to select the concepts that will added to the query. This section describes the user interface provided by the interactive program. The interactive interface is implemented in a SUN-4 computer using the X Toolkit Widgets of the X11 Windows Release 3.0. This mouse-driven interface whose general layout can be seen in Figure 2.3 serves as a front-end for the SMART retrieval system [Buc85]. The SMART system performs the different tasks needed for document retrieval: user query indexing, phrase generation, and query execution.

As Figure 2.3 shows the interface consists of five areas or panels:

- *Query Panel*: displays the text of the query and provides the major operations of the experiments like starting a new query, indexing a query, retrieving documents, etc.

- *Selection Panel*: displays a list of concepts and allows users to choose which will be added the query.

- *Top Document Panel*: presents the titles of the top documents retrieved, allows users to review them in more detail and give relevance assessments. That is, the users tell which of the documents retrieved are relevant.

- *Single Document Panel*: shows one document in more detail; the title and the abstract of the document are displayed. This panel also allows the user to give relevance assessment on the document shown.

- *Message Panel*: displays information and system messages.

All these panels are discussed in more detail in the following paragraphs. The emphasis will be in the *Query Panel* since it contains the most important operations.

The *Query Panel* includes the area where the users enter their queries, and provides the major operations of the system. Any combination of these operations is allowed by the user interface, however, as it will be described

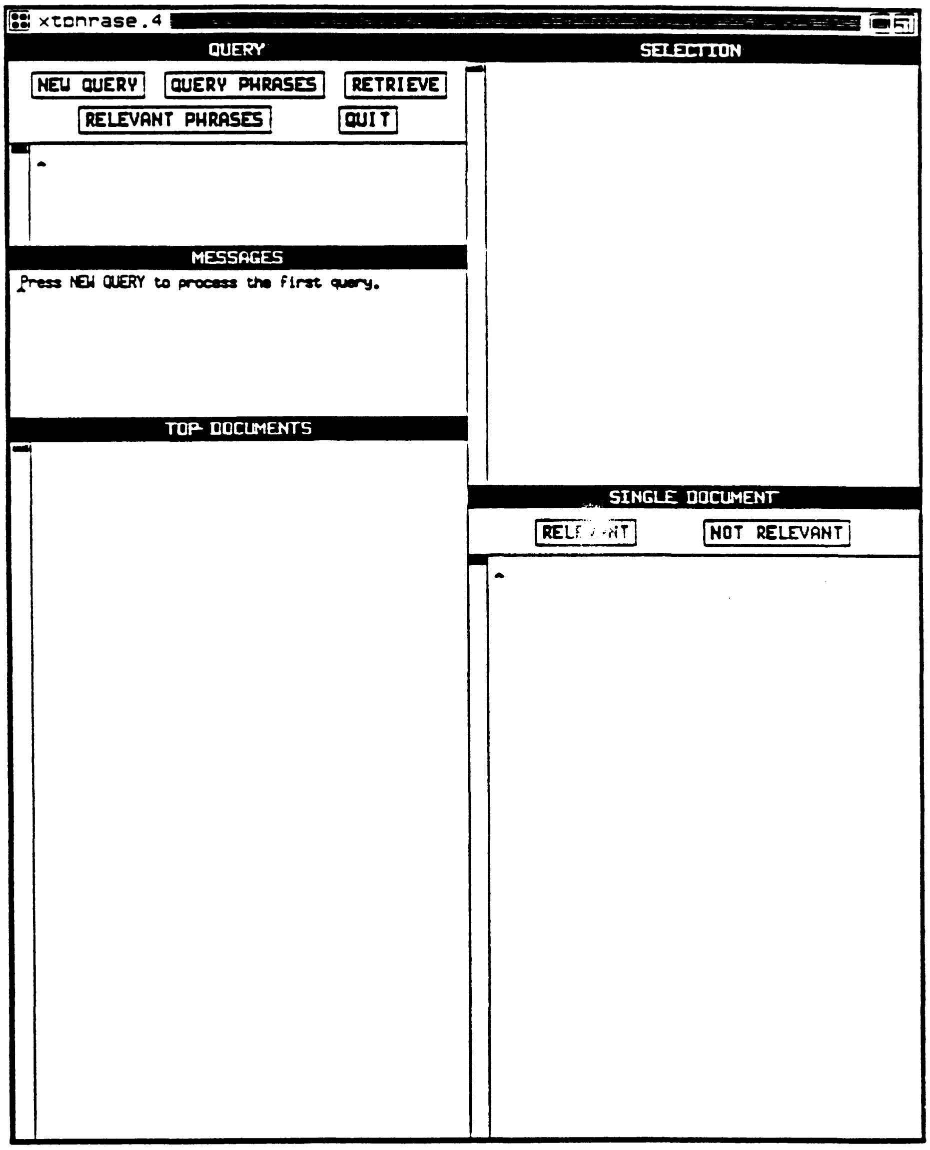


Figure 2.3: General layout of the user interface

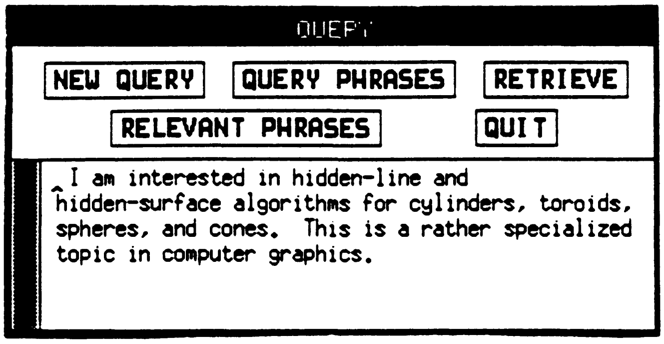


Figure 2.4: *QUERY Panel* displays text of the query

later, some restrictions are included to make the subjects follow to the proper sequences of operations that the different strategies require.

The operations of the *Query Panel* are:

- **NEW QUERY** writes the information gathered about the previous query, then resets the system so that users can enter a new query. Figure 2.4 shows this panel after some user has entered his query. Full editing of the query entered is available to the users.

- **QUERY PHRASES** generates query phrases and display them in the *Selection Panel.* There, users can choose those phrases they consider useful. Figure 2.5 shows an example of one list of phrases generated by this operation. Furthermore, figure 2.6 shows the same list after some of the phrases have been chosen.

- **RETRIEVE** runs the query expanded with the concepts chosen. The titles of the top documents retrieved are shown in the *Top Document Panel* where the users can review them in more detail and give assessments on their relevance. Figure 2.7 a list of top documents retrieved

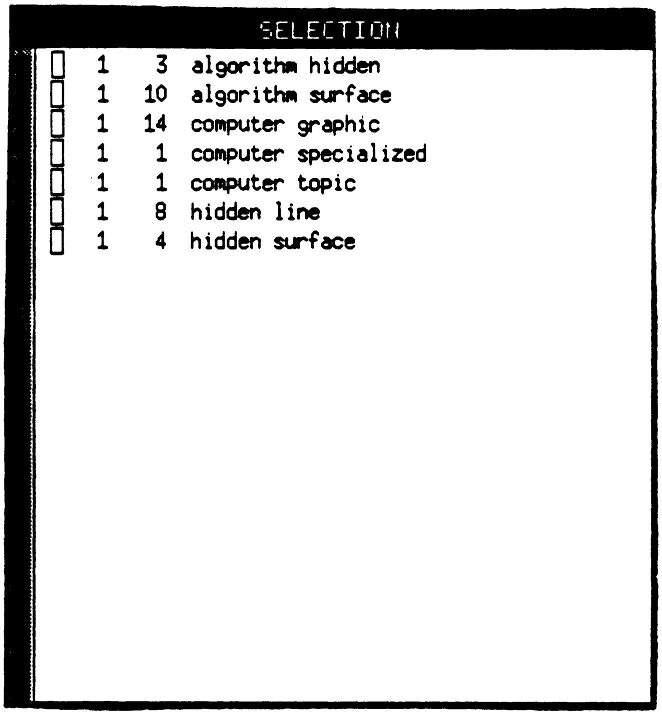


Figure 2.5: List of phrases as they are originally displayed

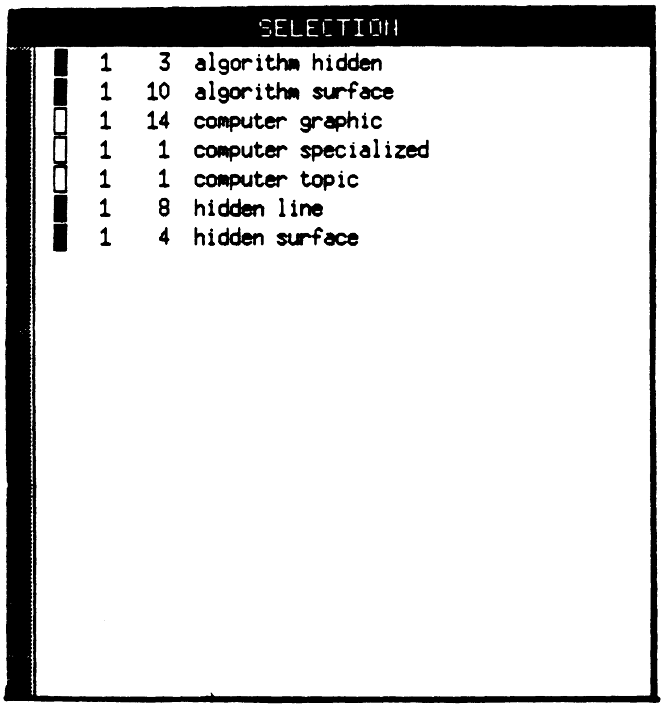


Figure 2.6: List of phrases after some of them have been selected

with some of the documents already marked as relevant/nonrelevant.

- **RELEVANT PHRASES** generates phrases or single terms (depending on the strategy) from the documents that have been marked as relevant by the users. The phrases or single terms are displayed in the *Selection Panel* where the users can choose which ones are added to the query. Figure 2.8 displays a set of phrases generated from relevant documents.

- **QUIT** writes the information gathered about the last query and exits.

The *Selection Panel* allows the user to choose which concepts will become part of the query. The concepts are displayed in alphabetical order and they are chosen using a mouse: each click of the left button of the mouse over a given concept will changed its state from *selected* to *not selected* and vice versa. As it can be seen in figure 2.6, a small square to the left of a concept becomes black when that concept is selected and becomes white when it is not selected. There are two numbers included in each line for each concept; they are displayed to give the users more information about the concepts. The first one is the number of retrieved relevant documents where the concept appears, and the second one is the the number of documents where the concept appears.

The *Top Document Panel* allows the users to review the documents retrieved and give assessments on their relevance. Each document is represented by its title, authors, journal and date of publication. The document entries are listed by feedback iteration. In other words, documents retrieved in later iterations are shown before documents retrieved in earlier iterations.

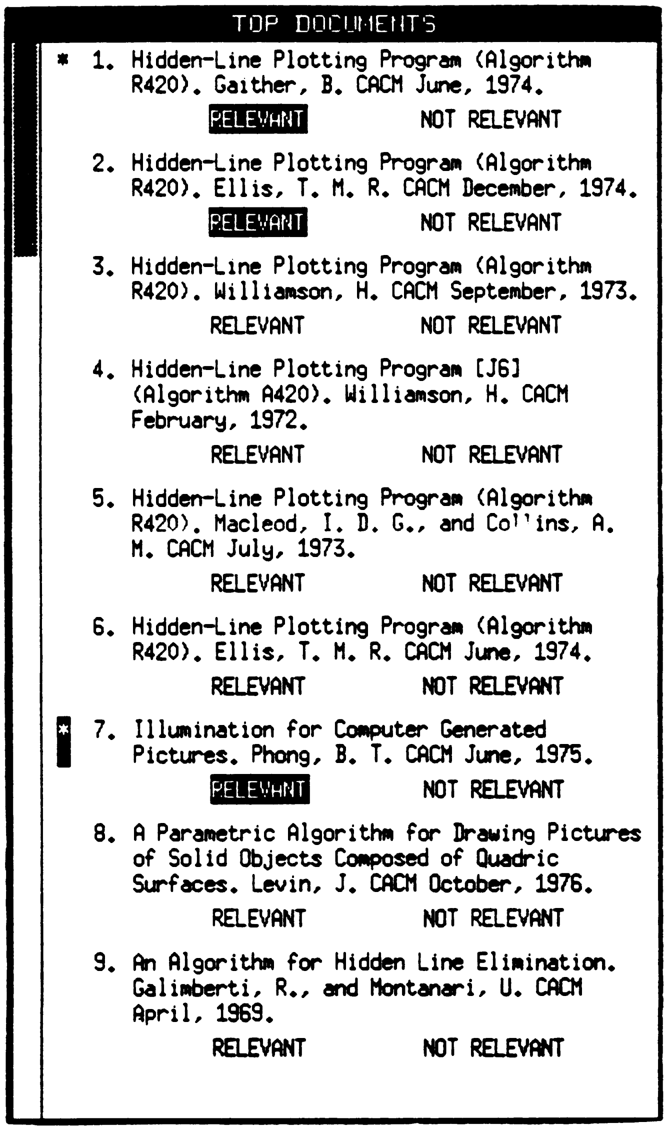


Figure 2.7: List of the top documents retrieved

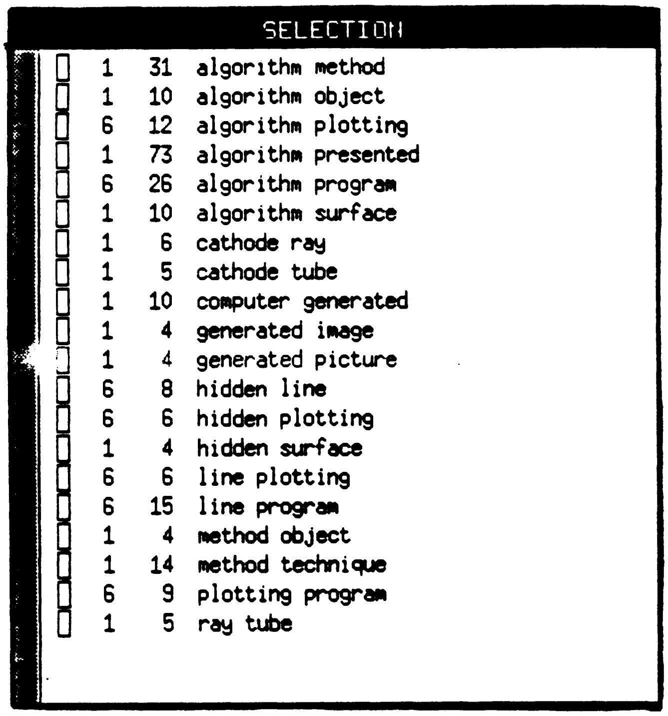


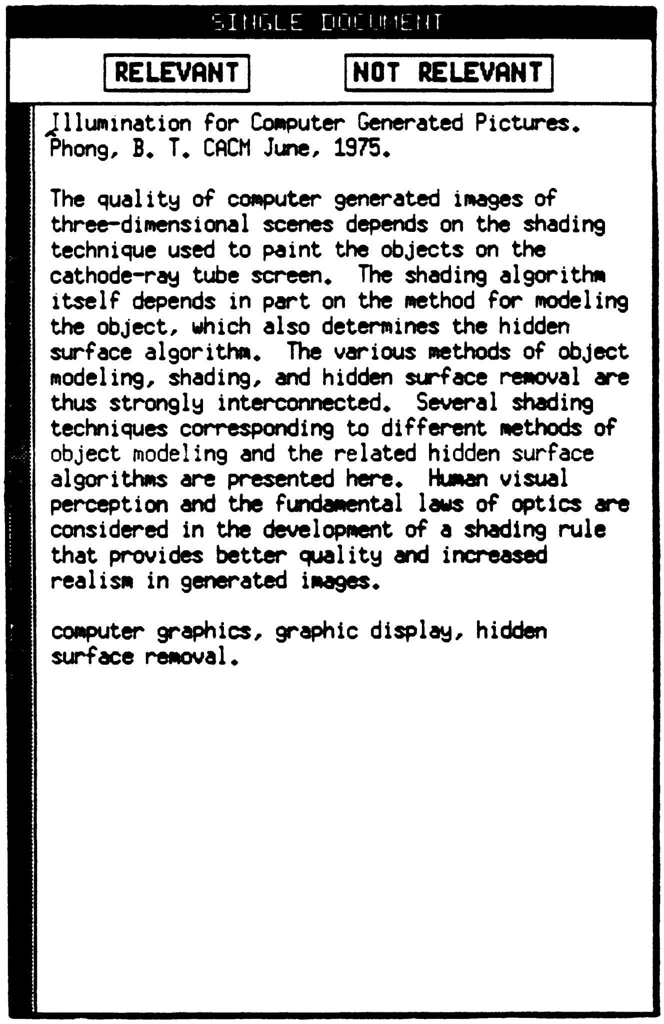
Figure 2.8: Phrases generated from relevant documents

Within each iteration, the documents are listed by decreasing similarity with the query so that documents with higher similarity appear first. For each document two areas marked *RELEVANT* and *NOT RELEVANT* are provided. Clicking the mouse on them will mark the document as relevant/nonrelevant. The area clicked will become black to show the status of the document as it is shown in figure 2.7 for document 7. The text of any of the documents listed in the *Top Document Panel* can be displayed in the *Single Document Panel.* For example, figure 2.9 shows the text for document 7 in figure 2.7. To display the text of a document, the mouse just have to be clicked over the entry of that document, then the title and the abstract of the document will appear in the *Single Document Panel.* The "*" to the right of the entries marks those documents which have been reviewed in the *Single Document Panel.*

As explained in the previous paragraph, the *Single Document Panel* shows the text (citation and abstract) of one of the documents displayed in the *Top Document Panel.* One example of the information displayed for each document can be seen in figure 2.9. Two operations, *RELEVANT* and *NOT RELEVANT*, are provided with this panel; they allow the users to give relevance assessment on the document shown on this panel. Note that the relevance assessments can also be given in the *Top Document Panel.*

2.5.1 Searcher Interface

The interactive experiments to be described in section 2.6 are performed by hired graduate students who from now on will be called *searchers.* These searchers run queries extracted from the standardized set of queries of the



SINGLE DOCUMENT

RELEVANT NOT RELEVANT

Illumination for Computer Generated Pictures.
Phong, B. T. CACM June, 1975.

The quality of computer generated images of three-dimensional scenes depends on the shading technique used to paint the objects on the cathode-ray tube screen. The shading algorithm itself depends in part on the method for modeling the object, which also determines the hidden surface algorithm. The various methods of object modeling, shading, and hidden surface removal are thus strongly interconnected. Several shading techniques corresponding to different methods of object modeling and the related hidden surface algorithms are presented here. Human visual perception and the fundamental laws of optics are considered in the development of a shading rule that provides better quality and increased realism in generated images.

computer graphics, graphic display, hidden surface removal.

Figure 2.9: Individual documents can be displayed in *Top Document Panel*

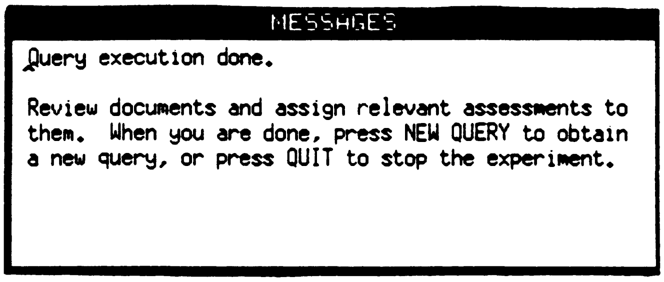


Figure 2.10: Help information and instructions are displayed in the *Help Panel*

CACM collection. In order to avoid uncontrolled variability in the actions of the searchers, for each experiment the searchers are guided to follow a specific sequence of operations. To guide the searches on their tasks the *Message Panel* provides them with information on what to do next to complete the experiment. One example of the guide provided to the searches can be seen in figure 2.10, where the system is telling a searcher that after selecting the phrases he must run the query.

The sequence of operations guiding the searchers depends on the strategy being performed. The following are the sequences for each one of the strategies defined in section 2.4:

- Phrase Selection (statistical or syntactical):

  1. Obtain query: the operation **NEW QUERY** presents one of the standard queries for the searcher to run.
  2. Generate phrases using **QUERY PHRASES**.
  3. Searcher selects the phrases to be included in the query.
  4. The expanded query is run by the operation **RETRIEVE**.

5. Searcher reviews the documents retrieved and give relevance assessments.

- Phrase Removal (statistical or syntactical):

  1. Obtain query: the operation **NEW QUERY** presents one of the standard queries for the searcher to run.
  2. Generate phrases using **QUERY PHRASES**.
  3. Searcher removes those phrases not to be included in the query
  4. The expanded query is run by the operation **RETRIEVE**.
  5. Searcher reviews the documents retrieved and give relevance assessments.

- Feedback (phrases or single terms):

  1. Obtain query: the operation **NEW QUERY** presents one of the standard queries for the searcher to run.
  2. The unmodified query is run by the operation **RETRIEVE**.
  3. Searcher reviews the documents retrieved and give relevance assessments.
  4. Searcher uses the operation **RELEVANT PHRASES** to generate phrases or single terms from retrieved relevant documents.
  5. Searcher selects the phrases or single terms to be included in the query.
  6. The expanded query is run by the operation **RETRIEVE**.

7. Searcher reviews the documents retrieved and give relevance assessments.

A final note needs to be made about the feedback strategies. If the list of feedback concepts to be presented to the searchers is actually generated from the documents they marked as relevant, then the searchers will probably end up choosing from different sets of concepts since they may have marked different documents as relevant. In order to evaluate the results however, the same set of feedback concepts must be presented to all the searchers. This is done by generating the feedback concepts from retrieved documents considered relevant under the standard set of relevance assessments available from the CACM collection. In other words, for the feedback runs the relevance assessments given by the searchers for the first run of the query are not used to generate the list of concepts.

2.6 Experimental Design

Each of the searchers performing the experiments ran a different subset of the CACM queries which covered a wide variety of topics in computer science. This diversity required users with a reasonable knowledge of different areas of computer science, so preference was given to graduate students. Twenty students were hired to run the experiments and they were paid $15 for their participation.

The experimental design takes into account the training effect. That is, the order of execution of the queries and strategies may affect the performance: the first queries run by a user are affected by his lack of familiarity with the system while the queries run in the middle of the search session may gain from the experience accumulated in previous runs. Finally, the last queries run may have bad performances because the user may be tired by then. To reduce the training effect, three extra queries are run at the beginning of the experimental session to explain the system and to train the searcher; this training is designed to reduce the disadvantage of the queries processed at the beginning. Furthermore, each user only executes eighteen of the thirty queries. This reduction in the number of queries also reduces the total effort needed to complete the experiments and avoids the degradation in performance expected at the end a session. Finally, the training effect on each query was reduced by randomizing the order of execution of the queries for each searcher; in this way, the queries may be run at the beginning of the session for some searchers, in the middle for others and at the end for a third group.

Chapter 3

Query Formulation Experiments

This chapter presents and analyzes the performance of the interactive and non-interactive strategies described in section 2.4. The non-interactive strategies are used to determine whether or not the performance of the interactive strategies are good enough to justify the intervention of the users. Many of the non-interactive strategies are based on the concept of *good* phrases. This concept was introduced by Harman [Har88] to estimate an upper bound of the user performances. The idea is that the best phrases the user can choose are those making the query more similar to the relevant documents not seen yet. Thus, a good phrase is one that appears in new relevant documents. The selection of concepts appearing in relevant documents not yet retrieved can only approximate the best possible selection because those concepts may degrade performance by improving the rank of non relevant documents more than the rank of relevant ones.

Clearly this is only an approximation since a concept appearing in relevant documents not yet retrieved may degrade performance by improving the rank of non relevant documents more than the rank of relevant ones. However, it is considered that this concept of *good* phrases gives a good estimation of the best improvements users can obtain.

For the evaluation of the experimental runs, the average precision at 21 recall points is used. [1] On the other hand, the statistical significance of the differences in the performances is evaluated using the Wilcoxon signed rank test for paired observations. A difference between two strategies is considered *significant* if the test gives a 0.05 probability of making a mistake when assuming there is a difference. Furthermore, the difference is considered *very significant* if the probability of error is smaller than 0.01. In some cases the difference may be significant but its relative size is not very large. A difference exceeding 10% is considered *material.*

3.1 Pre-search Strategies

The pre-search strategies enhance the formulation of a query before it is run for the first time. These strategies are organized in two sets. In the first one, the phrases presented to the searchers are generated by statistical associations of the single terms. In the second set, the phrases are obtained from a syntactical analysis of the query. This section presents the performance results obtained by the two sets of pre-search strategies. Following that presentation there is a comparison of the effectiveness of the two types

[1]The points go from 0.0 to 1.0 in 0.05 increments.

of strategies.

3.1.1 Statistical Phrases

Table 3.1 shows the average precision obtained by the statistical pre-search strategies. In the *selection* strategy the searchers directly choose the phrases to be included in the query. On the other hand, in the *removal* strategy the searchers indirectly choose the phrases by deleting the ones they did not consider appropriate. These two alternative ways to modify the queries are included in the experiments in order to compare them and find out whether or not one is better than the other. Besides the strategies mentioned above, two non-interactive strategies are also considered here. The strategy called *best* approximates the optimal choice of phrases by selecting *good* phrases. The second non-interactive strategy is called *all* and it simply takes all the phrases presented to the searchers and includes them in the query. Note that in order to justify the intervention of users, the strategies *selection* and *removal* must perform significantly better than *all*, since the last one can be done automatically without bothering the users. Another non-interactive strategy, *none*, is the base case: the query is run using only single terms without any phrase.

The first line of Table 3.1 gives the average precision of the different strategies; the percentage differences between the strategies and the base case *none* are shown in the second line; the last line gives the significant level of difference. As the table shows, all the strategies obtained *very significant* improvements over the base strategy. In all cases the difference was also *material* since the improvements are larger than 10%. The largest improvement

Table 3.1: Effectiveness of statistical phrase strategies

	None	Selection	Removal	Best	All
Avg. Prec.	0.2536	0.2824	0.2830	0.2931	0.2797
		+11.3	+11.6	+15.6	+10.3
Sign. Level		.01	.01	.01	.01

Table 3.2: Statistical phrase frequencies

	Presented			Selected		
	# phrases	# good	% good	# phrases	# good	% good
Selection	8.5	5.0	60.1	3.4	2.4	73.0
Removal	8.5	5.0	60.1	4.4	2.9	70.6

was obtained by the *best* strategy with a 15.6% increment in precision. Even though strategy *best* is significantly better than *all*, the difference is not material since the precision of the first one, 0.2931, is only 4.8% better than the precision of *all* (0.2797). The interactive strategies, *selection* and *removal* have similar performance; their precision, however, is closer to the precision of *all* than to precision of *best*. From these results it can be concluded that even though the searchers could have chosen better phrases, the improvements obtained are not large enough to justify the involvement of users since a simple non-interactive strategy can perform almost as better.

Table 3.2 shows some data about the phrases presented and selected by the searchers. As can be seen few phrases (8.5) are presented on average. From these phrases 5.0 phrases are good in the sense explained at the introduction of this chapter. In the *selection* strategy the searchers averaged 3.4 phrases per query, 2.4 of which considered good. Similarly in the *removal* strategy the users chose 4.4 phrases per query with 2.9 of these phrases considered good. These figures suggest that the users sharply reduce the number of phrases but tend to include more phrases in the *removal* strategy. The difference in the number of phrases seems too small to have an impact on the performance. Even though users selected less than half the phrases, they successfully increased the proportion of good phrases from 60% to 70%. Unfortunately the number of phrases involved is so small that random factors could have caused the increment in the proportion of good phrases.

3.1.2 Syntactical Phrases

Similar pre-search strategies to the ones described in the previous section are used to test the effect of adding phrases generated using syntactical analysis to the query. *Selection* and *removal* are the interactive strategies, and *best* and *all* are the non interactive ones. As before, *none* is the base case where no phrases are added to the query.

Table 3.3 presents the performance results obtained by the syntactical pre-search strategies. All strategies obtained *significant* improvements over the base strategy. Unlike the statistical strategies, the improvements are not *material* because they are smaller than 10%. *Best* got the largest improvement in precision, 9.6%, and it is significantly better than *all*. However, the magnitude of the difference is small, 2.6%. In general, the performance of

Table 3.3: Effectiveness of syntactical phrase strategies

	None	Selection	Removal	Best	All
Avg. Prec.	0.2536	0.2695	0.2716	0.2780	0.2710
		+6.2	+7.1	+9.6	+6.9
Sign. Level		.05	.05	.05	.05

Table 3.4: Syntactical phrase frequencies

	Presented			Selected		
	# phrases	# good	% good	# phrases	# good	% good
Selection	3.9	2.5	63.0	2.4	1.6	68.0
Removal	3.9	2.5	63.0	2.7	1.8	69.2

the four strategies is so similar that the largest difference between them is only 3.1%. As it was the case with the statistical strategies, the syntactical interactive strategies, *selection* and *removal*, did not get improvements large enough to justify not using the non-interactive *all* strategy.

The Table 3.4 presents some data about the phrases presented to the searchers and selected by them. As it can be seen very few phrases are presented; on average only 3.9 syntactical phrases are shown to the searchers; 2.5 of these phrases are considered good. For the *selection* strategy the searchers on average chose 2.4 phrases per query including 1.6 good phrases. In the *removal* strategy the searchers on average chose 2.7 phrases per query including 1.8 good phrases. These figures parallel the situation encountered with the statistical strategies. Searchers sharply reduce the number of phrases but tend to have more phrases in the *removal* strategy than in the *selection* strategy. The searchers improved the proportion of good phrases from 63% to around 69%.

3.1.3 Comparison of Statistical against Syntactical Phrases

A comparison of the results obtained by the statistical and syntactical pre-search strategies is now in order. For all the strategies, the statistical phrases have a greater precision than the syntactical phrases. The difference is the largest for the *best* strategy which is 5.4% more effective for statistical phrases than for syntactical phrases. On the other hand, the difference is the smallest for the *all* strategy, 3.2%.

This slight difference in effectiveness can be explained by the fact that although both types of phrases have similar percentages of good phrases presented and selected (around 70%) there are more statistical phrases than syntactical phrases. More precisely, 8.5 statistical phrases are presented to the searchers in contrast to only 3.9 syntactical phrases. Furthermore, the searchers select at least one more phrase in the statistical case than in the syntactical case. The more limited number of syntactical phrases selected makes weaker modifications to the query with the result of smaller improvements in precision.

In order to determine whether or not the better performance of statistical phrases is due to their larger frequencies, a new set of queries was prepared to remove that difference. Each one of these queries has statistical phrases in its phrase subvector; the number of these phrases was made to correspond to the number of syntactical phrases available that query. The reduction in the number of statistical phrases was obtained using the phrase ranking methodology used for the feedback phrases as explained in section 2.3. Several strategies are possible when the reduced set of phrases is applied to the statistical strategies presented before. First, the *all* strategy uses all the phrases in the reduced set; the *best* strategy extracts from the reduced set those phrases appearing in relevant documents not yet retrieved; and finally, the *select* and *removal* strategies correspond to the strategies where the user choices are restricted to the reduced set. The frequencies of *good* phrases for these strategies can be seen in Table 3.5 along with the frequencies for the statistical and syntactical phrase strategies. A comparison of the frequencies of the syntactic phrases and the frequencies of the reduced statistical phrases shows similar values.

Table 3.6 illustrates the change in the average precision for the reduced statistical phrases. The change made the statistical and syntactical phrases very similar in performance; the largest difference between the reduced statistical phrases and the syntactical phrases is only 1.6% for the *best* strategy, while the largest difference between the statistical phrases and the syntactical phrases is 5.4% also for the *best* strategy.

Therefore, it can be concluded that the statistical strategies have an small edge over the syntactical strategies because the former have a larger number of phrases. These results must be considered only as a trend since the sign

Table 3.5: Frequencies of reduced statistical phrases

	Presented			Selected		
	# phrases	# good	% good	# phrases	# good	% good
Statistical Phrases						
Selection	8.5	5.0	60.1	3.4	2.4	73.0
Removal	8.5	5.0	60.1	4.4	2.9	70.6
Reduced Statistical Phrases						
Selection	3.9	3.2	81.3	2.0	1.7	85.4
Removal	3.9	3.2	81.3	2.6	2.1	81.9
Syntactical Phrases						
Selection	3.9	2.5	63.0	2.4	1.6	68.0
Removal	3.9	2.5	63.0	2.7	1.8	69.2

Table 3.6: Effectiveness of reduced statistical phrase strategies

	None	Selection	Removal	Best	All
Statistical Phrases					
Avg. Prec.	0.2536	0.2824	0.2830	0.2931	0.2797
		+11.3	+11.6	+15.6	+10.3
Reduced Statistical Phrases					
Avg. Prec.	0.2536	0.2711	0.2690	0.2735	0.2709
		+6.9	+6.1	+7.8	+6.8
Syntactical Phrases					
Avg. Prec.	0.2536	0.2695	0.2716	0.2780	0.2710
		+6.2	+7.1	+9.6	+6.9

test only show statistically significant differences between the *best* and the *all* strategies for each of the three groups of phrases: reduced statistical, statistical and syntactical.

Given that the statistical phrase strategies are slightly more effective than the syntactical phrase strategies and because the first ones are easier to generate than the second ones, the use of syntactical phrases instead of statistical phrases can not be justified at this point. On the other hand, the number of phrases generated has an influence on the performance of the different strategies. The statistical strategies systematically outperformed the syntactical strategies whenever more statistical phrases than syntactical phrases were used; once the number of phrases was equalized the difference disappear.

3.2 Post-Search Strategies

The post-search strategies enhance the formulation of the query after it has been run at least once. The most important information they take into account is the relevance assessments given by the users. Based on the knowledge of which retrieved documents are relevant the query is modified by adding phrases or single terms extracted from those documents. In this study the post-search strategies are presented in three groups. While the first group is characterized by using single terms to modify the query, the second group uses statistically generated terms to modify the query. In contrast to the first two groups the third group does not restrict the number of phrases or single terms that can be used to modify the query. As explained in section 2.3 only twenty concepts are shown to the searchers in the interactive strategies,

Table 3.7: Effectiveness of single term feedback strategies

	None	Single term	Best	All
Avg. Prec.	0.1476	0.1815	0.1895	0.1891
		+23.0	+28.4	+28.1
Sign. Level		.01	.05	.05

so at most twenty concepts from retrieved relevant documents are used to modify the query. For the third group of strategies all phrases and single terms generated are used to modify the query.

3.2.1 Single Term Feedback Runs

The first group of feedback strategies contains the interactive single term feedback strategy, *single term.* In this strategy the users after running their queries and marking which of the retrieved documents are relevant, proceed to select among the single terms extracted from the retrieved relevant documents those to be included in the query. There are also two non interactive strategies in this group; the one called *st_best* selects those single terms which appear in relevant documents not yet retrieved. The other non-interactive strategy is called *st_all* and includes in the query all the single terms presented to the searchers. The *none* strategy is the base case where the query is run again without modifications.

Table 3.7 shows for each strategy the average precision, the percentage improvement over the base case and the significant level of those improvements. The evaluation of all the feedback runs is done using the reduced collection

Table 3.8: Feedback concept frequencies

	Presented			Selected		
	# phrases	# good	% good	# phrases	# good	% good
Terms	20.0	19.1	95.5	3.7	3.6	95.8
Phrases	19.9	12.4	62.2	4.2	3.0	71.5

methodology described in section 1.3.4. All the strategies have much better precision than *none*, and the improvements are not only significant in all cases but they are material. *St_best* and *all* have almost identical performance and they are 4.2% more effective than the interactive strategy, *single term*, but this difference is neither significant nor material. These results do not support the involvement of users in the relevance feedback process since the performance of the *single term* strategy was not materially superior than the performance of the non-interactive strategy *all*. An analysis of the *goodness* of the single terms presented will clarify why users only have limited impact given the single terms presented to them.

Table 3.8 shows some data for the single terms presented to the searchers and selected by them. As the Table shows, twenty single terms are displayed per query; nineteen of those single terms are considered good since they appear in relevant documents not yet retrieved. Following a similar trend notice in the pre-search strategies, the users drastically reduced the number of single terms. On average only 3.7 terms are chosen per query; most of them, 3.6, are good. Because of the large proportion of good terms, the

Table 3.9: Effectiveness of phrase feedback strategies

	None	Phrase	Best	All
Avg. Prec.	0.1476	0.1659	0.1951	0.1816
		+12.4	+32.2	+23.1
Sign. Level		.01	.01	.05

similar performance of *st_best* and *st_all* can be easily explained since they differ in only one term out of twenty. In other words, users have no choice but to select good terms.

3.2.2 Phrase Feedback Runs

The second group of feedback strategies contains the interactive phrase feedback strategy, *phrase*. In this strategy the users after running their queries and marking which of the retrieved documents are relevant, proceed to select among the phrases extracted from the retrieved relevant documents those to be included in the query. As with the first group, there are two non-interactive strategies in this group; the one called *ph_best* selects those phrases which appear in relevant documents not yet retrieved. The other non-interactive strategy is called *ph_all* and includes in the query all the phrases presented to the searchers.

Table 3.9 shows for each phrase feedback strategy the average precision, the percentage improvement over the base case and the significant level of those improvements. As before the reduced collection methodology was used in the evaluation of these feedback runs. All the strategies have much better precision than *none*, and the improvements are not only significant in all cases but they are material.

Of all the strategies, *best* has the largest improvement of all with an increment of 32.2% over *none*. The two non-interactive strategies have significant improvements over the interactive strategy: *best* is 17.6% more effective than *phrase* and *all* is 9.5% more effective than *phrase*. The very good performance of *best* shows that the searchers have the possibility of significantly increase the effectiveness of a feedback run. Note however that the searcher choices did not performed better than *ph_all* which suggests that they have trouble finding the *good* phrases.

Table 3.8 shows frequency information for the phrases presented to the searchers and selected. As the table details, almost twenty phrases per query are display, and 12.4 of those phrases are considered good since they appear in relevant documents not yet retrieved. Continuing the trend found before, searchers sharply reduced the number of phrases. On average only 4.2 phrases are chosen per query. The proportion (71%) of the good phrases chosen by the users is greater than the proportion (62%) of good phrases originally present among the phrases displayed. The searchers seem to be able to improve somewhat the proportion of good phrases; however, the magnitude of the change and the number of phrases chosen are too small to be significant.

3.2.3 Feedback Comparisons

Comparisons are now appropriate between the performance of the feedback runs using single terms and the performance of the feedback runs using phrases. First, *st_all* obtained an improvement of 28.1% over the base case, which is greater than the 23.1% improvement got by *ph_all*. This difference can be explained by the fact that single terms have a better proportion of good concepts than the phrases have. More specifically, 95.5% of the single terms presented to the searchers are *good*, while only 62.2% of the phrases presented are *good*.

The *phrase* strategy is 12.4% more effective than the base case which is smaller than the 23.0% improvement obtained by the *single term* strategy. Two factors combined to produced the lower precision of the *phrase* strategy. In the first place, its proportion of good phrases, 71.5%, is lower than the 95.8% of the *single term* strategy. Secondly, the number of phrases chosen by the searchers is sharply smaller than the total number of phrases presented to them. The combination of these two factors explains why a strategy like *ph_all* which also has a low proportion of good phrases but has many more phrases performs better than *phrase*.

The final comparison of this section is between strategies *st_best* and *ph_best*. In this case the situation is reversed because the phrase strategy *ph_best* is 32.2% more effective than the base case which is greater than the 28.4% improvement obtained by *st_best*. Furthermore, *ph_best* has the largest increment in precision of all the strategies discussed in this section. As mentioned before *st_best* do not differ too much from *st_all* because most all the single terms are good. On the other hand, *ph_best* is very different from *ph_all* since only 12 out of 20 phrases are good, so there are eight bad phrases extra in *ph_all* that are absent in *ph_best*. The advantage of *ph_best* over *st_best* arises from the fact that in both cases the concepts added are good, but phrases are precision enhancing devices so that the addition of so many good phrases will definitely have a much better impact on precision.

3.2.4 Full Feedback

The strategies analyzed so far, being interactive or non interactive restrict the modifications to the query to at most twenty concepts. This is done because in an interactive situation users are not willing to select concepts from a very long list. Relevance feedback, however, generates many concepts, making it necessary to extract a subset of the feedback concepts. Section 2.3 details the techniques used in this study to rank the concepts generated and extract a subset from them. This extraction sharply reduces the amount of information available. In order to study the effects of this reduction, four non-interactive strategies are run. The strategy *st_full* correspond to the usual feedback process where the query is modified using all the single terms available from retrieved relevant documents. On the other hand, *st_full_best* modifies the query using single terms generated as before but they must also appear in relevant documents not yet retrieved. The other two strategies, *ph_full* and *ph_full_best* are the respective versions of *st_full* and *st_full_best* for phrases.

Table 3.10 shows the average precision for the four strategies just described. All of them have very good improvements over the base, and in both cases the *best* strategies are significantly better than the *full* strategies. This difference is larger for the phrase strategies, where *ph_full_best* is 9.3% better than *ph_full*. For single terms strategies the corresponding difference between *st_full_best* and *st_full* is only 2.3%. Furthermore, the *full* strategies are significantly better than the restricted strategies, with the exception of

Table 3.10: Effectiveness full feedback strategies

	None	st_full_best	st_full	ph_full_best	ph_full
Avg. Prec.	0.1476	0.2307	0.2255	0.2306	0.2110
		+56.3	+52.8	+56.3	+43.0
Sign. Level		.01	.01	.01	.01

Table 3.11: Full feedback concept frequencies

	Generated			Extracted		
	# phrases	# good	% good	# phrases	# good	% good
Terms	85.9	72.1	81.1	20.0	19.1	95.5
Phrases	94.4	47.4	46.1	19.9	12.4	62.2

ph_full, even though this is 8.1% more effective than *ph_best*, that difference is not significant. As was the case with the restricted strategies, the phrase strategies seem to gain more by the selection of good concepts than the single term strategies. An explanation of this can be found in Table 3.11. The feedback process generates on average 85.9 single terms per query; 72.1 of those single terms are good which is 81%. On the other hand, 94.4 phrases per query are generated on average, but only 47.4 (46%) of them are good. These figures show that *ph_full_best* is much more different from *ph_full* than *st_full_best* is from *st_full*.

3.3 Runs Using Stemming

So far the results that have been described were obtained using the collections presented in section 2.2. As mentioned there, no stemming beyond the removal of plural endings like "s" is performed because in many cases the stems extracted are so short that end-users can not figure out the original words. This problem is unfortunate since stemming is known to be a simple and effective technique for the enhancing of retrieval performance.

This section reviews the impact that stemming has on the performance of the different strategies. In order to do that, two new document collections were prepared. These collections are stemmed versions of the two collections used to get the results described in previous sections. The stemming algorithm used is the one provided by the SMART system [Buc85]. One collection has phrases generated by the statistical methodology presented in section 2.1.2, while the other collection uses phrases created following the syntactical rules of section 2.1.3.

When the queries corresponding to the different strategies are stemmed, the concept weights require special attention because stemming modifies the term frequency and the document frequency of some concepts; values which are used to compute the weights of the concepts. In other words, not only the concepts have to be changed but they must have the correct weight. Figure 3.1 illustrates how the stemming translation takes place. Vector_1 is a query vector from the original, unstemmed, collection; Vector_2, is simply Vector_1 with all of its concepts stemmed. Note how some of Vector_2 concepts represent two or more of Vector_1 concepts. For example, the term *comput*

$$\begin{pmatrix} <\text{computer}, 0.101> \\ <\text{computation}, 0.118> \\ <\text{system}, 0.049> \\ <\text{message}, 0.482> \\ <\text{synchronization}, 0.181> \end{pmatrix} \Longrightarrow \begin{pmatrix} <\text{comput}, 0.145> \\ <\text{system}, 0.055> \\ <\text{mes}, 0.539> \\ <\text{synchron}, 0.183> \end{pmatrix}$$

$$\text{vector}_1 \qquad\qquad\qquad \text{vector}_2$$

Figure 3.1: Translation of vector from unstemmed to stemmed collection.

in Vector$_2$ correspond to two terms in Vector$_1$: *computer* and *computation*. Besides stemming the terms, new weights are computed since the internal and global frequencies of the terms change when they are stemmed; this explains why the pair $<\text{message}, 0.482>$ becomes $<\text{mes}, 0.539>$.

Tables 3.12, 3.13 and 3.14 show the results obtained when the translated queries are run. Also included in those tables are the results of the unstemmed runs The *Unstemmed* line gives the precision of the strategies running in the original, unstemmed, collections; on the other hand, the *Stemmed* line gives the precision obtained by the strategies running in the stemmed collections. The line following the *Stemmed* one gives the percentage of improvement of the stemmed runs over the unstemmed ones.

Overall, stemming improves the performance of the strategies presented in Table 3.12 by 10% over the unstemmed runs. The improvement is very consistent among the different strategies. While the larger increment is 13.6% for the statistical *selection* strategy, the smaller one is 10.4% for the statistical

Table 3.12: Effectiveness of pre-search strategies with stemming

	None	Selection	Removal	Best	All
Syntactical					
Unstemmed	0.2536	0.2695	0.2716	0.2780	0.2710
Stemmed	0.2767	0.2994	0.3041	0.3117	0.3030
	+9.1	+11.1	+12.0	+12.1	+11.8
Statistical					
Unstemmed	0.2536	0.2824	0.2830	0.2931	0.2797
Stemmed	0.2767	0.3208	0.3145	0.3237	0.3107
	+9.1	+13.6	+11.1	+10.4	+11.1

best strategy. Because of this consistency in the change leaves the analysis of the results is basically equal to the discussion done in the previous sections:

- The interactive strategies, *selection* and *retrieval*, did not reach the level of effectiveness of the *best* strategies. In other words, it seems that the good phrases do not look so to the searchers since they tend to choose very few of them.

- The precision of the *best* strategies, although somewhat better than the precision of the *all* strategies, is not good enough to justify the intervention of users.

- The statistical strategies are slightly more effective than the syntactical ones.

Table 3.13: Effectiveness of post-search strategies with stemming

	None	Selection	Best	All
Single term				
Unstemmed	0.1476	0.1815	0.1895	0.1891
Stemmed	0.1752	0.2407	0.2571	0.2564
	+18.7	+32.6	+35.7	+35.6
Phrase				
Unstemmed	0.1476	0.1659	0.1951	0.1816
Stemmed	0.1752	0.2116	0.2438	0.2437
	+18.7	+27.5	+25.0	+34.2

On the post-search strategies, stemming has stronger improvements than on the pre-search strategies, but they are less uniform. As table 3.13 shows, the single term strategies get larger improvements than the phrase strategies. Furthermore, the base case has an increment noticeably smaller than the increments obtained by other strategies. These changes, however, do not basically affect the analysis done before for the runs in the unstemmed collections:

- The strategies *st_best* and *st_all* have similar precision which is larger than the precision of *single term* but not significantly.
- The strategies *ph_best* and *ph_all* have similar precision which is significantly larger than the precision of *single term*

Table 3.14: Effectiveness of post-search strategies with stemming

	None	St_full_best	St_full	Ph_full_best	Ph_full
Full feedback					
Unstemmed	0.1476	0.2307	0.2255	0.2306	0.2110
Stemmed	0.1752	0.2952	0.2937	0.2814	0.2821
	+18.7	+28.0	+30.2	+22.0	+33.7

- All the strategies have much better effectiveness than the base case.
- In the interactive strategies, searchers did not reach the level of the *best* strategies.

For the full feedback strategies, Table 3.14 compares the effectiveness obtained in the stemmed collections with the results of the unstemmed collection. Stemming produces large increments in the precision of all strategies. The *full* strategies, however, have larger improvements than the *full_best* strategies. Because of this difference in the increments, now the *st_full* and the *st_full_best* have similar performance, as it is also the case for *ph_full* and *ph_full_best*. On the other hand, the single term full strategies under stemming clearly have better precision than the phrase full strategies; the difference, however, is not very big: 4.9%.

From the results displayed in Tables 3.12, 3.13 and 3.14, it can be concluded that stemming produces a significantly improvement on all strategies; but at the same time, the relative performance of each strategy with respect to the other strategies with few exceptions does not change.

3.4 Conclusions

This section summarizes the results obtained in this chapter. The first conclusion is that the different interactive strategies, *selection*, *removal*, *single terms* and *phrases* fail to increase the effectiveness of the search to a level that can justify the extra burden placed on the users when they have to read and choose concepts. Most of the time, these strategies did not performed significantly better than simple non-interactive strategies like the different versions of *all* which simply use all the available concepts.

Another important observation is that the optimal choices, represented by the different *best* strategies, can in principle perform better than the *all* strategies. The searchers, however, did not come close to the optimal choices due in part to their tendency to choose very few concepts. A deeper analysis of the phrases chosen helps to clarify the origin of the problems. There are two aspects involved during the selection process; searchers usually choose phrases with a semantically meaningful connection between the components, while at the same time they reject phrases with little semantical connection. Unfortunately, it is the case that quite frequently phrases that look semantically good do not appear in relevant documents, while other, less appealing, phrases are actually very useful. Table 3.15 shows how the phrases of the CACM query 36 are selected. The text of the query is:

> Fast algorithm for context-free language recognition or parsing.

As can be seen in the figure, for two phrases *language parsing* and *context free* both the selection searchers and the removal searchers agreed with the

Table 3.15: Concept selection for query 36

Concept	Selection	Removal	Best
language parsing	X	X	X
context free	X	X	X
context language		X	X
algorithm fast	X		X
free language	X		X
language recognition	X	X	
parsing recognition		X	
algorithm context			X
algorithm free			X
context fast			

optimal choice. For the following three phrases only one group of searchers decided to choose it. Note that the phrase *language recognition* which even though is semantically correct and is clearly one of the major topics of the query, does not contribute to the retrieval on new relevant documents since it is not part of the set of best choices. Finally, two phrases not chosen, *algorithm context* and *algorithm free* are actually useful to improve retrieval because they appear in relevant documents not yet retrieved.

Two reasons can be given to explain the usefulness of the little appealing phrases. In the first place, these phrases are pieces of larger and meaningful phrases; for example, *algorithm context* and *algorithm free* are two-term phrases generated from the four-term phrase *context-free language algorithm* which is clearly useful. Since the restriction to two term-phrases is likely to remain in place for the statistically generated phrases as long as the combinatorial explosion mentioned in page 38 can not be avoided, users must be adviced to choose two-term phrases which are part of larger meaningful phrases, even though the components of the two-term phrases do not have strong semantical connection. Another approach is to improve the quality of the syntactical phrases generated by Fagan [Fag87] and used in this study. Longer syntactical phrases can be easily generated without generating too many phrases.

On the other hand, as Lesk [Les69] argues, for medium collections some associations between terms are local and do not correspond to the general meaning of the terms. Because of these local associations, many phrases that are not chosen because the components are not related are actually useful because they are synonyms of terms that improve retrieval.

It is necessary to note that there is a contradiction between the best way to represent information for the users and the best way to represent the information for the system. This can be clearly seen in the feedback runs where the full feedback strategies performed much better than the strategies restricted to twenty feedback terms; in other words the reduction in the number of feedback terms needed to make user selection feasible has a strong negative impact on retrieval. Another example of this tension between user and system is in the use of stemming. As section 3.3 shows, stemming has a very important and positive effect on the effectiveness of retrieval; users, however, can not deal with the short stems generated sometimes, so little stemming should be done to phrases meant to be read by users.

Chapter 4

Browsing in a Cluster Hierarchy

A secondary goal of this thesis is to evaluate some database organizations that can retrieve new relevant documents by using a set of known relevant documents. This process, called *browsing* from now on, will be compared to relevance feedback, which is another technique using relevant documents previously retrieved. In contrast to relevance feedback, the browsing process studied here does not modify the query; new documents are found because of their relation and similarity with other documents.

This chapter presents an algorithm to look systematically around known relevant documents in search of new relevant documents. The underlying structure supporting the navigation of the document collection is a *cluster hierarchy.* Crouch [Cro89] presents a system suitable for interactive browsing of cluster hierarchies.

Cluster generation and retrieval are explained in the first sections of this chapter, followed by a description of the browsing algorithm to be used in the experimental runs. The chapter ends with an analysis of the results and a discussion of the merits of this methodology.

4.1 Document Clustering

In cluster retrieval, similar documents are grouped together so that a query is only matched against a specially-built representative of the cluster instead of being compared to each document in the cluster. If the similarity between the query and the cluster representative is high enough, then the cluster is inspected in more detail. Otherwise, the cluster and all its documents are ignored. Cluster retrieval was introduced to improve search efficiency compared with a sequential search since full inspection of all the documents of the collection is avoided [Sal71b]. For this effectiveness improvement to occur, the cluster hypothesis which establishes that similar documents are relevant to the same queries, must be valid. In other words, if a relevant document is found in a given cluster then some of the other documents of that cluster must also be relevant.

Cluster analysis has been used in many sciences: biology, astronomy, medicine, computer science and geology, to name a few. As a consequence, the literature on cluster analysis is very extensive. (See [Gor81,Spa80,Har75, SS73] for general uses of cluster analysis.) However, many of the techniques available in other applications are not useful in information retrieval because of the large dimensionality of the document space. The dimensionality of the document space in information retrieval corresponds to the number of terms used in the collection which is often very large.

Clusters studied in information retrieval are often created by *agglomerative hierarchical* algorithms. They are called *agglomerative* since they work by building up clusters from smaller clusters and documents. Unlike divisive algorithms, which start with a single cluster containing all the documents and proceed to subdivide it into smaller units [Wil88], the agglomerative hierarchical cluster algorithms build a hierarchy of clusters starting with single documents which are then merged into increasingly larger clusters until a single cluster remains containing all the documents. There is no overlap between clusters generated using this algorithm, no document is in two clusters that are not hierarchically related: one totally containing the other. The general algorithm for agglomerative hierarchical clustering is fairly simple:

- Put each document in a separate cluster. Compute the inter-cluster similarities.
- Repeat until no merge is possible:
  - Merge the pair of clusters with the highest similarity.
  - Update the similarities between the new cluster and the other clusters.

Different agglomerative clustering algorithms are derived from the basic algorithm by using different inter-cluster similarities. Three of them will be mentioned here: *single link*, because of its historical importance; *complete link*, due to its good performance; and *average group link*, because of its intermediary state between single and complete links.

For *single link*, the similarity between two clusters is the maximum similarity between any pair of documents with one document coming each cluster.

Single link was the first clustering algorithm to be introduced for document retrieval, [Sal71b], [JvR71], and it has been studied extensively: [JP73],[GR69], [Cro77], [Voo86b], [Sib73].

Figure 4.1 shows how the agglomerative hierarchical clustering algorithm works for the single link case. The similarity graph gives the non-zero similarities between seven different documents labelled A through F. The column titled *Merges* specifies the two elements being merged at each stage. In part *a)*, documents D and E merge because they have the greatest similarity, creating cluster 1. Part *b)* shows the similarity graph updated to take into account the creation of cluster 1. Note that the similarity between cluster 1 and document C is .4 because that is the largest similarity between C and any of the components of cluster 1, in this case document E. The similarity between F and cluster 1 is computed in the same way. The merging process continues in parts *c)* though *f)* until no new merge is possible. The resulting hierarchy is displayed in part *(a)* of figure 4.2.

Single link clustering has been criticized for its tendency to chain documents: merges usually involve a single document merged into a big cluster. This can be seen in part *(a)* of Figure 4.2, where documents C and F form a chain with cluster 1. This behavior is produced because the similarity of a given cluster with the remaining clusters can only increase when that cluster gets bigger. [LW67] called this property *space contraction.* For this reason, even though this type of cluster hierarchy is easy to create, single link is not regarded as a good choice for retrieval.

Complete link is an agglomerative hierarchical clustering algorithm that

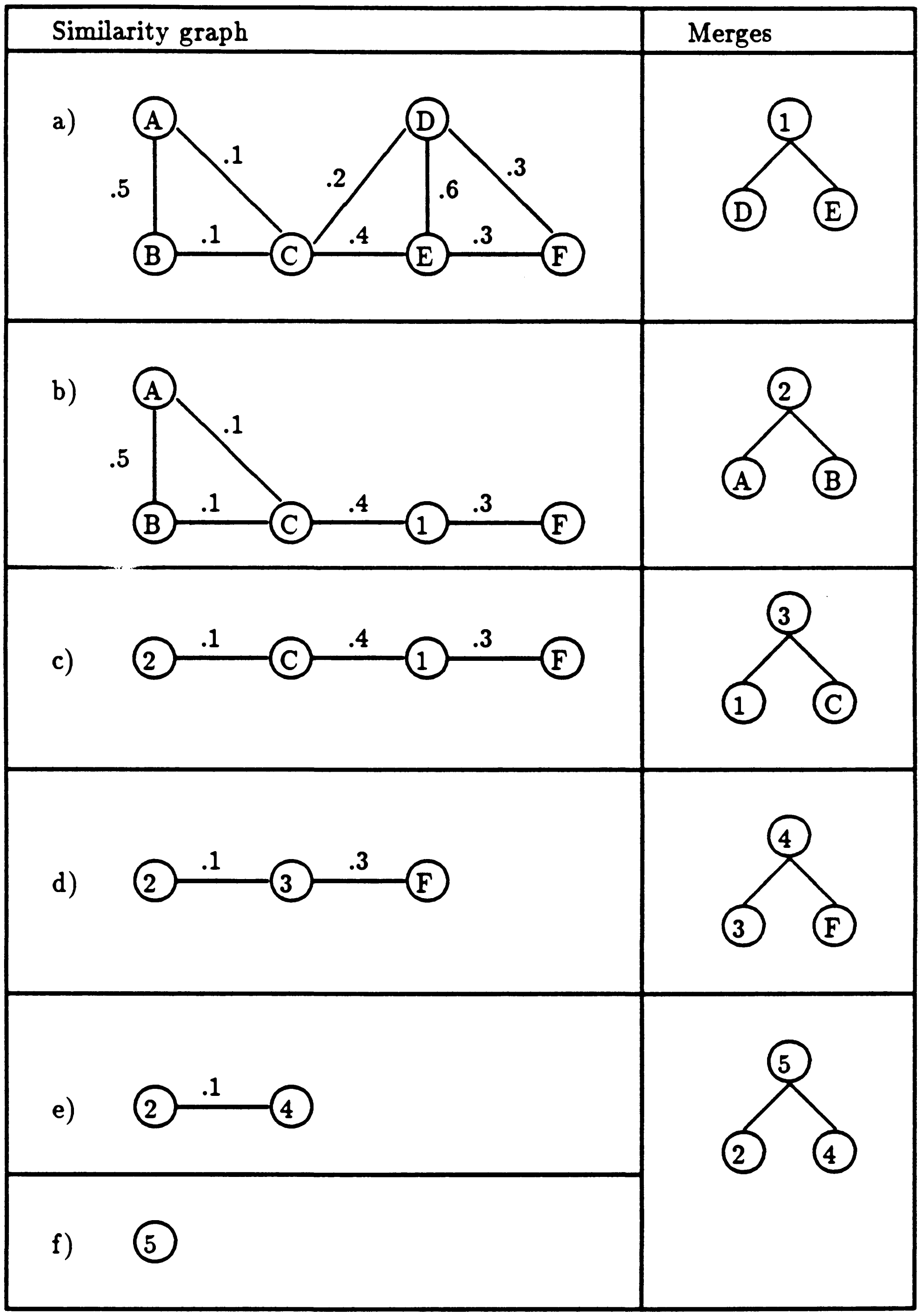


Figure 4.1: Single link clustering

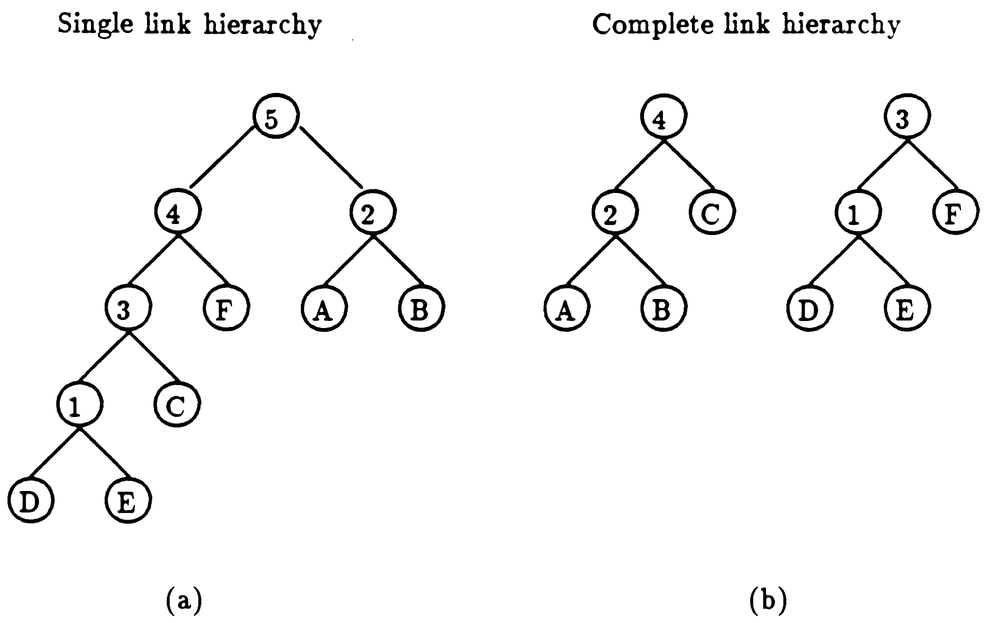


Figure 4.2: Two cluster hierarchies

defines the inter-cluster similarity as the minimum of the similarity between any document of one cluster and any document of the other cluster. This is a very strict condition for cluster merging since two clusters can not merge if they contain documents whose similarity is zero. As a consequence, complete link tends to create many clusters that have to be arbitrarily merged at the top of the hierarchy. These clusters are called *top level clusters.*

Figure 4.3 illustrates the complete link clustering algorithm. The process starts with the same similarity graph previously used for the single link process in figure 4.1. The first merge is also the same: documents D and E merge to create cluster 1. Starting in part *b)*, however, the processes begin to diverge. The similarity between C and cluster 1 in figure 4.3 is .2, since that is the *minimum* similarity between C and the components of cluster 1. On the other hand, the corresponding similarity for the single link case is .4,

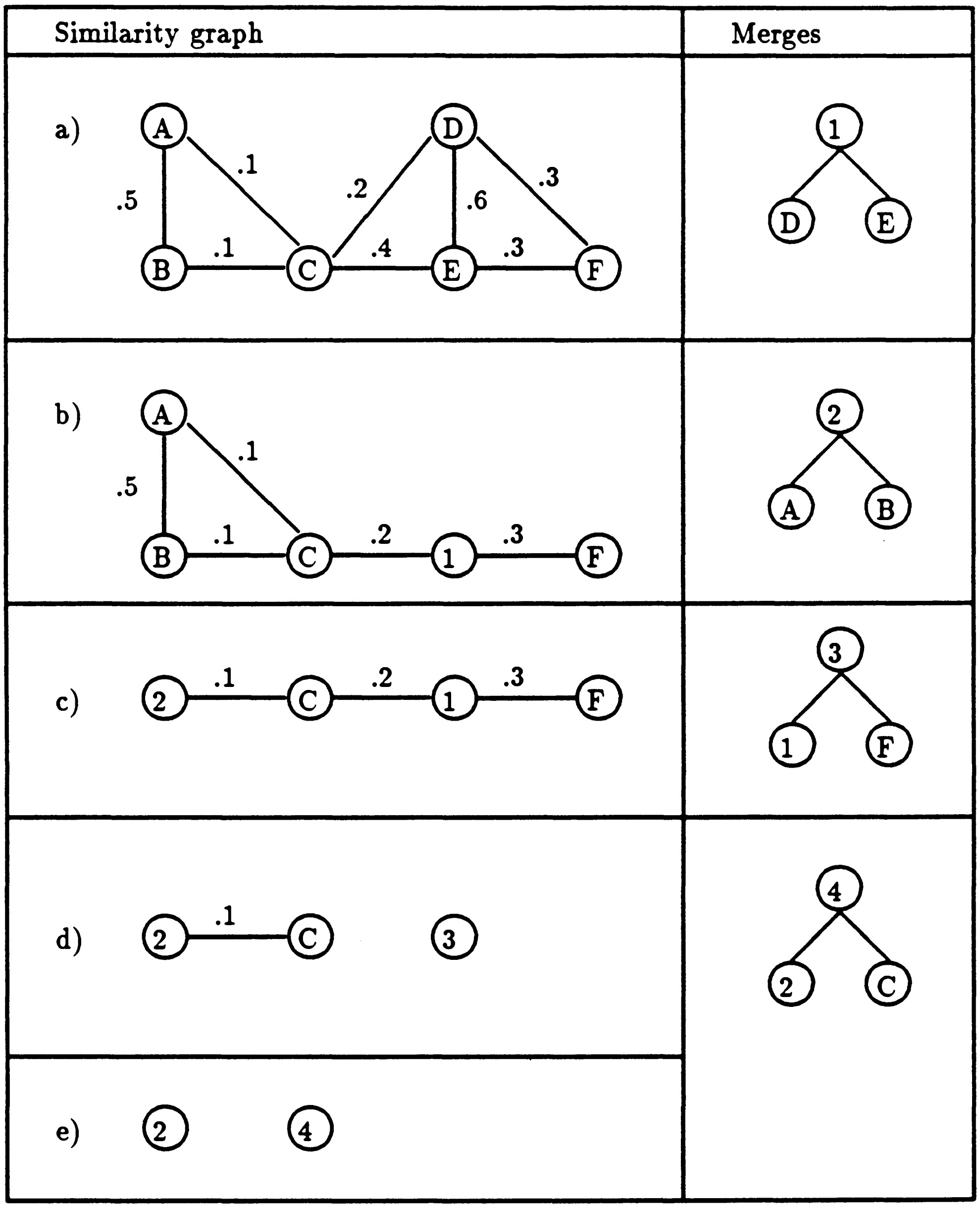


Figure 4.3: Complete link clustering

since it is the *maximum* of the similarities between C and the components of cluster 1. Because of that difference, the merge of part *c)* occurs between cluster 1 and document F for the complete link, instead of cluster 1 and document C for the single link. Note that cluster 3 in part *d)* has no link with document C because that cluster contains F, which has zero similarity with C. Part *(b)* of Figure 4.2 shows the final complete link hierarchy. As the figure illustrates, there are unmerged clusters; these top level clusters are arbitrarily merged to create a root representing the entire set of documents.

Several authors, [Voo85b,Wil88], consider complete link to be the most effective clustering algorithm for information retrieval. Voorhees [Voo85c], however, found that a complete link process is not as efficient as an inverted file retrieval method. Araya [AM90] also found that for a much larger collection even though some auxiliary inverted file was included to speed up the cluster search.

Given the considerable effort required by the complete link algorithm, some alternative use of the cluster hierarchy is needed to justify this effort. For this reason the possibility of using the cluster hierarchy for browsing runs is attractive.

The third and final agglomerative hierarchical clustering algorithm to be mention in this section is the *average group link.* For this algorithm, the inter-cluster similarity used is the average of the similarity between any document of one cluster and any document of the other cluster. This inter-cluster similarity is considered intermediary between the relaxed similarity of the single link and the strict similarity of complete link.

4.2 Cluster Retrieval

Many search algorithms are possible using cluster hierarchies. Basically, there are two classes of search algorithms in a cluster hierarchy: the top-down algorithms start at the root of the hierarchy and traverse the tree down top to bottom following one or more paths determined by the similarity between the query and the cluster representatives. The bottom-up algorithms start by computing the similarity between the query and the low-level clusters (clusters directly containing documents) and proceed to continue the search in those clusters with the largest similarity with the query. Cluster retrieval is not exhaustive; that is, not all the documents with non-zero similarities with the query are encountered. Therefore, there is no guarantee of retrieving the documents with the highest similarities.

Because the browsing algorithm to be studied in this chapter is a natural generalization of the *top down individual cluster search* this particular cluster search is described in more detail. Furthermore, previous work, [Voo86b], has shown that top-down individual cluster search is more effective than inverted file search, although it is not as efficient [Voo86a]. This type of top-down search follows many paths simultaneously and when a bottom cluster is encountered, each of its documents are compared against the query. The comparison of individual documents with the query allows the creation of a ranking of documents.

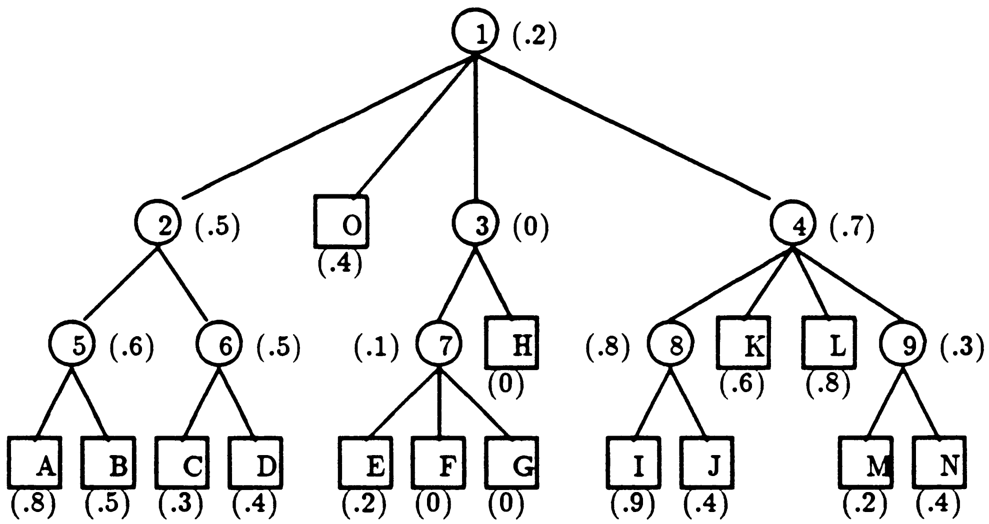
For retrieval purposes, a cluster representative called a *centroid* is computed for all the clusters of the hierarchy. These representatives are term vectors constructed by combining the term vectors of the documents included

- Put the root in the **heap**.
- While the **heap** is not empty and more documents are wanted do:
  - Take the element of the **heap** with the largest similarity; call it **top**.
  - if **top** is a document then include it in the set of retrieved documents
  - if **top** is a cluster then put all its children into the **heap** after their similarities with the query have been computed.

Figure 4.4: Top down individual cluster search algorithm

in the cluster these centroids represent. Therefore, the similarity between a cluster and the query is computed by determining the similarity between the cluster centroid and the query. There are many ways to create the centroids from the document vectors. This thesis uses centroids created following the instructions detailed in [Voo85c].

The top-down search is controlled by the number of documents the user wants. A greater effectiveness will result from retrieving a larger number of documents but the efficiency will suffer because a more extensive area of the hierarchy will have to be inspected. A heap data structure is used to contain the elements of the hierarchy, documents or clusters, to be inspected. While Figure 4.4 contains a description of the top down individual cluster search, Figure 4.2 illustrates an example of a search using the top down individual cluster search. The circles in that figure represents clusters with more than one document, while the squares represent the clusters with a single document. The numbers in parenthesis close to the circles and squares are the similarity values between the centroids or documents and the query.



1. pop [1,.2]
2. add [2,.5], [4,.7] and [O,.4]
3. pop [4,.7]
4. add [8,.8], [9,.3],[L,.8] and [K,.6]
5. pop [L,.8]
6. pop [8,.8]
7. add [I,.9] and [J,.4]
8. pop [I,.9]
9. pop [K,.6]

Retrieved documents: [I,.9],[L,.8] and [K,.6].

Figure 4.5: Example of top down individual cluster search

The notation [**node, sim.**] is used to refer to a node and its query similarity. Three documents are retrieved after the nine operations shown. The *pop* operation extracts from the heap the element with the highest similarity which is shown to the right; the *add* operations put the element listed into the heap.

As mentioned before, the cluster hierarchies created by the complete link clustering algorithm have a large number of top level clusters, caused by the strict merging criterion of complete link. To merge two clusters, the number of non-zero similarities needed is equal to the product of the sizes of the clusters, so when two clusters are large the number of non-zero similarities needed to merge them is very large.

One consequence of the large number of top level centroids is that the first iteration of the top down individual algorithm degrades into a sequential search on a large number of clusters. Obviously, once the collection reaches a certain size this sequential search is too slow. Some manipulations of the cluster hierarchy have been studied to overcome this problem, see [AM90]. Among the manipulations presented there, two upward extensions of the complete link hierarchy were defined. In the first extension, the hierarchy is extended upwards by applying the complete link clustering algorithm to the centroids representing the top level clusters. In this way, if two centroids merge their corresponding clusters also merge. Since two centroids may have non-zero similarities even though their respective clusters do not merge under the complete link algorithm, this technique allows merging two clusters that would not otherwise merge under the complete link inter-cluster similarity

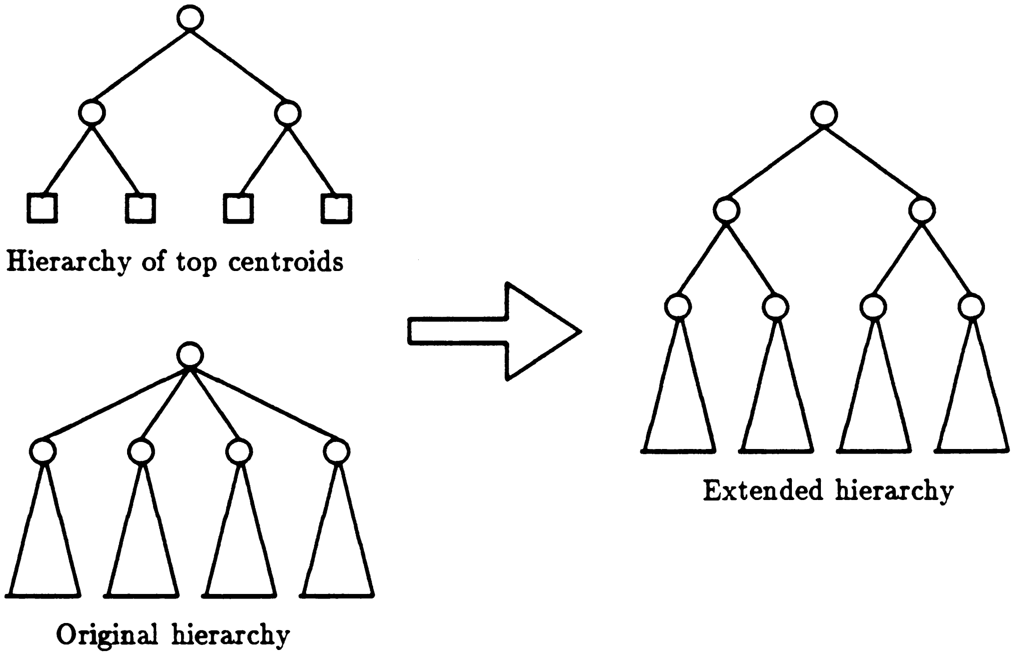


Figure 4.6: Cluster hierarchy extension

measure.

In the second extension the hierarchy is extended upwards by applying the group average link algorithm. Because this inter-cluster similarity is less strict that the one used by complete link, the top level clusters can continue merging beyond the complete link limit. Figure 4.2 illustrates the way in which the hierarchies are extended.

4.3 Relevance Feedback using Inverted File

Because browsing methodologies use relevant documents retrieved in previous runs, it is natural to compare their performance against relevance feedback techniques. As explained in section 1.3.4, the relevance feedback process modifies the vector representing the initial query using the vectors representing the retrieved documents, especially the relevant ones.

The feedback queries are run using an inverted file. Given a query, the inverted search can locate documents with the highest similarities with the query without having to inspect all the documents in the collection. In other words, documents with zero similarities to the query are not processed. The inverted file associates each term t of the collection with a list of **<document id, weight>** pairs, telling which documents contain term t and the weight of t in those documents.

The inverted retrieval algorithm used in this thesis is described in [Buc85] as *basic algorithm.* Enough space is needed for an array with as many entries as documents in the collection. This array will contain the partial similarities between the documents and the query. Another array keeps track of the documents with the top similarities found so far. At the beginning of the search all the entries of the partial similarity array are set to zero; then for each query term, its corresponding inverted list is read and for each document in that list its partial similarity its increased by the product of the query weight of the term and the term weight in the inverted list. If the partial similarity of a document is large enough, that document is put into the top similarity array. At the end, after processing all query terms, the partial similarity array will contain the inner product similarities between the query and all the documents, and the top similarity array will contain the documents with the highest similarities.

4.4 Browsing Algorithm

The browsing algorithm to be detailed here is a natural derivation of the top down individual search algorithm presented in section 4.2. It is also an approximation of a reasonable strategy users can use to browse in a hierarchy. Basically the browsing algorithm and the top down individual search algorithm differ in terms where the search begins: the top down search begins at the root; the browsing search begins at the relevant retrieved. They also differ in the way the hierarchy is traversed: in the top down case, the search path always goes down, towards the leaves; in contrast, the search path of the browsing algorithm may go up as parents of previously inspected nodes are inspected themselves.

Browsing is implemented using a heap containing the hierarchy nodes to be inspected. Inside the heap, the elements are sorted by their similarity with the query. At the beginning the heap contains the parents of the relevant documents previously retrieved. After initialization the algorithm takes the element of the heap with the highest similarity. If that element is a document (a leaf of the hierarchy tree) then that document is retrieved. Otherwise the element is a hierarchy node, in that case its children and parent are included into the heap. During the browsing search, the nodes being inspected are marked to avoid processing them again. Figure 4.7 presents the algorithm in more detail. One important consideration is that the algorithm should not reach the root of the hierarchy: in that case, if the root is taken out of the heap, then all the similarities between the query and the top level centroids have to be computed and the algorithm becomes a top down individual search.

- Initialization [all nodes unmarked]
  1. Make **heap** empty.
  2. Mark all nodes corresponding to documents retrieved.
  3. Mark and put in heap the parents of the retrieved relevant documents.
- Browsing
  - While more documents are wanted and **heap** is not empty do:
    1. Take the node of the **heap** with the largest similarity, call it **n**.
    2. If **n** is a document put it in the set of retrieved documents.
    3. If **n** is an internal node then
       * Mark and put into the **heap** all unmarked children of **n**.
       * Mark and put into the **heap** the parent of **n** (unless it is marked or it is the root.)

Figure 4.7: Browsing Algorithm

Figure 4.8 illustrates the browsing algorithm. Three relevant documents have been retrieved: b, d, i, and one non-relevant has been retrieved h. An * marks the nodes visited so far and the nodes in the heap are marked by a square. The figure shows the steps needed to retrieve three new documents.

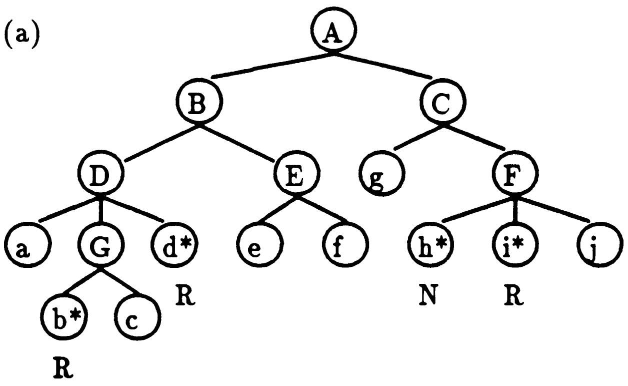
4.5 Collections

Three collections are used in the browsing experiments: CACM, a computer science collection; INSPEC, an electronic engineering collection; and NEWS, a collection built on top of INSPEC by adding several thousand electronic messages on computer science. Table 4.1 details the characteristics of these collections. The NEWS collection uses the same queries as the INSPEC collection and has the same relevance assessments; in other words, none of the electronic messages added is considered relevant to the queries. This is reasonable to assume since most of the articles in the electronic messages are too narrow and specific for the kind of information described in the INSPEC queries.

Some of the queries from each collection were discarded because they do not retrieve relevant documents in their initial runs. Since relevant documents are the starting point for both browsing and relevance feedback then queries not retrieving relevant documents are useless for the experiments.

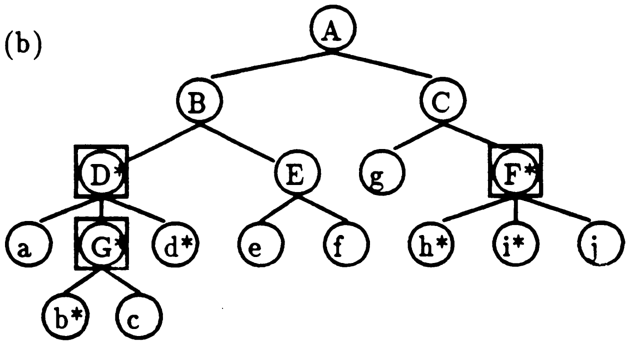
Each collection has three different cluster hierarchies. One of the hierarchies, called *complete hierarchy* in tables 4.3-4.5, was created using the complete link clustering algorithm. The other two are upwards extension of the complete hierarchy; one of the extensions, called *complete extension*, was

(a)



Mark retrieved documents

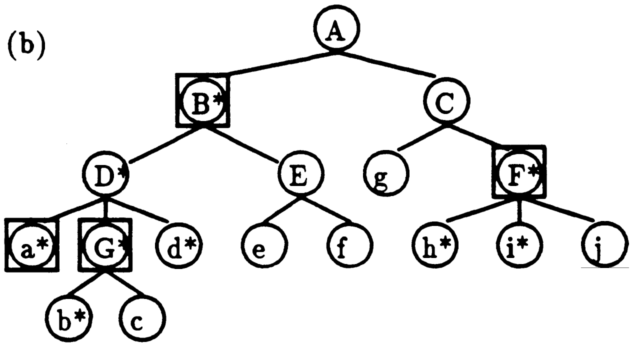
(b)



Mark and put parents in heap.

Heap: D, F, G.

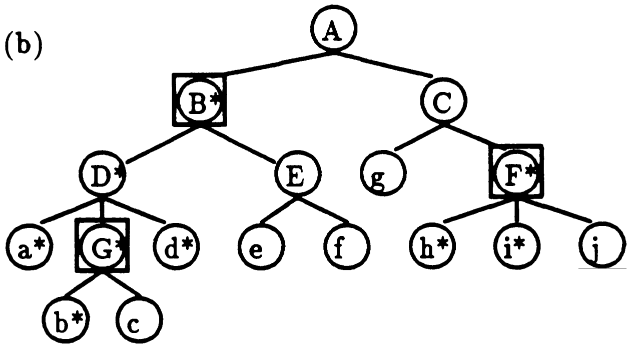
(b)



Top of heap (D) is cluster:
Mark and put in heap:
a, child; and
B!= root, parent
(both unmarked.)

Heap: a, B, F, G.

(b)



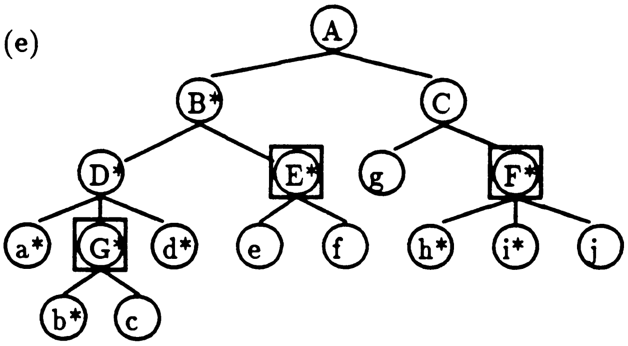
Top of heap (a) is document:
a is retrieved.

Heap: B, F, G.

Figure 4.8: Example of browsing algorithm

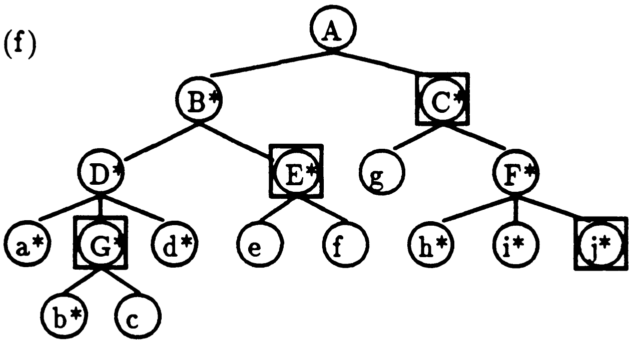
Figure 4.8: (Continued)

(e)



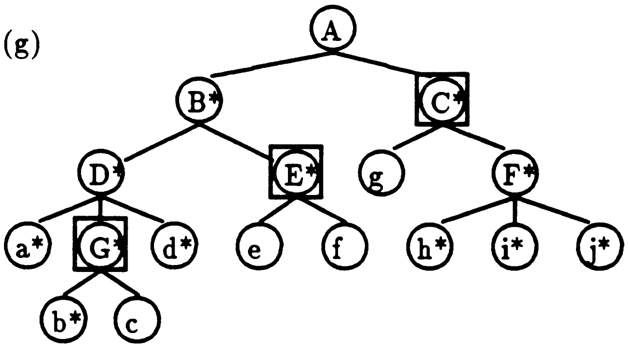
Top of heap (B) is cluster:
Mark and put in heap:
E, child unmarked
A=root, do not add
Heap: F, G, E.

(f)



Top of heap (F) is cluster:
Mark and put in heap:
j, child unmarked
C, parent, not root
Heap: j, G, C, E.

(g)



Top of heap (j) is document:
j is retrieved.

Heap: G, C, E.

Table 4.1: Collection Characteristics

	CACM	INSPEC	NEWS
number docs	3204	12684	54115
number terms	8503	14573	97080
mean terms per doc	22.5	32.5	42.9
number queries	44	74	66
mean rel. docs per qry	17.2	32.9	34.3
total rel. docs	755	2433	2263
exhaustive rel. assessments	no	no	no

created by applying the complete link clustering algorithm to the centroids of the top level clusters of *complete hierarchy*; the other extension, called *average extension* was created by applying the average link clustering algorithm to the top level clusters of *complete hierarchy.* The characteristics of these hierarchies can be seen in table 4.2. As this table shows, the NEWS hierarchies are four times bigger than the INSPEC hierarchies, which are themselves four times bigger than the CACM hierarchies. This progression in size is intended to give insight into the behavior of the algorithms as the collections grow.

4.6 Analysis of Results

Several experimental runs were performed to compare the browsing algorithm to relevance feedback using an inverted file. The results of these experiments are shown in Tables 4.3-4.5. The first column of these tables presents the precision obtained running the original query again without any changes; the

Table 4.2: Characteristics of the hierarchies created

		Complete Hierarchy	Complete Extension	Average Extension
CACM	leaves	3204	3204	3204
	top level	327	45	2
	size top level	9.8	71.2	1602
	internal nodes	2729	3010	3041
	total nodes	5933	6214	6245
INSPEC	leaves	12684	12684	12684
	top level	821	79	2
	size top level	15.4	160.6	6342
	internal nodes	11858	12598	12642
	total nodes	24542	25282	25326
NEWS	leaves	54115	54115	54115
	top level	3300	378	11
	size top level	16.4	143.2	4920
	internal nodes	50392	53311	53527
	total nodes	104507	107426	107642

second column shows the precision for relevance feedback; and the last three columns present the precision obtained for simulated browsing in the complete link hierarchy and its two extensions. The precision figure given is the precision after retrieving 15 documents using the reduced collection technique for evaluating feedback runs. Under *# Rel. Ret.* the total number of relevant documents found for all queries is given; on the other hand, *# q.* gives the number of queries failing to retrieve any new relevant documents. The next four lines are efficiency data averaged by query. These lines contain the cpu time, the number of input/output operations the elapsed time and a normalized time. The normalized time, used because many different conditions may affect the elapsed time, is equal to cpu time + i/o ops * 0.030.

One observation must be made about the type of computers on which the experiments were performed. The CACM collection runs were processed in a Microvax computer, while the INSPEC and the NEWS news collection runs were processed in a SUN-4 computer. This difference in the computers used makes inappropriate a direct comparison between the efficiency in the CACM collection and the efficiency in the INSPEC and NEWS collection. However, the efficiency of the different methods can be compared within the same collection, and the efficiency on larger collections can be inferred by extrapolating the changes noticed between the INSPEC and NEWS collections, since these two collections were run in the same environment.

Table 4.3 shows the results for the CACM collection. In this collection cluster browsing perform much better than relevance feedback. More specifically, the precision obtained by cluster browsing in the extended hierarchies is

Table 4.3: Browsing Algorithm for CACM

	Inverted no change	Inverted feedback	Complete hierarchy	Complete extension	Average extension
Precision	.123	.166	.143	.195	.240
# Rel. Ret.	116	130	92	126	155
# q.	9	8	14	9	4
CPU Time	.2	1.1	.6	.9	1.4
I/O Ops.	15	127	21	41	83
Elapsed t	.5	2.5	1.2	1.6	3.0
Normalized t	.7	4.9	1.2	2.1	3.9

17% and 44% better than the precision obtained by relevance feedback (.195, .240 for extended hierarchies, .166 for feedback.) Furthermore, the average extended hierarchy has fewer queries not retrieving relevant documents (4 for average extension vs. 9 for feedback.) The normalized time of the cluster alternatives is smaller than the normalized time used by feedback. This is reasonable given the fact that the feedback queries are much larger than the original queries, so that inverted processing is more demanding in the case of feedback queries. For CACM the extensions on the complete hierarchy improved the performance of cluster browsing; the complete and the average extensions are respectively 36% and 68% more effective than the basic complete hierarchy. Running the original query without change, is 26running the relevance feedback query, although it is faster.

In the INSPEC collection the situation changes. Table 4.4 indicates that

Table 4.4: Browsing Algorithm for INSPEC

	Inverted no change	Inverted feedback	Complete hierarchy	Complete extension	Average extension
Precision	.203	.241	.192	.206	.224
# Rel. Ret.	280	340	213	229	249
# q.	8	6	20	17	14
CPU Time	.1	.7	.2	.3	.4
I/O Ops.	6	33	20	28	35
Elapsed t	.5	3.3	.5	.7	.9
Normalized t	.3	1.7	.8	1.1	1.5

normal relevance feedback is more effective: the average extension, which has the best performance, has a precision 7% lower than the feedback process. The clusters also perform poorly in the number of queries retrieving no relevant documents: the best of them (average extension) has 14 queries retrieving no relevant documents compared with only 6 for feedback. For INSPEC the extended hierarchies also performed better than the basic complete hierarchy, but the improvements were smaller: 7% for the complete extension and 17% for the average extension. Even though the cluster alternatives are faster than feedback, the most effective cluster, average extension, is only marginally faster than feedback. Running the query again with no changes, first column of Table 4.4, is very fast and much closer in precision to the cluster runs than in the CACM case; it even has better precision than browsing using the basic complete link cluster.

The largest collection, NEWS, produces similar results to those found in INSPEC. Table 4.5 shows that feedback is the most effective technique again. In this case the best cluster alternative is the complete extension, but it is only marginally (3%) better than the average extension and the precision is 4% and 6% lower than feedback. The situation with the queries retrieving no relevant documents is the same as in INSPEC; all the cluster alternatives seem to have many more queries failing to retrieve at least one relevant document. Following the trend noted in INSPEC, the hierarchy extensions performed better than the basic complete hierarchy, but their advantage was reduced to 7% for the complete extension and 4% for the average extension. On the efficiency side, the cluster alternatives continue to be faster than feedback, 1.3 for complete extension vs. 6.8 for feedback. The inverted without change is not as fast as in INSPEC and its precision deteriorates more than the cluster runs.

Summarizing the results for the browsing algorithm, it seems that the main problem encountered by that algorithm in the largest collections is not the drop in precision (7%) which can be justified by the much faster speed of the cluster browsing, but the large number of queries not retrieving relevant documents.

4.7 Distribution of Relevant Documents

The results presented in the previous section leave several questions open. It is not clear for example, why browsing is much more effective than relevance feedback for the CACM collection while not as effective for the other two

Table 4.5: Browsing Algorithm for NEWS

	Inverted no change	Inverted feedback	Complete hierarchy	Complete extension	Average extension
Precision	.168	.203	.183	.196	.191
# Rel. Ret.	214	257	181	194	189
# q.	11	8	20	18	14
CPU Time	.2	1.2	.2	.3	.3
I/O Ops.	24	182	25	33	41
Elapsed t	.5	3.3	.6	.7	.9
Normalized t	.9	6.8	1.0	1.3	1.5

collections. In order to understand the performance of the browsing search, this section studies the distribution of the relevant documents in the cluster hierarchies. This distribution is important because the browsing search will not retrieve relevant documents which happen to be in a top level cluster which has no known relevant documents. As mentioned in section 4.4, the browsing algorithm avoids reaching the root since that will imply a full top down search. One consequence of this restriction is that the browsing search does not spread from one top level cluster to another one because in order to do that the root must be reached. Therefore, if many relevant documents are included in top level clusters with no known relevant documents then no new relevant documents will be retrieved and performance will suffer.

4.7.1 Cluster Browsing

The analysis of the distribution of the relevant documents in the cluster hierarchies is done by identifying the following sets of relevant documents:

Unreachable: relevant documents in top level clusters without known relevant documents. New relevant documents can not be retrieved by the browsing search since that search has to start at some node inside the top level cluster.

Known: relevant documents retrieved in a previous run.

Reachable: relevant documents in top level clusters with at least one known relevant document. These documents can be retrieved since the browsing search has a node where to start.

Retrieved: relevant documents in the *reachable* set actually found by the browsing search.

Table 4.6 shows the percentage of relevant documents in the *unreachable* set for the three collections and the three cluster hierarchies. As can be seen from that table, a very important percentage of the relevant documents is out of reach for the complete hierarchy and the complete extension hierarchy in all the three collections. On the contrary, in the average extension hierarchy all the relevant documents are reachable because there are very few top level clusters. Table 4.2 shows that the average extension hierarchy only has two top level clusters in the CACM and INSPEC collections, while there are only eleven top level clusters in the NEWS collection. The fact that all the relevant documents are reachable using the average extension hierarchy, explains why

Table 4.6: Percentage of relevant documents unreachable by the browsing algorithm

Collection	Complete hierarchy	Complete extension	Average extension
CACM	59.9	47.6	0.0
INSPEC	65.6	50.8	0.0
NEWS	70.3	61.5	0.1

in general browsing is more effective in that type of hierarchy in the complete hierarchy or the complete extension hierarchy.

The percentage of relevant documents which are reachable is presented in Table 4.7. In that table, for each collection and each of hierarchy three percentages are given. The *Known* line gives the percentage of relevant documents retrieved before the browsing search; the *Reachable* line shows the percentage of relevant documents that are not known and can be retrieved by the browsing search; finally, the *Retrieved* line presents the percentage of new relevant documents actually retrieved. The success of the browsing algorithm to find new relevant documents depends on the collection searched and the hierarchy used. Overall, for CACM, browsing has a very good performance in the complete hierarchy and the complete extension hierarchys. In the complete hierarchy, 13.0% of the relevant documents were retrieved by the browsing search from a maximum of 15.1% which are reachable. For the complete extension, 17.4% was retrieved from a maximum of 27.3%. The

Table 4.7: Percentage distribution of relevant documents

Collection		Complete hierarchy	Complete extension	Average extension
CACM	Known	25.0	25.0	25.0
	Reachable	15.1	27.3	74.9
	Retrieved	13.0	17.4	21.2
INSPEC	Known	20.8	20.8	20.8
	Reachable	13.6	28.4	79.2
	Retrieved	8.4	9.6	10.7
NEWS	Known	17.1	17.1	17.1
	Reachable	12.6	21.4	82.8
	Retrieved	7.7	8.2	8.3

other two collections, however, do not show as good a performance as CACM for the same two hierarchies.

Because all documents are reachable in the average extension hierarchy, the analysis of the performance in that hierarchy should take into account that only 15 documents are retrieved, limiting the number of reachable relevant documents actually retrieved. For that reason, the performance of the browsing search in the CACM collection for the average extension should be considered good: 21.2% of the relevant documents were retrieved from a maximum of 74.9% which are reachable. On the other hand, the other two collections are less successful than CACM. For these collections the percentage of retrieved relevant documents change little from the complete extension to the average extension even though the reachable set is much bigger in the second hierarchy.

The analysis of Tables 4.6 and 4.7 show that larger clusters help retrieval because they provide a larger set of reachable relevant documents, but at the same time, the browsing search is more successful at retrieving relevant documents in smaller clusters.

4.7.2 Relevance Feedback

Relevance feedback, unlike cluster browsing, does not have the limitation of retrieving relevant documents only from the reachable set. In principle, relevance feedback can retrieve any document. Table 4.8 shows, however, that relevance feedback is not very successful at retrieving relevant documents from the unreachable set. The *browsing* line of this table gives the percentage of relevant documents that are not reachable by the browsing algorithm,

Table 4.8: Distribution of retrieved relevant for feedback

Collection		Complete hierarchy		Complete extension		Average extension	
		UnRch.	Rch.	UnRch.	Rch.	UnRch.	Rch.
CACM	browsing	59.9	15.1	47.6	27.3	0.0	74.9
	feedback	6.8	5.3	5.5	6.6	0.0	12.1
INSPEC	browsing	65.6	13.6	50.8	28.4	0.0	79.2
	feedback	5.7	5.8	4.2	7.3	0.0	11.5
NEWS	browsing	70.3	12.6	61.5	21.4	0.0	82.8
	feedback	4.0	5.4	3.2	6.1	0.0	9.3

and the percentage of relevant documents retrieved by that algorithm. The *feedback* line gives the percentage of relevant documents retrieved by relevance feedback for the unreachable set and the reachable set. As can be seen in Table 4.8, relevance feedback does not retrieve a large number of unreachable relevant documents. This is an interesting result since it suggests that the characteristics of many of the unreachable documents make them different from the known relevant documents which are the base of the browsing search and the relevance feedback search.

4.7.3 Cluster Hypothesis

The good results obtained by the cluster browsing in CACM can be explained by several characteristics of the collections detailed in [Voo85b,Voo85a]. There Voorhees introduces a test to check the validity of the cluster hypothesis

Table 4.9: Percentage of relevant documents among nearest neighbors of relevant documents

	CACM	INSPEC	NEWS
0	43	64	69
1	26	25	22
2	16	8	8
3	10	2	1
4	3	1	0
5	3	0	0

for the different collections. For each relevant document, its most similar documents, *nearest neighbors*, are obtained and the percentage of relevant documents among the nearest neighbors is computed. The larger the percentage of relevant nearest neighbors, the more likely the cluster hypothesis is valid for a particular collection. Table 4.9 presents the percentage of relevant documents among the five nearest neighbors of relevant documents. The distribution shown in that table differs from the distribution given by [Voo85b] because some queries were removed due to their unsuitability for feedback. As the table shows, for CACM, 57% of the relevant documents have at least one relevant document among their five nearest neighbors. On the other hand, only 36% of the relevant documents of INSPEC and 31% of the relevant documents of NEWS have at least one relevant document among their five nearest neighbors. The larger percentage of relevant nearest neighbors of the CACM collection makes it reasonable to expect that browsing in CACM is more likely to produce better results than in the other collections.

4.8 Conclusions

In general, the effectiveness of cluster browsing in a given collection depends more on the validity of the cluster hypothesis than on the size of the collection. As illustration, note that cluster browsing was able to perform well in the average extension hierarchy for the CACM collection, even though the top level clusters of that hierarchy are quite big.

The extended hierarchies performed better than the basic hierarchy in all collections. As the collection grows, however, the advantage of the extended hierarchies over the basic hierarchy becomes smaller. This behavior is caused by the competition of two factors. First, the extended hierarchies have larger clusters making it possible to the browsing algorithm to search a larger portion of the collection. However, the larger the clusters, the less effective is the browsing algorithm.

Chapter 5

Browsing in a Network

The previous chapter showed how a cluster hierarchy can be used to search around known relevant documents in order to find new relevant documents. This chapter also deals with the detection of new relevant documents but in the context of an information retrieval network. Several authors [Wil88, CP85,GLW86] advocate the use of networks in information retrieval. In such networks, several types of *nodes* are interconnected using different types of *links*, so that there is a rich set of relationships between documents and other entities of the system. This chapter evaluates how effectively such an organization can be used to search for new relevant documents based on the information provided by known relevant documents. The algorithm presented in this chapter systematically searches a network starting from nodes representing known relevant documents. Because the algorithm approximates to a certain point the browsing activities of users trying to find new relevant documents, some conclusions may be obtained about the usefulness of the network organization in an interactive information retrieval environment.

5.1 Network Organization

Basically, networks include three components: *nodes*, *links* and *attributes*. The nodes store the information on the basic entities of the network. Relationships between these nodes are represented by links. The specific items of information available on nodes and links are called attributes. One of these attributes is the *type* of the node or the link; for example, a node may have type *document* or type *term*, while a link may have type *citation* or type *associated term*. The other attributes of a node or link depend on the type of that node or link. Some links, for example, may have a *strength* value associated with them, while others may lack that value.

The network organization has been employed in many different areas in computer science. In databases, the network organization forms the basis for the network database model. The concepts of node and link are also the fundamental characteristics of hypertext [Con87,Hal88], where users search the network by following the links connecting a piece of text stored in one node with nodes containing related information. Artificial Intelligence [CM86] uses semantic networks for knowledge representation in areas like natural language processing.

In information retrieval, networks offer many advantages:

- Easy access to all the information available about a specific item. That is, given an item of interest, all the information available for that item is readily available by following links (provided that there are links leading to that information.)

- Additional sources of information can be easily incorporated. For example, a network representing a thesaurus can be independently created from a network representing a collection of documents. These two networks can be combined easily into a network that provides query expansion mechanisms using the thesaurus relationships.

- Easy browsing. The presence of links enables a non-systematic search of the collection. In other words, a path of nodes can be followed without having to examine all the intermediate alternatives.

There are some disadvantages, however, to the use of networks for the representation of information.

- Some important systematic search methodologies may not perform efficiently. For example, processing a query using inverted retrieval requires the use of inverted lists for each query term; which may have to be constructed from the term-document links.

- For large networks user browsing is difficult due to the large number of nodes and links.

5.2 Types of Nodes and Links

A number of different types of nodes and links are normally found in information retrieval networks. *Document nodes* represent the documents of a collection. The specific attributes of a document node depend on the particular organization of the network. Information, which is represented in some networks by node attributes, may be represented by individual nodes in other networks. For example, author information can be stored either as an attribute of a document node or as a node of type *author*.

Another important type of node is the *term node*. These nodes represent the indexing units (terms). Some organizations also include *concept nodes* representing higher level notions associated with terms. In those systems, the term-concept links gives the probability that the presence of a term implies a given concept. For example, in a particular network, the presence of term *bat* may imply the concept baseball with probability of 0.6; while the same term implies the concept *dracula* with probability of only 0.1. In addition to these three types of nodes, other nodes may be present to represent other items of information like author, publisher, journal, etc.

Several types of links are provided to relate the nodes described above.

document-term: these links associate a document node with one of the term nodes representing the content of the document. The strength of the link corresponds to the weight of that term in the given document.

term-document: these links associate a term node with one of the document nodes containing that term. As above, the strength of the link corresponds to the weight of the given term in that document.

term-term: these links associate terms in different ways. Some links may be based on thesaurus relationships like broader term, narrower term and synonym, while other links relate terms according to their cooccurrence frequency.

document-document: these links associate document nodes according to different relationships. Some links may be based on citation relationships like *cited by* or *cite*. Other links are based on the similarity between the two documents defining the link.

5.3 Search Methodologies in a Network

Many search methodologies are possible in a network. Some of the search methodologies are better adjusted than others to the network structure. This section reviews how these methodologies can be implemented in a network. The presentation emphasizes *spreading activation* because this method naturally uses the node-link organization of the network.

5.3.1 Inverted Retrieval in a Network

The inverted search organization was presented in section 4.3. In the context of a network organization, inverted lists may be formed by different term-document links. The way these links are implemented has a direct effect on the efficiency of the inverted search operations. An implementation of term-document links which stores all the links for a given term in an array, can efficiently perform the inverted list operations. Otherwise this array must be created dynamically as is done in [CP85]. The dynamic creation of inverted lists, however, can be very inefficient for large lists.

5.3.2 Cluster Retrieval in a Network

The document-document links based on interdocument similarities define an overlapping cluster organization where each document is in the center of a cluster formed by that document and all the documents connected to it by document-document links. Croft and Parenty [CP85] perform cluster retrieval by computing the centroids of all the clusters corresponding to documents with terms in common with the query. After the centroids have been computed, the clusters whose centroids have the largest similarity with the query are retrieved. Depending on the different cluster retrieval methodology used, either all the documents in the retrieved clusters are retrieved or the documents are compared to the query and only the most similar to the query are retrieved.

More specifically, the cluster retrieval process first takes the query terms and using the term-document links finds all the documents with at least one term in common with the query. Then, for each document found, the document-document links are used to find related documents and create the centroid of the cluster defined by those related documents. Once each centroid is created, the query-centroid similarity is computed, and the system keeps track of clusters whose centroids have the best similarities.

5.3.3 Oddy's Browsing Search

This exploratory type of search was introduced in section 1.4.2. The search is done in one region of the network defined by certain term nodes plus the documents associated with them by term-document links. The system ranks the documents inside the search region and the best ranked document and their terms are presented to the user so that he can evaluate the relevancy of that document. The user changes the search region by adding new terms to the region or deleting old ones. Whenever the search region is changed, the system recomputes the ranks of all the documents inside the search region.

Because of its heavy interaction with the user, this kind of search is not feasible for a systematic search of a large network. Instead the author [Odd77] claims that the main advantage of this approach is that it can retrieve documents even though the user does not have a clear idea of what he is looking for. The iterative nature of the search also helps the user to clarify his goals before a systematic search is performed.

5.3.4 Spreading Activation

Spreading activation is a search methodology based on the idea of a weight propagating from node to node. The nodes encountered during this process have a value called the *activation level* assigned to them. This activation level is then used to determine whether or not that node is worth inspecting in more detail. Several factors are usually combined to compute the activation level of a node: the activation weight being distributed, the strength of the link being used, and the distance from the nodes where the activation starts.

Figure 5.1 illustrates the spreading activation process. As part a) of that figure shows, the process starts in the lower left node, **A**, which is assigned an initial activation level, 80. In part b), the activation level is passed on to the neighbors of **A**: **B**, **D** and **E**. The amount of activation each node receives can be computed in many ways. In the example of Figure 5.1, the activation level of the newly activated nodes is computed by taking the product of the activation level of the node activating them and the weight of the link. As part b) shows, the activation level of node **B** is the product of the weight of the **A-B** link and the activation level of **A**: $0.2 * 80 = 16$. The activation levels of nodes **D** and **E** are 16 and 32, respectively. Part c) shows what

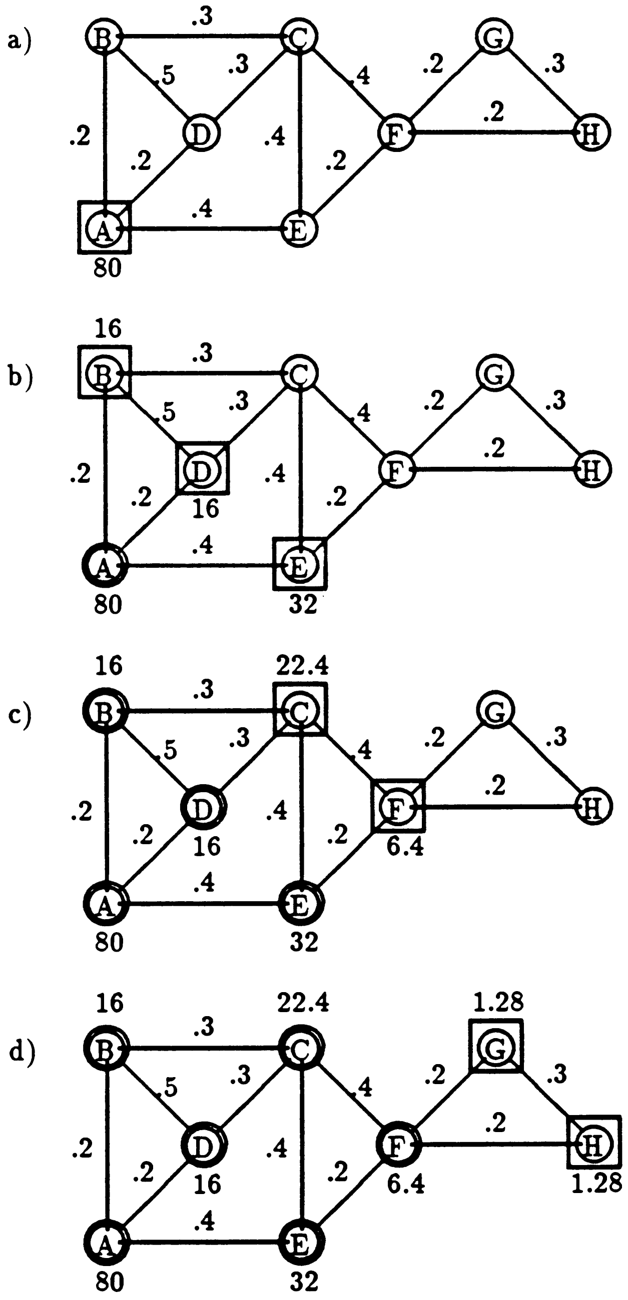


Figure 5.1: Example of spreading activation

happens when a node is activated from different paths. Nodes B, D and E activate node C; they respectively passed activation levels 4.8, 4.8, and 12.8 to C. These three values are added together to obtain the activation level of C, 22.4. Other functions such as MIN or MAX can be used to merge the different activation levels into a single value. In part d) the activation level being passed to the next nodes has become much smaller than the initial value; for nodes G and H the value is 1.28. This fading of the activation levels is one way in which the spreading process can be controlled before it covers most of the network. Usually a threshold value is set so that no node is activated if its level is smaller than the threshold. With a threshold of 9 for example, the activation will not go farther than node E.

5.3.5 Spreading Activation in Information Retrieval

Spreading activation can be used in a variety of tasks in information retrieval. Three tasks directly related to information retrieval are vocabulary expansion, systematic search of relevant documents, and search for new relevant documents.

In vocabulary expansion, the activation process starts at certain term nodes, usually query terms; from these nodes, the activation moves across the network until a set of related terms is found. One way the expansion can proceed is by following term-term links corresponding to thesaurus relationships; another way is by following the term-document links and then, for each document reaching the corresponding document-term links. The second alternative has the advantage that it dynamically adjusts to changes in the way terms are used. However, this method usually involves a much larger portion of the network. These techniques of vocabulary expansion are similar to the associative linear retrieval systems developed by some authors [SB88b].

Vocabulary expansion by spreading activation has the advantage of naturally combining different sets of vocabularies [Pre80]. For example in an environment with controlled and uncontrolled vocabulary, even though users may be familiar with one of the vocabularies, the query formulation may be expanded to include terms of the other vocabulary.

Salton and Buckley analyze a simple spreading activation methodology [SB88b], which when interpreted as a weighting system, will not perform at the level of a standard *tf*idf* weighting. Furthermore, normalizing the query vector after the weights have been assigned was found to improve performance.

A systematic search for relevant documents using spreading activation starts at the nodes of the query terms; from these nodes the search spreads as far as possible, assigning activation level values to the nodes encountered during the expansion. At the end, those document nodes with the highest activation level are retrieved and presented to the users. Cohen [CK87] found that a full match between the query and the document nodes with the highest activation level is necessary to increase the precision of the retrieved set. The use of spreading activation in a systematic search eventually touches all the nodes and links of the network, which is normally inefficient. Rather than studying the performance of spreading activation in a systematic search of an entire collection, this chapter analyzes the use of spreading activation to search for new relevant documents based on a set of known relevant documents.

5.3.6 Constrained Spreading Activation

An unrestricted spreading activation process can quickly involve most of the nodes of the network. This large expansion of the search is unacceptable from a performance point of view. Clearly, it is necessary to restrict the number of nodes activated during the spreading activation process. This control is referred as *constrained spreading activation.* Spreading activation can often be constrained using the following strategies:

- distance: a limit is imposed on the number of links traversed so that the activation can not travel very far from the originating nodes.
- fading: the activation weight becomes smaller with each link traversed until it becomes so small (smaller than a threshold) that the activation process stops.
- fan-out degree: nodes with a very large number of links are not activated, so that the activation of a large number of other nodes is avoided.
- link type: the type of a link determines whether or not that link is used in the search.

5.4 Nearest-Neighbor Network

The network studied in this chapter consists of two types of nodes: *document* and *term*; and three types of links: *document-term*, *term-document*, and *document-document.* Documents associated by the last type of link are called *nearest neighbors*, abbreviated NN.

NNs have been studied as devices for document clustering, and some authors consider them more cost-effective than full clustering, [Wil88,GLW86]. NN clustering differ from the cluster methodologies presented in Chapter 4 because the NNs define an overlapping non-hierarchical cluster organization of the collection. This organization has for each document a cluster consisting of that document and its NNs. Clearly, a given document may be part of several clusters since that document may be the NN of several documents. On the other hand each document is in only one low level cluster in the cluster hierarchies seen in Chapter 4.

The NN network presented above is implemented using the following three files:

1. A *document* file containing the term vectors for each document of the collection. Another way to look at this file is to consider it as the document-term link file.

2. An inverted file containing for each term the list of documents where that term appears. In other words, this file contains the term-document links.

3. An inverted file implementing the NN links: for each document the list of its NNs is assumed to be available.

Basically, this organization constitutes a conventional inverted-file retrieval system with an additional file providing the NN links. An efficient implementation of relevance feedback is possible using this organization because the inverted lists needed for inverted retrieval are easily accessible. On the other hand, the NN inverted file supports navigation of the network by nearest-neighbor links.

5.5 Browsing Algorithm

The spreading activation search algorithm presented in this section is an approximation of the way users browse throughout a collection. Basically, it is assumed here that users will examine in more detail those documents more similar to the most *interesting* documents found so far. Whenever a document does not lead to other interesting documents, users go back to previously found interesting documents and explore their nearest neighbors. An activation level value, which is computed for each document node encountered during the search, approximates the interest evidenced in a particular document. Only those documents with the highest activation level are inspected in more detail and their nearest neighbors are brought into the search. This browsing algorithm uses a *heap* data structure to store documents encountered so far and to keep them sorted by activation level.

The browsing algorithm of this section is very similar to the algorithm presented in section 4.4. At the beginning, the heap is initialized with all the NNs of the known relevant documents; then as long as more documents need to be retrieved and the heap is not empty, the document with the highest activation level is taken out of the heap and added to the set of retrieved documents. Furthermore, the NNs of the document taken out of the heap

- Mark all document nodes as *not-visited.*
- Make **heap** empty.
- For each node n corresponding to a relevant retrieved:
  - Mark **n** as *visited.*
  - Put all NNs of **n** with non-zero similarity with the query into the heap.
  - Mark all NNs of **n** as *visited.*
- While more documents are wanted and **heap** is not empty do:
  - Take the node of the heap with the largest query similarity, n.
  - Put **n** in the set of retrieved documents.
  - Put all not-visited NNs of **n** with non-zero similarity with the query into the heap.
  - Mark all NNs of **n** as *visited.*

Figure 5.2: Browsing algorithm in a network

are accessed, their activation levels computed, and put into the heap. Each document is flagged when its activation level is computed, so that it will be inspected only once. Figure 5.2 presents the browsing algorithm in more detail. In Figure 5.3 the algorithm is shown working on a small network. In that figure, a square surrounds the nodes of the current elements of the heap, while double circles represent those nodes marked as visited. The elements of the heap are represented [**n**, **a**] pairs, where the first number is the node and the second number is the activation level. In this example, the activation level value is computed by taking the product of the weights of the links.

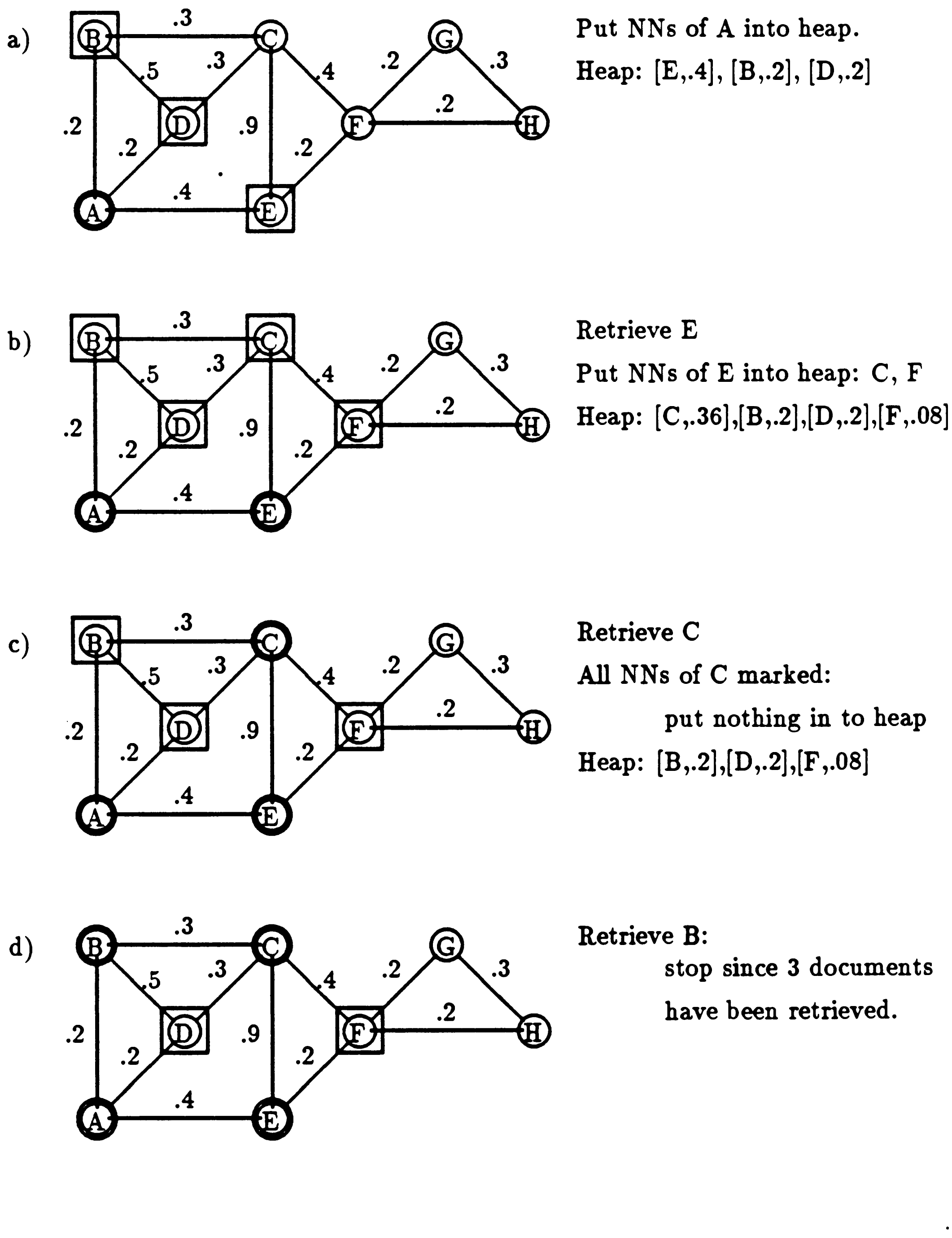


Figure 5.3: Browsing algorithm on a small network

5.6 Activation Level

The effectiveness of the browsing algorithm presented above depends on how well the activation level of each node represents the relevance of that node. There is not a single specific way in which the activation level can be computed, and many different alternatives have been proposed. Even though, the activation level formula used in this study is ad hoc, its general structure follows principles found useful by other researchers. In particular, a retrieval model is presented in van Rijsbergen [vR86] where the system must determine the plausibility of a relationship between the query and a document by combining together different sources of evidence about that relationship. For example, if a given document is retrieved by both cluster search and inverted retrieval search, the plausibility of that document being relevant may be considered greater than if the document is only retrieved by one of the searches. In this study, the activation level is used to measure the plausibility of a relevance relationship between a document and a query. Two sources of evidence are used to determine the level of activation of a given node. One source of evidence is the similarity between the query and that node; the second source of evidence comes from the path used to reach the node from one of the nodes where the search started. A fading factor is introduced to reduce the important of the second source of evidence because that information is less important when the search is far from the starting points [CLC88,CK87].

Formula 5.1 shows how the activation level is computed for this study.

$$ACT\_LEVEL(D_i) = \alpha \cdot Sim(Q, D_i) + \beta \cdot \gamma^k \cdot Sim(D_i, D_j). \qquad (5.1)$$

Where:

- Sim is the vector similarity function,
- D_i is the document whose activation level is being computed,
- Q is the query vector,
- D_j is document from which D_i is reached ,
- α is the importance attached to the similarity between the query and D_i,
- β is the importance attached to the link leading to D_i,
- γ is a fading factor that reduces the importance of $Sim(D_i, D_j)$ as the search moves away from the starting documents,
- k is number of links between D_i and any of the starting documents.

5.7 Experimental Runs

Three parameters are involved in formula 5.1: α, $\beta$and γ. Furthermore, two other parameters affect the performance of the search: $\mathcal{N}$ the number of NNs stored for each document, and $\mathcal{L}$ the maximum number of links that can be traversed from the starting points of the search.

In order to find the best combination of these five parameters and compare the performance obtained with a standard relevance feedback system, many experimental runs were performed using different sets of parameters. These experiments were done with the same collections presented in section 4.5:

Table 5.1: Set of parameter values tested

Parameter	Values
α	0.0, 1.0
β	0.0, 0.3, 0.5, 0.8, 1.0
γ	0.3, 0.5, 0.8, 1.0
$\mathcal{L}$	1, 2, 3, 4, 5, 10, ∞
$\mathcal{N}$	5, 10, 15, 20

CACM, INSPEC and NEWS. Each collection consists of three files: a document file, an inverted file, and a NN file which is an inverted file giving for each document a list of its nearest neighbors. Table 5.1 gives the different parameter values used in the experiments. Parameter $\mathcal{N}$ is especially important since it has a direct impact on the storage requirements of the collection; the larger $\mathcal{N}$ the bigger the NN inverted file will be. For this reason, the impact of $\mathcal{N}$ will be highlighted in the analysis of the results.

5.8 Analysis of Results

This section analyzes the effect of each of the following five parameters on retrieval performance: importance of the query similarity (α), importance of the NN similarity (β), fading factor (γ), maximum numbers of links traversed ($\mathcal{L}$), and number of NNs per document ($\mathcal{N}$). The analysis optimizes one parameter at a time; that is, after the best value is determined for a given parameter, that value is used in the analysis of other parameters. In general, this local optimization process does not necessarily produce an optimal combination of parameters. However, many of the parameters, α, $\mathcal{L}$ and $\mathcal{N}$, have best values that are clearly cut and little affected by the interaction of the other parameters.

The experimental runs are evaluated using the precision after 15 documents have been retrieved. This evaluation measure was chosen because the browsing search does not produce a complete rank of an entire collection. For many of the tables in this section, the figures shown are the average of the precision after 15 obtained over several different runs.

5.8.1 Importance of Query Similarity: α

As can be seen in Formula 5.1 in page 141, α is the weight associated to the similarity between a document and a query. The larger this parameter, the more important is the query similarity to determine the activation level of a document.

This parameter was tested at two levels: 0.0 and 1.0. In the first case, the query similarity is totally ignored and only the strength of the NN links is used to set the activation level. In the second case, the query similarity becomes the major factor of the activation level. The results obtained showed that removing the query as a source of evidence, $\alpha = 0.0$, systematically worsens the performance of the search. Because of this result, the analysis of the remaining parameters is done uses $\alpha = 1.0$, which reduces Formula 5.1 to:

$$ACT\_LEVEL(D_i) = Sim(Q, D_i) + \beta \cdot \gamma^k \cdot Sim(D_i, D_j). \tag{5.2}$$

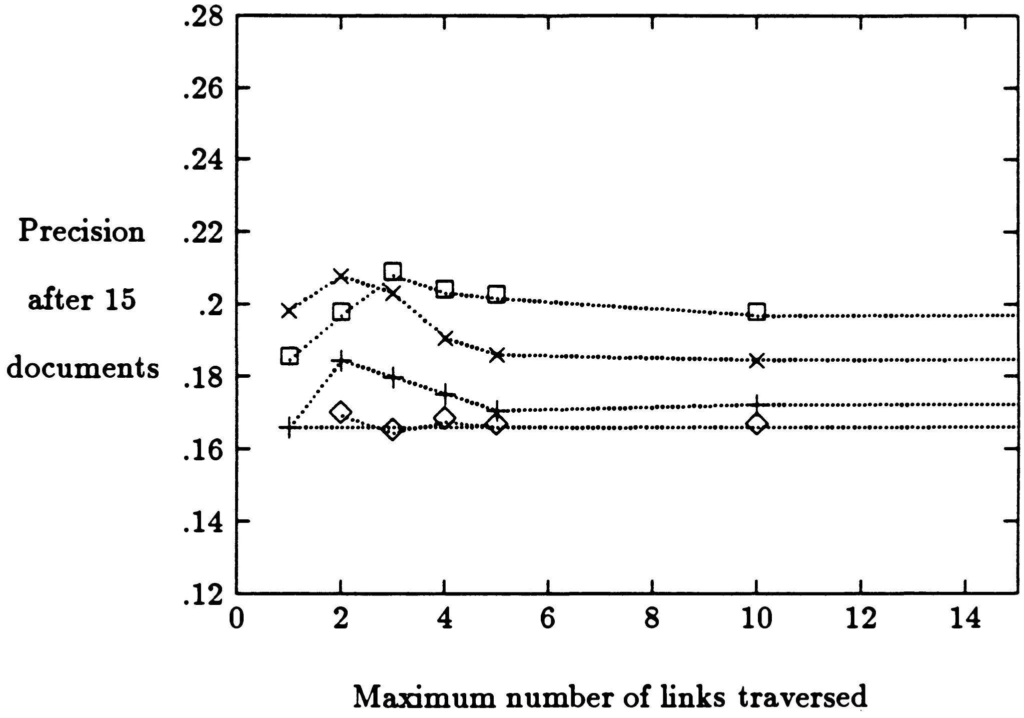


Figure 5.4: Typical case of the behavior of the maximum number of links traversed: $\mathcal{L}$

5.8.2 Maximum Number of Links Traversed: $\mathcal{L}$

In order to constrain the browsing search, the number of NN links that can be traversed away from the starting points of the search is limited by the parameter $\mathcal{L}$. Seven values were used with this parameter: 1, 2, 3, 4, 5, 10 and ∞. Where value allows the activation process to expand arbitrarily far from the starting points. The results obtained confirm the observation made by other researchers [CLC88,CK87] that low values of $\mathcal{L}$ are good enough, and that performance does not necessarily improve with larger values. Figure 5.4 illustrates the typical behavior of $\mathcal{N}$. Each curve in the figure correspond to the precision obtained using different number of NNs per document ($\mathcal{N}$). The symbols $\diamond$, $+$, $\square$ and $\times$ represent the results obtained when $\mathcal{N}$ has values 5, 10, 15 and 20. As shown, after an initial period of changes, all curves become constant and no further changes occur. The level at which the curves become constant vary with the other parameters; but it is always the case that for $\mathcal{L} = 5$ all the curves have reached their constant value. In most cases, this leveling of retrieval is a consequence of the of the fading factor parameter (γ) which quickly trivializes the contribution of the NNs to the activation level. On the other hand, since only 15 documents are retrieved, no path with more than 15 links is really useful. In general $\mathcal{L}$ should be regarded as a logarithmic parameter in the sense that a small increment of $\mathcal{L}$ correspond to a very large increment of the portion of the collection that can be reached from the starting points. Because values of $\mathcal{L}$ larger than 5 do not improve retrieval, only the values 1 through 5 will be used for the analysis of the parameters α, β, γ and $\mathcal{N}$ in the following subsections.

5.8.3 NN Similarity Ignored: $\beta = 0$

The parameter β in Formula 5.1 determines the importance attached to the NN similarity when the activation level of a document node is computed. Table 5.1 shows the five values of β tested in the experimental runs. This subsection only analyzes the case when $\beta = 0$, that is, the activation level only takes into account the query similarity and ignores the NN similarity. The following subsection deals with the non-zero values of β, where the NN similarities are taken into account with different strength. When the NN

Table 5.2: Average precision for 5 levels of $\mathcal{N}$, $\beta = 0.0$

		Number of NNs, $\mathcal{N}$			
	feedback	5	10	15	20
CACM	0.1659	0.1550	0.1752	0.1988	0.1972
		-6.6%	5.6%	19.8%	18.9%
INSPEC	0.2400	0.2064	0.2295	0.2336	0.2365
		-14.4%	-4.4%	-2.7%	-1.5%
NEWS	0.2030	0.1711	0.2101	0.2184	0.2109
		-15.7%	3.5%	7.6%	3.9%

similarity is ignored, the Formula 5.1 is reduced to:

$$ACT\_LEVEL(D_i) = 1.0 \cdot Sim(Q, D_i) + 0.0 \cdot \gamma^k \cdot Sim(D_i, D_j) = Sim(Q, D_i), \tag{5.3}$$

since the importance of the query similarity (α) is set to 1.0. Another way to view the situation, is to consider that when $\beta = 0.0$ the search follows the NN links without taking into consideration their strength. This situation is similar to the type of search done in Chapter 4 where the only information used during the search was the query similarity for clusters and documents.

Table 5.2 shows the precision after 15 documents have been retrieved, as a function of the number of NNs when $\beta = 0.0$; the percentage of change relative to the precision of relevance feedback is also shown. For each collection and each level of the number of NN per document($\mathcal{N}$) the precision shown is obtained by averaging the precision of five levels of $\mathcal{L}$: 1-5.

The browsing algorithm performs well in the CACM collection especially for large values of $\mathcal{N}$. $\mathcal{N} = 15$ and $\mathcal{N} = 20$ have similar precision and they are almost 20% better than relevance feedback. For INSPEC, the performance of the browsing algorithm is worse than the performance of relevance feedback. The values of 15 and 20 are also the best ones for $\mathcal{N}$, but they only reach the level of precision of relevance feedback without improving it. Finally, for the NEWS collection, the best performance was obtained by $\mathcal{N} = 15$ which is 7.6% better than relevance feedback. Overall, the larger values of $\mathcal{N}$ perform better than the smaller values.

The different levels of success (or the lack of it) of the browsing search in the three collections is analyzed in more detail in section 5.8.6. In that section, it is shown that the distribution of relevant documents as nearest neighbors of other relevant documents can explain the different levels of success of the browsing search.

5.8.4 Fading Factor: γ

The previous subsection analyzed the case when the activation level is computed ignoring the NN similarity ($\beta = 0$). In contrast, this subsection and the following one study the case where the NN similarity is taken into account ($\beta \neq 0$). The activation level depends not only on the number of NN ($\mathcal{N}$), and the maximum number of links traversed ($\mathcal{L}$), but also depends on the different levels of the fading factor (γ), and the importance of the NN similarity (β).

This section studies the effect of the fading factor (γ) on the retrieval performance of the browsing algorithm. The analysis is based on Table 5.3 Table 5.3 The values shown in that table are obtained by averaging the precision after 15 documents have been retrieved, over all the non-zero levels of the NN similarity (β) and the maximum number of links traversed ($\mathcal{L}$). For each of the three collections, the precision values are shown as functions of the fading factor (γ), and the number of NNs per document ($\mathcal{N}$). For all collections and all values of the number of NNs, the lower the fading factor, the better the precision obtained. In other words, the precision obtained when $\gamma = 0.3$ is better than the precision obtained when $\gamma = 0.5$, which in turn is much better than the precision obtained by $\gamma = 0.8$ or $\gamma = 1.0$. By being small, γ quickly reduces the activation level of documents far from the known relevant documents to just the query similarity. In other words, long paths of NN links are not reliable as sources of information.

Although the fading factor (γ) must be small, it can not be zero because in that case, the activation level formula becomes:

$$ACT\_LEVEL(D_i) = 1.0 \cdot Sim(Q, D_i) + \beta \cdot (0.0)^k \cdot Sim(D_i, D_j) = Sim(Q, D_i), \tag{5.4}$$

which is exactly the same case obtained when the NN similarity is ignored ($\beta = 0$). On the other hand, the performance of the browsing search when $\beta = 0$ is not as good as the best performance shown in Table 5.3 for the best values of γ, showing that retrieval is improved when γ is greater than zero.

Table 5.3: Precision as function of the fading factor (γ) and the number of NN per document ($\mathcal{N}$)

			Fading factor: γ			
	fdbk	$\mathcal{N}$	0.3	0.5	0.8	1.0
CACM	0.1659	5	0.1598	0.1570	0.1483	0.1382
			-3.7%	-5.4%	-10.6%	-16.7%
		10	0.1861	0.1802	0.1618	0.1389
			12.2%	8.6%	-2.5%	-16.3%
		15	0.1995	0.1851	0.1611	0.1323
			20.3%	11.6%	-2.9%	-20.3%
		20	0.1990	0.1781	0.1468	0.1264
			20.0%	7.4%	-11.5%	-23.8%
INSPEC	0.2400	5	0.2045	0.2016	0.1968	0.1857
			-14.8%	-16.0%	-18.0%	-22.6%
		10	0.2465	0.2420	0.2173	0.1902
			2.7%	0.8%	-9.5%	-20.8%
		15	0.2636	0.2578	0.2350	0.2119
			9.8%	7.4%	-2.1%	-11.7%
		20	0.2567	0.2488	0.2216	0.1984
			7.0%	3.7%	-7.7%	-17.3%
NEWS	0.2030	5	0.1716	0.1694	0.1646	0.1590
			-15.5%	-16.6%	-18.9%	-21.7%
		10	0.2190	0.2146	0.2007	0.1865
			7.9%	5.7%	-1.1%	-8.1%
		15	0.2289	0.2255	0.2098	0.1951
			12.8%	11.1%	3.3%	-3.9%
		20	0.2279	0.2250	0.2097	0.1967
			12.3%	10.8%	3.3%	-3.1%

5.8.5 Effect of the NN Similarity, $\beta \neq 0$

This subsection studies the effect on retrieval of the NN similarity (β); also studied is the effect of the number of NNs per document ($\mathcal{N}$). During the analysis a fading factor value of 0.3 is used because that is the best value found for that parameter in the previous subsection.

Table 5.4 details the performance obtained by the different values of β for the three collections and the different levels of $\mathcal{N}$. As usual the values shown are the average of the precision for the five levels of $\mathcal{N}$. Overall, the results obtained in the three collections are similar although the performance figures vary with each collection. Table 5.4 shows that the number of NNs per document, $\mathcal{N}$, has more impact on the precision than the value of β. In other words, when $\mathcal{N}$ is changed while β is kept fixed, the precision changes more than when β is changed but $\mathcal{N}$ is fixed.

In general, using 15 NNs per document ($\mathcal{N} = 15$) gives the best performance for all collections and all values of β. This value of $\mathcal{N}$ also produces significant improvements over relevance feedback. In contrast, smaller values of $\mathcal{N}$ ($\mathcal{N} = 5$, $\mathcal{N} = 10$) are not as good as $\mathcal{N} = 15$, and in the case of $\mathcal{N} = 5$, its precision is even lower than the precision of relevance feedback. As the next section discusses, the distribution of relevant documents as NNs of other relevant documents is such that for small values of $\mathcal{N}$, not enough documents are inspected to have a reasonable probability of finding a relevant document.

The best value for the NN similarity weight (β) depends on the collection and on the number of NNs per document ($\mathcal{N}$). If the analysis is restricted to the best value of $\mathcal{N}$, $\mathcal{N} = 15$, then the best values for (β) in the CACM,

Table 5.4: Precision as function of NN similarity weight (β) and number of NNs per document ($\mathcal{N}$)

	fdbk	$\mathcal{N}$	NN similarity weight: β 0.3	0.5	0.8	1.0
CACM	0.1659	5	0.1566	0.1559	0.1628	0.1640
			-5.6%	-6.0%	-1.9%	-1.1%
		10	0.1876	0.1870	0.1851	0.1848
			13.1%	12.7%	11.6%	11.4%
		15	0.2041	0.1981	0.1985	0.1972
			23.0%	19.4%	19.7%	18.9%
		20	0.2059	0.2031	0.1959	0.1910
			24.1%	22.4%	18.1%	15.1%
INSPEC	0.2400	5	0.2068	0.2048	0.2043	0.2020
			-13.8%	-14.7%	-14.9%	-15.8%
		10	0.2416	0.2464	0.2494	0.2487
			0.7%	2.7%	3.9%	3.6%
		15	0.2560	0.2615	0.2693	0.2677
			6.7%	9.0%	12.2%	11.5%
		20	0.2553	0.2610	0.2557	0.2549
			6.4%	8.8%	6.5%	6.2%
NEWS	0.2030	5	0.1727	0.1713	0.1715	0.1709
			-14.9%	-15.6%	-15.5%	-15.8%
		10	0.2218	0.2202	0.2186	0.2155
			9.3%	8.5%	7.7%	6.2%
		15	0.2248	0.2297	0.2283	0.2328
			10.7%	13.2%	12.5%	14.7%
		20	0.2222	0.2275	0.2295	0.2323
			9.5%	12.1%	13.1%	14.4%

INSPEC and NEWS collections are respectively, 0.3, 0.8 and 1.0. Overall however, large values of β give consistently good results, showing that in principle, the NN similarity information is as important as the query similarity information.

Although the best value of β depends on the collection, some values consistently give good performance in all three collections; for example, using $\beta = 0.8$ the performance in CACM and NEWS will only be 3.3% and 2.2% short of the best performance. $\beta = 1.0$ also gives good performance in all collections.

5.8.6 Analysis of the Effect of $\mathcal{N}$

The better performance of larger values of $\mathcal{N}$ can be explained by analyzing the distribution of relevant documents among the NNs of relevant documents, since the browsing search depends on finding new relevant documents close to a known relevant document. Table 5.5 shows for each collection the percentage of relevant documents with a given number of relevant nearest neighbors. For example, 43% of the relevant documents of CACM have no relevant documents among the top 5 NNs; on the other hand, 23% of the relevant documents of CACM have five or more relevant documents among the top 20 NNs. Overall, CACM has a larger percentage of its relevant documents close to other relevant documents than INSPEC or NEWS. These two collections have very similar distribution of NNs. As The table shows, when $\mathcal{N}$ is large, the number of relevant documents with at least one relevant document as nearest neighbor increases, but the largest improvement occurs from 5 to 10 NNs corresponding to the usually large gap on the performance of $\mathcal{N} = 5$

Table 5.5: Distribution of relevant documents among NNs

	CACM				INSPEC				NEWS			
	5	10	15	20	5	10	15	20	5	10	15	20
0	43	29	23	19	64	51	42	35	69	56	48	42
1	26	24	21	19	25	26	29	28	22	27	28	27
2	16	18	18	17	8	13	14	18	8	10	12	15
3	10	11	11	13	2	6	7	7	1	5	7	7
4	3	6	10	9	1	2	5	6	0	2	3	5
5+	3	12	17	23	0	1	3	5	0	1	2	5

and $\mathcal{N} = 10$. The distribution of relevant documents can not explain the performance difference between INSPEC and NEWS since the distribution for these two collections is very similar. The better performance found in NEWS is a consequence of the much worse performance of relevance feedback in that collection. In other words browsing is less affected than relevance feedback by the size of the collection.

5.9 Summary of Results

Table 5.6 summarizes the results obtained in this chapter. Each row of the Table presents the case of one of the parameters studied. The column *Results* gives a brief presentation of the best values found in each case, while the column *Observations* interprets the results in a more general context.

Overall, the performance obtained by the network browsing using the best set of parameter values renders browsing attractive. However, the need to

Table 5.6: Summary of results

Parameter	Results	Observations
$\alpha = 0.0$	Systematically worse than $\alpha = 1.0$	The query similarity is a more important (reliable) source of information than the NNs.
n	Values larger than 5 do not change precision. Small values (1,2,3) are usually the best.	Only a limited number of links need to be traversed.
$\beta = 0.0$	Best performance for large $\mathcal{N}$, especially $\mathcal{N} = 15$ ($\mathcal{N} = 20$ also good). No improvement for INSPEC.	Need more than a few NNs to search effectively. Query similarity alone does not produce improvements over relevance feedback.
γ	The smaller the better. $\gamma = 0.3$ the best, $\gamma = 0.5$ also good.	NNs are not very reliable, so their influence must quickly decrease as the search moves far from the starting points.
β	$\mathcal{N}$ has more impact on effectiveness than β. $\mathcal{N} = 5$ has low performance. $\mathcal{N} = 15$ has best performance, but $\mathcal{N} = 20$ is competitive. Best value of β depends on the collection, but large β performs consistently well.	Give similar importance to the query ($\alpha = 1.0$) and to the NNs ($\beta = 0.8, 1.0$); but use γ to reduce the influence of the NNs.
N	$\mathcal{N} = 15$ is the best value but $\mathcal{N} = 20$ is close second.	Too few NNs ($\mathcal{N} = 5$) are not enough to find relevant NNs.

Table 5.7: Precision as function of $\beta$and $\mathcal{N}$

	relevance feedback	average extension	$\mathcal{N} = 15$ $\beta = 0.8$
CACM	0.166	0.240	0.199
		44.7%	19.8%
INSPEC	0.240	0.224	0.269
		-6.7%	12.2%
NEWS	0.203	0.191	0.228
		-5.9%	12.5%

compute the NNs is a disadvantage of this methodology. For a large collection, computing the NNs is not a trivial operation, but a total recomputation of the NNs is not necessary if the collection is expanded. The NNs of new documents can be easily computed and only those documents with a new document as a new NN need to be updated.

5.10 Cluster Browsing vs. Network Browsing

This section compares the performance results obtained by the network browsing methodology described in this chapter, and the results obtained by the cluster hierarchy browsing process of the previous chapter. Table 5.7 presents the average precision obtained by three different process: relevance feedback, cluster browsing using the average extension hierarchy presented in section 4.5, and network browsing with parameters $\mathcal{N} = 15$ and $\beta = 0.8$. Also shown in that table is the percentage of change with respect to relevance feedback. The precision figures of Table 5.7 show that network browsing is more consistent than cluster browsing; the later obtains excellent improvement over relevance feedback in the CACM collection, 44.7%, but has no better performance than relevance feedback for the other two collections. On the contrary, network browsing consistently performs better than relevance feedback for the three collections. This difference in performance suggests that cluster browsing is more dependent on the distribution of relevant nearest neighbors than network browsing. As mentioned before, CACM has a better distribution of relevant nearest neighbors than either INSPEC or NEWS; and it is precisely in CACM where the performance of cluster browsing is much better than even the performance of network browsing. In other words, to decide which of the browsing methodologies is better the major factor to be taken into account is the distribution of relevant nearest neighbors. If there is a good possibility for a relevant document to have another relevant document as a nearest neighbor, then cluster browsing can be expected to perform well; otherwise, when relevant nearest neighbors are rare, network browsing will perform better.

On the efficiency side, both, cluster and network browsing, are very fast since only a limited subset of the collection is inspected. Because of the limited scope of the search, the efficiency of both browsing methodologies does not deteriorate with an increase in the size of the collection.

Chapter 6

Conclusions

The overall goal of this thesis is the analysis and evaluation of mechanisms that can be used to assist users during the search process. Two aspects of this search process have been studied in detail. First, interactive experiments are performed to evaluate techniques that allow users to improve the formulation of their search statements by the addition of term phrases. On the other side, two file organizations are analyzed in terms of how good they are to provide users with new information. The systematic search provided by these two organizations can be easily adapted to guide users in an ordered browse of a large collection of data.

More specifically, the first part of this thesis evaluates different strategies to improve the formulation of queries by adding term phrases generated from the query text. Two phrase generation methodologies are compared. One of them uses statistical information about the position, frequency and co-occurrence of the different single terms in the query to combine into a term phrase, those single terms meeting certain conditions. The other phrase generation methodology uses a syntactical analysis of the query to create phrases by combining words related by syntactical relationships. The statistical approach was found to improve retrieval better than the syntactical approach; this difference is fundamentally due to the larger set of phrases that the statistical approach generates.

Some of the experiments on the first part of the thesis present users with a list of phrases. From that list, the users chooses those phrases to be added to the query. In general, this form of user intervention does not perform much better than some simple non-interactive alternatives like adding to the query all the phrases presented to the users. In other interactive experiments, the users also have to choose phrases or single terms from a list presented to them; but that list is generated from known relevant documents. In other words, these experiments introduce a user selection step into the standard query modification process by relevance feedback. Again, this form of user intervention does no perform much better than simple non-interactive strategies. In some cases however, a potential for large improvements on retrieval has been found. But to realize those improvements users should have experience choosing phrases and they also must be motivated to scan long lists of phrases.

A major factor limiting the improvements users can make is the restriction to two term phrases. This restriction was introduced because of the potentially large number of phrases that can be generated when longer phrases are allowed. Many useful phrases are not chosen by users because the two terms of those phrases are not strongly connected. Those phrases, however, are part of larger phrases clearly related to the query being searched; by restricting the phrases to only two terms, many good multi-term phrases must be cut into several pieces. In an operating environment, it will be necessary to train users to recognize these pieces of phrases. One alternative worth exploring is to refine the syntactical phrase generation methodology, Fagan [Fag87], used in this study. In principle, this syntactic generation can produce multi-term phrases without a combinatorial explosion on the number of phrases generated, and these phrases will be longer and more appealing to the users.

Another limitation on the effectiveness of users comes from the fact that the collection used in these experiments is not very large. It is known that in cases like this, some terms are local synonyms of other terms. In other words two terms, normally not considered synonyms, behave like synonyms in the restricted environment of a collection. For example, the terms *machine* and *computer* can be considered synonyms in a computer science collection. The local synonyms may cause users to miss useful phrases because one of the phrase terms is replaced by a local synonym; leaving a phrase that does not seem appropriate from the global point of view of the user.

The results of the query modification experiments suggest that the interactive mechanisms may be better directed towards allowing the users to explore the vocabulary of the collection. In other words, a browsing mechanism can be provided so that users may find related terms and alternative ways to formulate their searches. This browsing process can be done in a manually-built structure like a thesaurus, or an organization created by statistical associations between the terms.

In the second part of this thesis, two structures are studied in terms of the new information they can provide to the users. The first structure is the complete link cluster hierarchy. Three versions of this structure are studied; the basic complete link hierarchy, an upward extension of the basic case created by the complete link clustering of the top level centroids of the basic hierarchy, and an upward extension created by average link clustering of the top level clusters of the basic hierarchy. The browsing algorithm used in the cluster experiments starts the search at known relevant documents and then orderly expands the search among the other documents in the same top level cluster. One important limitation of the browsing algorithm is that the search does not spread from one top level cluster to another because that is equivalent to a full systematic search of the entire collection. In general, the results of the cluster browsing experiments show that this methodology can be, under certain circumstances, a useful alternative to relevance feedback. Although for the larger collections used, cluster browsing does not perform as well as relevance feedback, this was mainly caused by weak nearest neighbor relationships between relevant documents and not by the larger size of the collections.

Two opposite effects were detected in the results. When the top level clusters are small, the browsing search can effectively detect new relevant documents in those clusters. On the other hand, because the clusters are small, only a limited portion of the cluster hierarchy can be inspected. One consequence of this limited search is that many relevant documents are unreachable by the browsing algorithm because they happen to be in top level clusters without known relevant documents.

The other structure analyzed is a network of documents connected by links representing the nearest neighbor relationships between different documents. In this environment the browsing search is implemented as a spreading activation process. Starting at known relevant documents, the search proceeds to inspect neighboring documents according to an activation value. This value is computed taking into account two sources of information about the possible relevance of a given node. The first indication that a document may be relevant is the similarity between that document and the query. The second evidence is the strength of the nearest neighbor similarity path between that document an a known relevant document. Because the unreliability of the second source, a fading factor was included to reduce the influence of that source of evidence as the search moves away from the known relevant documents. Besides the activation value, two parameters affect of the browsing search: the number of nearest neigbors available per document, and the maximum number of links that can be traversed away from the starting nodes.

The results of the experiments run for the network organization showed that both of the sources of information mentioned above must be similarly weighted; although emphasizing the query similarity information does not deteriorate retrieval. On the other hand, the fading factor should be small to quickly reduce the influence of the nearest neighbor links on the activation level. Furthermore, the experiments showed that retrieval performance is very dependent of the number of nearest neighbors available per document. The best value found was 15; smaller values do not allow the browsing algorithm to quickly cover a large portion of the network around the known relevant documents, while larger values does not improve retrieval.

Both, the cluster hierarchy browsing and the network browsing, are methodologies that should be explore as interactive devices to guide users to browse around in a large collection. This device will allow users to inspect arbitrary parts of the collection and also recall the document with the largest activation value. In other words, a systematic search process provided by the search methodologies presented before is combined with a user directed inspection of the collection.

Appendix A

Syntactic Rules

Conjoined noun phrases: The phrases are generated by distributing the modifier of the last conjunct among the heads of all conjuncts. Figure A.1 shows how the modifier of the last conjunct, NOUN_3 is distributed among the different conjuncts, NP_2, NP_3 and NOUN_3, to create the phrases n_4 n_1, n_4 n_1, and n_4 n_1.

Conjoined modifiers: The phrases are generated by associating the head of each conjunct with the head of the noun phrase. Figure A.2 illustrates the case: NOUN_3 is distributed among NOUN_1 and NOUN_2 to create the phrases n_1 n_3 and n_2 n_3.

Prepositional phrases: The phrases are generated by extracting the noun phrase and processing it according to the noun phrase rules. Figure A.3 illustrates the technique. Parse tree *(a)* is converted into parse tree *(b)* by extracting the noun phrase; then the phrases are generated using the conjoined modifier rule.

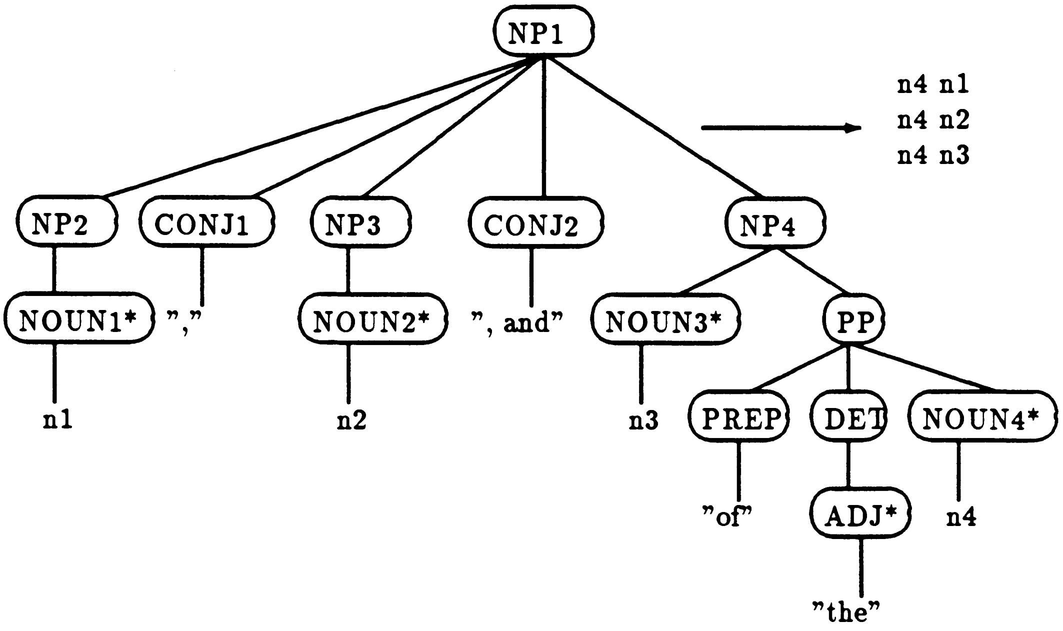


Figure A.1: Conjoined noun phrases

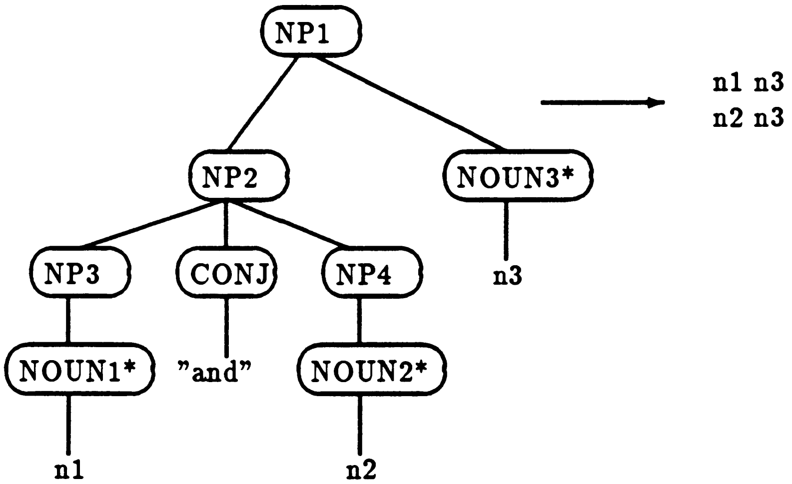


Figure A.2: Conjoined modifiers

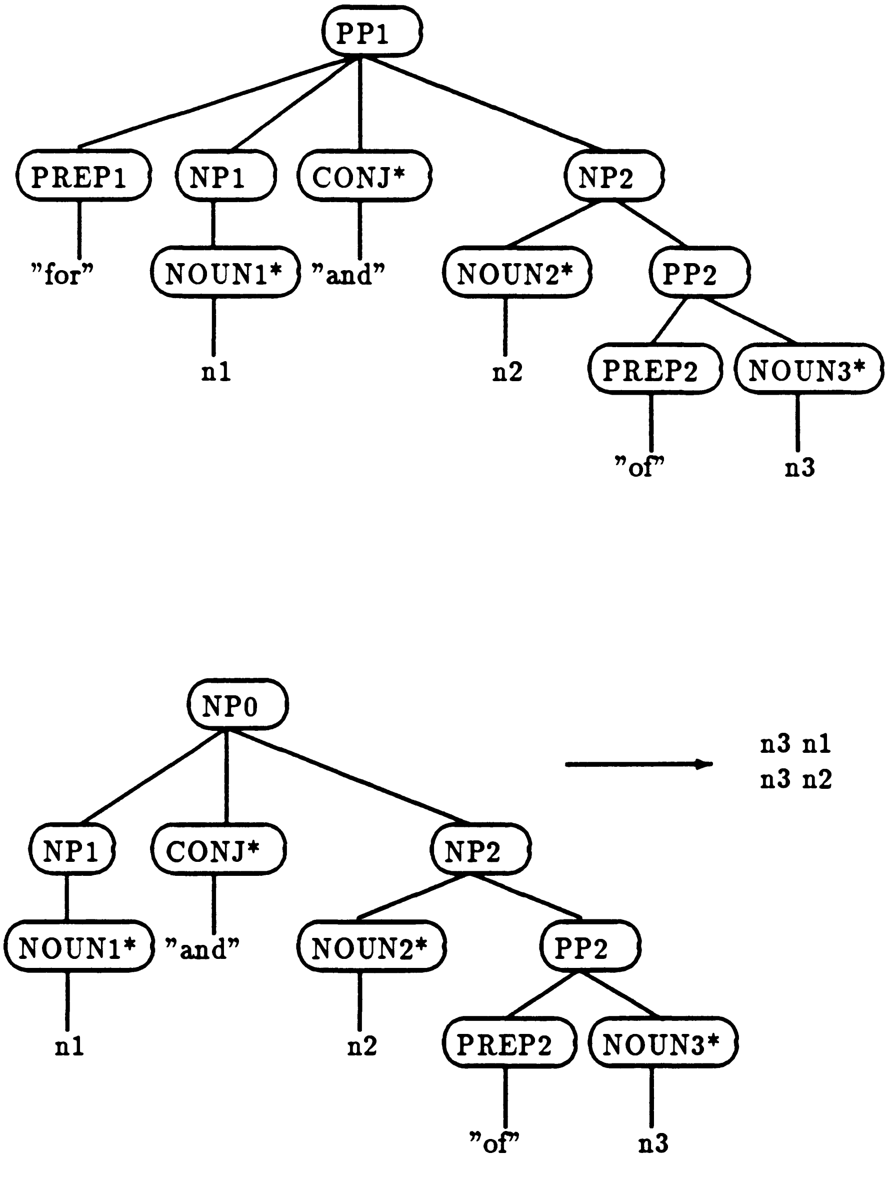


Figure A.3: Prepositional phrases

Bibliography

[AM90] J.E. Araya-Monge. Comparisons of inverted file processing with clustered file manipulations. Technical Report TR 90-1107, Department of Computer Science, Cornell University, Ithaca, New York, 1990.

[AMY88] R.M. Akscyn, D.L. McCracken, and E.A. Yoder. KMS: A distributed hypermedia system for managing knowledge in organizations. *Communications of the CACM*, 31:820–835, 1988.

[Bat79a] M.J. Bates. Idea tactics. *Journal of the American Society for Information Science*, 30:280–289, 1979.

[Bat79b] M.J. Bates. Information search tactics. *Journal of the American Society for Information Science*, 30:205–214, 1979.

[Bat86] M.J. Bates. Subject access in online catalogs: A design model. *Journal of the American Society for Information Science*, 37:357–376, 1986.

[BOB82] N.J. Belkin, R.N. Oddy, and H.M. Brooks. ASK for information retrieval: Part 1. Background and theory. *Journal of Documentation*, 38:61–71, 1982.

[Boo78] A. Bookstein. On the perils of merging boolean and weighted retrieval systems. *Journal of the American Society for Information Science*, 29:156–158, 1978.

[Bro87] H.M. Brooks. Expert systems and intelligent information retrieval. *Information Processing and Management*, 23:367–382, 1987.

[BS86] C. Burgess and K. Swigger. A graphical database interface for casual naive users. *Information Processing and Management*, 22:511–521, 1986.

[Buc85] C. Buckley. Implementation of the SMART information retrieval system. Technical Report 85-686, Computer Science Department, Cornell University, Ithaca, New York, 1985.

[BV85] N.J. Belkin and A. Vickery. *Interaction in Information System: A Review of Research from Document Retrieval to Knowledge-Based Systems.* Longwood Pub. Group, Dover, N.H., 1985.

[CCA89] D.B. Crouch, C.J. Crouch, and G. Andreas. The use of cluster hierarchies in hypertext information retrieval. In *Hypertext'89 Proceedings*, pages 225–237, Pittsburgh, Pennsylvania, 1989.

[CCR71] Y.K. Chang, C. Cirillo, and J. Razon. Evaluation of feedback retrieval using modified freezing, residual collection, and test and control groups. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 355–370. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[CK87] P.R. Cohen and R. Kjeldsen. Information retrieval by constrained spreading activation in semantic networks. *Information Processing and Management*, 23:255–268, 1987.

[CLC88] W.B. Croft, T.J. Lucia, and P.R. Cohen. Retrieving documents by plausible inference: An preliminary study. In *Proceedings of the Eleventh ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 481–494, Grenoble, France, 1988.

[CM86] E. Charniak and D.V. McDermott. *Introduction to Artificial Intelligence.* Addison Wesley, Reading, Massachusetts, 1986.

[Con87] J. Conklin. Hypertext: An introduction and survey. *Computer*, 20:17–41, 1987.

[CP85] W.B. Croft and T.J. Parenty. A comparison of a network structure and a database system used for document retrieval. *Information Systems*, 10:377–390, 1985.

[Cro77] W.B. Croft. Clustering of large files of documents using the single-link method. *Journal of the American Society for Information Science*, 28:341–344, 1977.

[CW88] J.F. Cove and B.C. Walsh. Online text retrieval via browsing. *Information Processing and Management*, 24:31–37, 1988.

[DD80] M. Dillon and J. Desper. Automatic relevance feedback in boolean retrieval systems. *Journal of Documentation*, 36:197–208, 1980.

[DG83] M. Dillon and A.S. Gray. FASIT: A fully automatic syntactically based indexing system. *Journal of the American Society for Information Science*, 34:99–108, 1983.

[Doy62] L.B. Doyle. Indexing and abstracting by association. *American Documentation*, 13:378–390, 1962.

[Fag87] J. Fagan. Experiments in automatic phrase indexing for document retrieval: A comparison of syntactic and nonsyntactic methods. Technical Report TR 87-868 (Ph.D. Thesis), Department of Computer Science, Cornell University, Ithaca, New York, 1987.

[Fen81] C.H. Fenichel. Online searching: Measures that discriminate among users with different types of experiences. *Journal of the American Society for Information Science*, 32:23–32, 1981.

[Fid86] R. Fidel. Towards expert systems for the selection of search keys. *Journal of the American Society for Information Science*, 37:37–44, 1986.

[Fos88] C.L. Foss. Effective browsing in hypertext systems. In *RIAO 88*, pages 82–98, 1988.

[GJ63] V.E. Giuliano and P.E. Jones. Linear association information retrieval. In P. Howerton, editor, *Vistas in Information Handling*, chapter 2, pages 30–54. Spartan Books, Rochelle Park, New Jersey, 1963.

[GLW86] A. Griffiths, H.C. Luckhurst, and P. Willett. Using inter-document similarity information in document retrieval systems. *Journal of the American Society for Information Science*, 37:3–11, 1986.

[Gor81] A.D. Gordon. *Classification*. Chapman and Hall, London, 1981.

[GR69] J.C. Gower and G.J.S. Ross. Minimum spanning trees and single linkage cluster analysis. *Applied Statistics*, 18:54–64, 1969.

[GT83] G. Guida and C. Tasso. An expert intermediary system for interactive document retrieval. *Automatica*, 19:759–766, 1983.

[Hal88] F.G. Halasz. Reflections on NoteCards: Seven issues for the next generation of hypermedia systems. *Communications of the CACM*, 31:836–852, 1988.

[Har75] J.A. Hartigan. *Clustering Algorithms*. Wiley, New York, 1975.

[Har88] D. Harman. Towards interactive query enhancement. In *Proceedings of the Eleventh ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 321–331, Grenoble, France, 1988.

[Hil73] D.J. Hillman. Customized user services via interactions with LEADERMART. *Information Storage and Retrieval*, 9:587–596, 1973.

[Hil83] C.R. Hildreth. To boolean or not to boolean? *Information Technology and Libraries*, 2:235–237, 1983.

[HJM+82] G.E. Heidorn, K. Jensen, L.A. Miller, F.J. Byrd, and M.S. Chodorow. The EPISTOLE text critiquing system. *IBM System Journal*, 21:305–326, 1982.

[HL85] D.T. Hawkins and L.R. Levy. Front end software for online database searching. Part 1: Definitions, system features, and evaluation. *Online*, 9:30–37, 1985.

[HL86] D.T. Hawkins and L.R. Levy. Front end software for online database searching. Part 3: Product selection chart and bibliography. *Online*, 10:49–58, 1986.

[Ide71] E. Ide. New experiments in relevance feedback. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 337–354. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[IS71] E. Ide and G. Salton. Interactive search strategies and dynamic file organization in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 373–393. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[JHMR83] K. Jensen, G.E. Heidorn, L.A. Miller, and Y. Ravin. Parse fitting and prose fitting: Getting hold of ill formedness. *American Journal of Computational Linguistics*, 9:147–160, 1983.

[JP73] R.A. Jarvis and E.A. Patrick. Clustering using a similarity measure based on shared nearest neighbours. *IEEE Transactions on Computers*, c-22:1025–1034, 1973.

[JvR71] N. Jardine and C.J. van Rijsbergen. The use of hierarchic clustering in information retrieval. *Information Storage and Retrieval*, 8:217–240, 1971.

[JW72] G. Jones and P. Wise. Updating thesaurus classifications in response to data base changes. Technical Report Scientific Report No. ISR-21, Section XIII, Department of Computer Science, Cornell University, Ithaca, New York, December 1972.

[Kli73] P.H. Klinbiel. Machine-aided indexing of technical literature. *Information Storage and Retrieval*, 9:79–84, 1973.

[Les69] M.E. Lesk. Word-word association in document retrieval systems. *American Documentation*, 20:27–38, 1969.

[Lov68] J.B. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11:22–31, 1968.

[LS71] M.E. Lesk and G. Salton. Interactive search and retrieval methods using automatic information displays. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 487–505. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[LW67] G.N. Lance and W.T. Williams. A general theory of classificatory sorting strategies. 1. hierarchical systems. *Computer Journal*, 9:373–380, 1967.

[Mas88] M. Massicotte. Improved browsable displays for online subject access. *Information Technology and Libraries*, 7:373–380, 1988.

[MCBC89] Charles T. Meadow, Barbara A. Cerny, Christine L. Borgman, and Donald O. Case. Online access to knowledge: System design. *Journal of American Society for Information Science*, 40:86–98, 1989.

[MHA82] C.T. Meadow, T.T. Hewett, and E.S. Aversa. A computer intermediary for interactive database searching. I. Design. *Journal of the American Society for Information Science*, 33:325–332, 1982.

[MNRH84] D.P. Metzler, T. Noreault, L. Richey, and B. Heidorn. Dependency parsing for information retrieval. In C.J. van Rijsbergen, editor, *Research and Development in Information Retrieval: Proceedings of the Third Joint BCS and ACM Symposium*, pages 313-324, Cambridge, July 1984. Cambrige University Press.

[Mot86] A. Motro. BAROQUE: A browser for relational databases. *ACM Transactions on Office Information Systems*, 4:164–181, 1986.

[MR81] R.S. Marcus and J.F. Reintjes. A translating computer interface for end-user operation of heterogeneous retrival systems. I. Design. *Journal of the American Society for Information Science*, 32:287–303, 1981.

[Odd77] R.N. Oddy. Information retrieval through man-machine dialogue. *Journal of Documentation*, 33:1–14, 1977.

[PD79] M.G. Pacak and G.S. Dunham. Computers and medical language. *Medical Informatics*, 4:13–27, 1979.

[Por80] M.F. Porter. An algorithm for suffix stripping. *Program: Automated Library and Information Systems*, 14:130–137, 1980.

[Pre80] S.E. Preece. An online associative query modification methodology. *Online Review*, 4:375–382, 1980.

[RGL87] J.R. Remde, L.M. Gomez, and T.K. Landauer. Superbook: An automatic tool for information exploration - hypertext? In *Proceedings of Hypertext'87*, pages 175–188, 1987.

[RJ71] J.J. Rocchio Jr. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[RM87] R. Rada and B.K. Martin. Augmenting thesauri for information systems. *ACM Transactions on Office Information Systems*, 5:378–392, 1987.

[Sal66] G. Salton. Automatic phrase matching. In D.G. Hays, editor, *Readings in Automatic Langhuage Processing*, pages 169–188. American Elsevier Publishing Co., Inc., New York, 1966.

[Sal68] G. Salton. *Automatic Information Organization and Retrieval.* McGraw-Hill, Inc., New York, 1968.

[Sal71a] G. Salton. Experiments in automatic thesarus construction for information retrieval. In *Proceedings IFIP Congress*, pages 43–49. IFIP, 1971.

[Sal71b] G. Salton. *The SMART Retrieval System: Experiments in Automatic Document Processing.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[Sal75a] G. Salton. *Dynamic Information and Library Processing.* Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1975.

[Sal75b] G. Salton. *A Theory of Indexing*, volume 18 of *Regional Conference Series in Applied Mathematics.* Society for Industrial and Applied Mathematics, Philadelphia, 1975.

[Sal86] G. Salton. Another look at automatic text-retrieval systems. *Communications of the ACM*, 29:648–656, 1986.

[SB88a] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. Technical Report TR 88-898, Department of Computer Science, Cornell University, Ithaca, New York, 1988. To appear in the Journal of the American Society for Information Science.

[SB88b] G. Salton and C. Buckley. On the use of spreading activation methods in automatic information retrieval. In *Proceedings of the Eleventh ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 147–160, Grenoble, France, 1988.

[SFV85] G. Salton, E.A. Fox, and E. Voorhees. Advanced feedback methods in information retrieval. *Journal of the American Society for Information Science*, 36:200–210, 1985.

[Sib73] R. Sibson. Slink: An optimally efficient algorithm for the single link cluster method. *Computer Journal*, 16:30–34, 1973.

[SJ71] Karen Sparck Jones. *Automatic Keyword Classification for Information Retrieval.* Butterworths, London, 1971.

[SJ72] Karen Sparck Jones. A statistical interpretation of term specificity and its application to retrieval. *Journal of Documentation*, 28:11–20, 1972.

[SM83] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, Inc., New York, 1983.

[SM86] B. Shneiderman and J. Morariu. *The Interactive Encyclopedia System (TIES).* Department of Computer Science, University of Maryland, College Park, Maryland, 1986.

[Spa80] H. Spath. *Cluster Analysis Algorithms.* Ellis Horwood, Chichester, 1980.

[SS73] P.H.A. Sneath and R.R. Sokal. *Numerical Taxonomy.* Freeman, San Francisco, 1973.

[SVF84] G. Salton, E. Voorhees, and E.A. Fox. A comparison of two methods for boolean query relevance feedback. *Information Processing and Management*, 20:637–651, 1984.

[SWY75] G. Salton, A. Wong, and C.S. Yang. A vector space model for automatic indexing. *Communications of the Association for Computing Machinery*, 18:613–620, 1975.

[SY73] G. Salton and C.S. Yang. On the specification of term values in automatic indexing. *Journal of the American Society for Information Science*, 29:351–372, 1973.

[SYY75] G. Salton, C.S. Yang, and C.T. Yu. A theory of term importance in automatic text analysis. *Journal of the American Society for Information Science*, 26:33–44, 1975.

[UD83] J. Ulmschneider and T. Doszkocs. A practical stemming algorithm for online search assistance. *Online Review*, 7:301–315, 1983.

[Vla83] G. Vladutz. Natural language segmentation techniques applied to the automatic compilation of printed subject indexes and for online database access. In *Proceedings of the Conference on Applied Natural Language Processing*, pages 136–142, Santa Monica, California, February 1983. Association for Computational Linguistics.

[Voo85a] E.M. Voorhees. The cluster hypothesis revisited. Technical Report 85-658, Cornell University, Ithaca, New York, 1985.

[Voo85b] E.M. Voorhees. *The Effectiveness and Efficiency of Agglomerative Hierarchic Clustering in Document Retrieval.* Ph.D. dissertation, Computer Science Department, Cornell University, Ithaca, New York, 1985.

[Voo85c] E.M. Voorhees. The efficiency of inverted index and cluster searches. In *Proceedings of the Ninth International Conference on Research and Development in Information Retrieval*, pages 164–174, 1985.

[Voo86a] Ellen M. Voorhees. The efficiency of inverted index and cluster searches. In *Proceedings of the Ninth International Conference on Research and Development in Information Retrieval*, pages 164–174, Ithaca, New York, 1986.

[Voo86b] Ellen M. Voorhees. Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. *Information Processing and Management*, 22:465–476, 1986.

[vR86] C.J. van Rijsbergen. A non-classical logic for information retrieval. *Computer Journal*, 29:481–485, 1986.

[Wil88] P. Willett. Recent trends in hierarchic document clustering: A critical review. *Information Processing and Management*, 24:577–597, 1988.

[WYB88] S.K.M. Wong, Y.Y. Yao, and P. Bollman. Linear structure in information retrieval. In *Proceedings of the Eleventh ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 219–232, Grenoble, France, 1988.

[WYSB89] S.K.M. Wong, Y.Y. Yao, G. Salton, and C. Buckley. Formalization and evaluation of linear relevance feedback. Technical Report TR 89-992, Department of Computer Science, Cornell University, Ithaca, New York, 1989.