Order Number 8328584

EXTENDING THE BOOLEAN AND VECTOR SPACE MODELS OF INFORMATION RETRIEVAL WITH P-NORM QUERIES AND MULTIPLE CONCEPT TYPES

Fox, Edward Alan, Ph.D.

Cornell University, 1983

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

EXTENDING THE BOOLEAN AND VECTOR SPACE MODELS OF INFORMATION RETRIEVAL WITH P-NORM QUERIES AND MULTIPLE CONCEPT TYPES

A Thesis

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

Edward Alan Fox

August 1983

Biographical Sketch

Edward Alan Fox was born on May 14, 1950 on Long Island, New York and completed his high school education there at George W. Hewlett High School. He received the degree of Bachelor of Science in Electrical Engineering in 1972 from the Massachusetts Institute of Technology. He entered the graduate school of Cornell University in September, 1978 and received the degree of Master of Science in Computer Science in 1981. In August 1982 he began work as the Manager of Information Systems at the International Institute of Tropical Agriculture, Ibadan, Nigeria. Returning to the U.S. in 1983, he accepted the post of Assistant Professor in the Department of Computer Science at Virginia Polytechnic Institute and State University. He is a member of the Association for Computing Machinery, the American Society for Information Science, and the Society of the Sigma Xi.

TO CAROL, JEFFREY AND GREGORY

Preface

Gerard Salton, my thesis advisor, helped pioneer the development of automatic indexing and information retrieval systems. M.M. Kessler, my undergraduate thesis advisor, developed the notion of bibliographic coupling between documents, which was later extended by Henry Small to make use of co-citations as a measure of similarity. Harry Wu proposed the p-norm model as a generalization of the Boolean and vector space models of information retrieval.

This thesis is an attempt to integrate these contributions into a workable framework that can be of practical value to users of future information retrieval systems. An updated version of the SMART system was used for experimental validation of ideas proposed, and hopefully will continue to be a test bed for improved methods of information retrieval.

Acknowledgments

I would like to express my profound gratitude to my thesis advisor, Professor Gerard Salton, for years of support, guidance, kindness, and patient assistance. I also wish to especially thank the other two members of my special committee: Professor Joseph E. Grimes for many discussions about linguistics and information retrieval, and Professor Fred Schneider for teaching me the value of theoretical models in the systems area of computer science. All three were very understanding about complications relating to my move to and return from Nigeria, and about the burden of their helping finalize this very long thesis. In addition, I would like to express my appreciation to Professor Sally McConnell-Ginet who helped introduce me to the complexity of linguistic semantics and represented Professor Grimes during his sabbatical.

Many people and organizations provided data for experimental studies. Document collections were supplied by Jeff Pache of INSPEC - The Institution of Electrical Engineers in England. Robert Dattola at Xerox Corporation, and Henry Small of ISI©, the Institute for Scientific Information©. The INSPEC query collection was provided by William Frakes at Syracuse University. Many individuals contributed queries. My wife Carol helped identify the citations among CACM articles. Jill Warner finished looking up and then typed the CACM citations data as well as most of the ISI abstracts.

Past and present members of the SMART project assisted with program development, collection preparation, helpful discussions, and with the conduct of some of my experiments. Class projects by Bob Harper, Andy Hanuchevsky and Alison Brown helped with data and program preparation. Particular thanks go to Alex Yeh for work on early prototype programs, George Bennet and Conrad Cady for data analysis, Susan Manning for

preparing computer graphs, Bruno Wolf for proofreading, Pauline Tso for relevance judgments, work on SMART statistical programs, and the running of experiments. Elena Siefrid, SMART staff programmer, deserves thanks for her patience as well as praise for dedicated aid of all aspects of the research work. My grateful appreciation goes to Chris Buckley and Ellen Voorhees for several years of thoughtful collaboration and especially for their suggestions and aid during the last year of my thesis preparation.

The final typing and production of this thesis would not have been possible without the assistance of a number of people in the Cornell Department of Computer Science. Geri Pinkham, SMART project secretary, gave ongoing assistance to the group on many matters. Aid from outside was also valuable; Jeff Kurtze formatted Appendix B and Richard Kinch helped with data transfer problems.

The National Science Foundation, through grant IST 80-17589, provided support for these studies, and through grants to Cornell University supplied the computer facilities which made all the experiments possible.

Finally, I wish to express my heart felt thanks to my wife and children, for encouraging me and sacrificing in so many ways.

Table of Contents

CHAPTER 1

INTRODUCTION

An important concern today is how people will be able to locate specific information cut of the vast collections of data now in existence. Various computerized information retrieval (IR) systems provide storage and search facilities so that appropriate selections can be obtained in response to a user's question. Database and question answering systems both perform these functions. In this thesis, database methods will be frequently referred to while question answering will only be occasionally mentioned.

At the risk of overloading vague and often misused terminology, the phrase "information retrieval" will also refer to a third type of system, one where some item such as a book, an article, a paragraph, a business letter, or a message is retrieved in response to a particular query. It is this somewhat specialized type of information retrieval system that will be the focus of subsequent discussions. Usually, items retrieved by such a system contain some textual elements, and retrieval is based at least in part on matches or partial matches between query terms and terms in the "document". Hence, if there is ambiguity between the two uses of IR, the qualifier "textual" will be added to help distinguish the specialized from the general meaning. Many adaptations and applications of methods used in textual IR systems are appropriate in other related types of systems, so these will be touched upon in later chapters. However, the emphasis will generally be upon integrating and extending document handling methods presently used in commercially available systems with those being developed and tested in laboratory settings.

In subsequent sections of this chapter, background material for the rest of the thesis is given. Information retrieval (textual) is distinguished from database and question answering, and then the "Boolean" and "vector space" approaches to IR are described. Extensions to those approaches have been proposed in a piecemeal basis over the years: using bibliographic information, connecting with factual information, and most recently, allowing "p-norm" queries.

In this chapter the various extensions are introduced only briefly however, since many of them, though simple conceptually, need a fair amount of elaboration and discussion to be clearly understood by the lay person. Consequently, the last section of this chapter is given as an outline for the rest of the thesis – a sort of annotated table of contents. That is particularly useful for two reasons. First, the reader may wish only to look at certain parts of the work that more closely relate to a given interest. Secondly, certain methods developed as part of the general investigation have turned out to be rather successful in experiments conducted and so may be worth studying on their own. To mention several: automatic construction of Boolean queries based on feedback information, the use of regression to identify proper parameter settings for retrieval, and the modification of query vectors through relevance feedback of document representations which include multiple concept types.

All of the above points will be amply explained in the subsequent text or in the many references cited. It is hoped that the casual reader will be able to grasp the essential points, to pursue further reading if there is interest, and where appropriate, to skip technical elaborations or details. Specialists in the area of information retrieval should be able to follow all of the text, or to skip about to relevant sections after surveying the preliminaries. But

first, it is crucial to see what is meant by information retrieval as a general field and by the three subfields of database, question answering, and (textual) information retrieval.

1.1. Approaches to Information Retrieval

1.1.1. Database Systems as Background

Since the late 1950's computers have been employed to record and later provide retrieval capabilities for textual, numerical and other types of data, stored and organized in many different ways on large capacity "secondary" memory devices (such as disk drives). Database management systems are now commonly available which provide storage, manipulation and retrieval functions on data that is typically well organized according to a predefined plan or schema [Date 1982]. Simple numeric or alphabetic items such as "Salary" or "Name" are just a few of the attributes that might be defined in connection with a person; other sets of attributes are readily chosen for entities such as "Project" or "Department." Recent studies have explored the feasibility of extending such systems to accommodate more complex objects, such as a chart of data, or a memorandum, which might have fixed fields (e.g., "Author" and "Date") as well as text [Haskin & Lorie 1981].

A simple partial solution is that described in [Dattola 1979] where a database management system (of the "CODASYL network" type) is used to store bibliographic facts, supplementing a specially designed "vector space" component designed for document retrieval. However, other types of database systems, especially relational [Codd 1970] ones, are easier than network types to directly adapt to the needs of (textual) information retrieval [Macleod 1979, Crawford 1981].

Relational systems are basically aimed at manipulating and interrelating tables of data, each made up of many rows and a number of clearly defined columns. The original relational approach has been the focus of various proposals for extension to more realistically model the real world without inappropriate structural restrictions on data organization [Kent 1979]; one aim is to capture more of the meaning [Codd 1979] through the data and its representation. One such recommendation geared toward utility for information retrieval is where the programming language concept of an abstract data type is used to generalize beyond the simple data types usually employed, and special operators allow flexible high level handling of resulting complex structures [Fox 1981].

1.1.2. Question Answering Systems

In recent years some (textual) information retrieval researchers have considered how artificial intelligence work relates to their area ([Walker, Karlgren & Kay 1977], [MacCafferty & Gray 1979], [Salton 1979]) and artificial intelligence researchers have tried to apply their work to retrieval problems ([Sager 1975], [Bolc 1978], [Schank 1980]).

The broad field of artificial intelligence is concerned with representation of knowledge and with intelligent processes ([Bobrow & Collins 1975], [Schank & Abelson 1977], [Findler 1979]). Thus, to answer questions, the relevant information must be represented, such as in structured databases, in one of many types of semantic nets, via (extended) predicate logic, or using ad hoc formalisms. In some systems the interrelationship of process and data is exploited and the knowledge representation has inherent active components [Small,S. 1980].

In general though, question answering requires two separate active phases, that of syntactically and semantically analyzing the question and that of searching and locating the desired data in the knowledge base [Salton & McGill 1983 Chapter 7]. Many systems have

been devised that understand certain subsets of natural language, construct some internal representation of the question, and then conduct a search ([Plath 1976], [Harris 1977], [Codd et al. 1978], [Hendrix et al. 1978], [Martin 1978], [Waltz 1978]). The key search issues are those of finding an exact match, of inferencing from related facts to construct an answer, or of pinpointing sections of the knowledge base that relate. Except when exact matching in a database-type organization is needed, question answering systems are typically not geared toward handling hundreds of thousands or millions of documents. It remains to be seen when or whether detailed knowledge representations of text and question answering systems will be adaptable in that environment.

1.1.3. Textual Information Retrieval Systems

Textual information retrieval systems, referred to as IR systems when there is no ambiguity, are often called document retrieval systems when used for storage and retrieval of various kinds of documents. These have generally dealt with the handling of large volumes of textual information, such as collections of published documents in certain fields such as chemistry or psychology. In response to a suitably phrased query, the title, abstract or full text is retrieved for the user. While some separation is often made between the fixed fields, like author and publication name, and the main text of the document, there are typically only a small number of predefined attributes associated with the document (e.g., STAIRS system [IBM Corporation 1979]). The body of text is dealt with as a whole, as far as the user is concerned; and the system is responsible for selecting the proper documents, often based on the matching of some internal form of the query with some internal form of the document.

Some IR systems store the full text of the various documents or other items being manipulated. In addition to being able to locate documents of interest, the user may be able to retrieve and/or examine paragraphs, passages, sentences, or single word occurrences (in context). Since these extra capabilities are straightforward generalizations of document retrieval methods, they will not be considered in detail. It should be clear, however, that additional precision in specifying a query is possible, such as when a user requires that in some paragraph "regression" and "linear" be present and that the document also has the phrase "information retrieval system." The Responsa project has demonstrated how such methods can be implemented and applied to certain types of in-depth document analysis [Attar & Fraenkel 1981].

Whether full text or just a small number of descriptors are used to characterize a document, the type of query form employed in many commercially available systems resembles that of a Boolean expression. In certain cases the "AND" operator (designating co-occurrence on the document level) is replaced by other operators indicating co-occurrence within a paragraph, sentence, or other unit based on a specified measure of distance. Though such "metrical" operators are more specific than a simple "AND", for the sake of simplicity systems with such capabilities will be referred to as "Boolean systems" to distinguish them from "vector systems."

1.1.4. Boolean Logic Implementations

In Boolean information retrieval systems, the burden of taking the original query, typically submitted in the natural language as a statement of one's interest, and making the transformation to a form employing Boolean connectives (e.g., "Information AND Retrieval"), rests upon the user or, more commonly, upon a search intermediary trained in

the subject area and in search formulation techniques. Often, too, trained indexers laboriously analyze new documents being added to the collection and assign special identifiers that become part of the document representation [Borko & Bernier 1978].

1.1.5. Vector Space Model

Since the early 1960's an alternate approach, based on automatic techniques for query construction and document indexing, has been the subject of extensive research. Comparisons with Boolean systems have shown these vector-based methods to be at least as effective [Salton 1972b], and yet they are not in widespread use as of the present. Perhaps with proper generalization, that situation will change.

1.1.6. Need for Generalization

In future years, information retrieval systems will become more commonly available, and will be accessed in offices as well as libraries. However, there is need for a generalization of the Boolean and vector approaches to provide the additional functionality required. The next section focuses on weakness of these approaches, in order to determine how they should be further developed to provide more effective information retrieval capability.

1.2. Current Approaches and Limitations

1.2.1. Boolean Systems

1.2.1.1. Availability

The 1982 edition of *Computer-Readable Databases* [Williams 1982] advertises over 750 databases available, almost all searchable using Boolean systems. The INSPEC database alone, part of which was used for experiments described in this thesis, has over 1.6 million

records searchable through the DIALOG Boolean system.[1] Tutorials like [Meadow & Cochrane 1981] describe many of the commonly used document collections and commercially available search services, and serve as guides to the proper operation of systems which interpret Boolean logic statements. Advice is given on query formulation, search strategies, and the use of enhancements to the simple Boolean expressions normally used – so Boolean systems can become more effectively and more widely utilized. In spite of their ubiquitous nature, however, Boolean systems have a number of disadvantages and limitations. Some of these are outlined below; for another list the reader is referred to [Salton, Fox & Wu 1982].

1.2.1.2. Indexing

The typical Boolean system stores documents that have been assigned key words or descriptors by trained indexers. The indexing task is a difficult and often tedious one; techniques are taught in graduate programs and in texts such as [Borko & Bernier 1978]. If systems were employed which would make this process unnecessary, great savings in manpower expended could result and job satisfaction of many skilled librarians might increase.

In certain environments, manual indexing is not feasible because of privacy or other considerations. In the case of an electronic mail system which handles office information, it is impractical to employ an indexer to read all the electronic mail and assign index terms. Authors may be willing to assign keywords, when they remember to do so; but it is unlikely that their patterns of keyword assignment will be consistent with those of other authors or searchers.

---

[1] DIALOG (Trademark) sheet describing their INSPEC collection, February 1981.

Recently, more and more Boolean systems have allowed searching the full text of documents. However, providing sufficient storage to make this possible is still expensive. Though algorithms and special hardware for fast string matching have been developed (e.g., [Boyer & Moore 1977], [Haskin 1980]), searching huge databases in that fashion is still time consuming. In addition, without a thesaurus or other aid, choosing the proper query terms in a free text environment can be very difficult.

1.2.1.3. Inverted Files

Boolean systems typically operate by first having the computer retrieve a list of document pointers or identifiers from a so-called inverted file, for each term in the query. These lists are then restricted using intersection for AND connectives, merged into a union for OR operators, or shortened by set subtraction for AND NOT constructions. For short queries, these operations are very efficient; for long queries other methods may be more appropriate [Salton & Wu 1981].

Typically, inverted files are static structures, and so they must periodically undergo a costly reorganization operation. In the dynamic environment of an office, such reorganization of active message files could be very awkward. This situation can be resolved, however, by the proper choice of more complex data structures (e.g., B-trees [Bayer & McCreight 1972]), though at the cost of some extra overhead in space required.

As already mentioned, in some systems, to provide added precision in describing queries, the AND operator is replaced by a "within sentence" or "within n words" connector [Meadow & Cochrane 1981]. These metrical operations require more complex storage structures than those present in a simple Boolean system; for an interesting formal treatment see [Attar & Fraenkel 1977].

1.2.1.4. Query Formulation

Another disadvantage of Boolean systems is the difficulty involved in formulating good queries. Searchers must be trained, and even then there is considerable variability among the abilities of different searchers [Katzer et al. 1982]. There is even a wide variation in performance between queries submitted by a single searcher. All of this is not surprising, since a search intermediary must: 1) understand someone's question from an often ambiguous statement of interest, 2) guess what vocabulary is used in unknown relevant documents, and 3) keep in mind the frequencies of terms, truncated terms, and combinations of terms in preparing a query that will select just the right number for retrieval.

Further, in more complex systems, deciding which of many possible operators to use in a given case can become very complicated (i.e., should "information" and "retrieval" be adjacent or could they instead simply be in the same sentence such as when occurring in the phrase "retrieval of information").

1.2.1.5. Presentation of Results

Boolean systems respond to queries by indicating the size of the set of documents satisfying the query and allowing printing of elements of that (unordered) set. The formulation process must be repeated if the set is too large or too small or if more documents are desired after scanning over the initial retrieved set. No ranking of the output is provided in most cases – a serious problem if the query retrieves more than a few dozen articles. Some systems like Stairs [IBM Corporation 1979] have offered simple procedures for ranking the retrieved set; probably they are not more widely used because the ones chosen there are not particularly effective ranking procedures [McGill, Koll & Noreault 1979].

1.2.1.6. Boolean Logic

In addition to the problem of query formulation, there is the issue of how appropriate Boolean logic is to serve as the basis for describing queries. While it may be possible to know the meaning of a sentence based upon knowing the conditions that make the sentence true [Kempson 1977 page 23], it is doubtful whether the characteristic function [Zadeh 1965] for typical queries precisely identifies exactly the set of articles desired by a user.

Common sense reasoning suggests that sometimes implementing "A AND B" via intersection would retrieve more than was really desired, perhaps because the context of joint occurrence of A and B in a document was not that desired. On the other extreme, it is often the case that interpreting Boolean operators according to their strict definition is an inappropriate requirement for retrieval of certain of the documents relevant to a given query. Thus, term A and a synonym of B might appear together in a document that nevertheless did not satisfy the query "A AND B."

[Verhoeff, Goffman & Belzer 1961] objected that if the logical operators AND and OR are implemented using set intersection and set union, then "A AND B" would retrieve more than average users want, while "A OR B" might leave out desired material. Their argument was based on the assumptions that users differ in their perspective of what is relevant to a given query and that the performance of a system is measured by a linear equation considering judgments of each such user. Averaging over those users, one can assign probabilities of the relevance of documents to queries, and can determine a cutoff point for retrieval that will optimize the measure of overall user-wide query satisfaction. It is then not necessarily the case that, for example, just because a document containing both A and B should be retrieved in response to those two single term queries, that users would also

concur sufficiently to warrant that it should be retrieved in response to query "A AND B".

In recent years, various efforts have been made to extend Boolean logic; some linguists, for example, have studied many-valued and fuzzy logics [McCawley 1981 page 360]. The strictness of AND and the looseness of OR are tempered by defining valuations which may also map onto one or more intermediate values between TRUE and FALSE (see for example [Sanford 1976], [Kaufman 1977], or [Kay & McDaniel 1978]). In addition, the imprecision of assigning index terms to documents can be modelled through the use of membership functions which supply the same sort of quantitative information as do weights on vector elements. Using such values, retrieved sets of documents could be ranked according to a membership function which gives the presumed degree of relevance to a given query.

Though theoretical papers in the information retrieval literature have dealt with these and related issues, often using fuzzy set theory as a basis for a more flexible interpretation of the Boolean connectives (e.g., [Tahani 1976], [Robertson 1978], [Radecki 1979, 1982], [Waller 1979], and [Bookstein 1980]), this problem has not been thoroughly investigated. That is in part why the p-norm model, which generalizes concepts of fuzzy set theory in a form particularly appropriate for retrieval tasks, is described in detail in subsequent chapters which cover analytical and experimental findings.

1.2.2. Vector Models

1.2.2.1. Automatic Indexing

Before listing some limitations of vector methods, the basic techniques should be explained. Typically, a user's natural language query is automatically indexed, just like the documents, to obtain a vector with weights. If the terms present in documents are num-

bered 1 to T (e.g., T=10,446 for the *Communications of the Association for Computing Machinery,* – *CACM* – collection of 3204 abstracts), then a document can be represented by a vector with 1's in places where terms are present in the document and 0's elsewhere. In the numbering system of Cornell's collection of abstracts from the *CACM*, the article by Verhoeff et al., "Inefficiency of the Use of Boolean Functions for Information Retrieval Systems," referred to in section 1.2.1.6, could then be represented as the alphabetized list of significant words

(Boolean, functions, inefficiency,
information, retrieval, systems, use)

or by the following bit vector with the lower line labelling term numbers of stems that are present.

```
0...00100...0100...0100...0100...0100...0100....0100....00
|    |     |    |    |    |    |    |    |    |    |
1 1521  3964  4683  4700  8067  9282 10110 10446
```

Now, weights are typically applied to these vectors: common words like "use" (term 10110), which occurs in 685 of the 3204 articles stored, receive a low weight according to their inverse document frequency (idf) [Salton 1975 page 443], e.g.,

$$idf_i = \log_2\left(\frac{\text{no. postings for most frequent term} + \epsilon}{\text{no. of docs. with } i^{th} \text{ term}}\right) \tag{1-1}$$

$$idf_{use} = \log_2\left(\frac{\text{number of postings for the most frequent term} + \epsilon}{\text{number of documents with the term "use"}}\right)$$

$$= .94$$

where $\epsilon = 0.001$

For the title of interest, once very common ("stop") words are removed, automatic indexing yields Table 1.1, showing frequency (i.e., postings = count of the number of documents containing the term) and idf values.

In general the i^{th} document D_i can be thought of either as the full vector

$$\vec{D}_i = (d_{i1}, d_{i2}, \cdots, d_{iT}) \tag{1-2}$$

where there are T terms and where d_{ij} is the weight of term j in the i^{th} document. Since most weights are zero, a condensed form is:

$$\vec{D}_i = (<c_{i1}, d_{i1}>, \cdots, <c_{ip}, d_{ip}>) \tag{1-3}$$

where c_{ij} designates the j^{th} term out of the p present in the i^{th} document and d_{ij} is, as before, the associated weight for c_{ij}.

Table 1.1: Indexed Title Example

Title: "Inefficiency of the Use of Boolean Functions for Information Retrieval Systems"

Indexing Output:

Word	Concept Number	Frequency	Idf Weight
Boolean	1521	32	5.36
functions	3963	339	1.96
inefficiency	4683	9	7.19
information	4700	253	2.38
retrieval	8067	94	3.81
systems	9282	688	0.94
use	10110	685	0.94

Hence, the example document given in Table 1.1 can be represented as

(<1521,5.36>, <3963,1.96>, <4683,7.19>, <4700,2.38>,
<8067,3.81>, <9282,0.94>, <10110,0.94>).

1.2.2.2. Ranked Retrieval

To decide if two vectors, e.g., a document $\vec{D}_i$ and a query $\vec{Q}_j$, are closely related, a similarity function can be employed such as [Salton 1975 page 121]

$$SIM^{COS}(\vec{D}_i, \vec{Q}_j) = \frac{\sum_{k=1}^{T} d_{ik} \cdot q_{jk}}{(\sum d_{ik}^2 \cdot \sum q_{jk}^2)^{1/2}}. \tag{1-4}$$

For a query $\vec{Q}_j$ the computer system can produce a list of all documents in the collection, ranked in decreasing order of the cosine query-document similarity $SIM^{cos}(\vec{D}_i, \vec{Q}_j)$. Indeed, this is one of the major advantages of vector over Boolean systems [Salton 1975 page 18].

1.2.2.3. Limitations

With this background, one can readily see some of the limitations of vector techniques. First, there is the inherent difficulty of using a "flat" vector representation. Though a vector element can represent a stem, a word, a phrase, or a thesaurus category, that element is a fixed unit and has no defined association with the other elements of the vector. Indeed, many vector methods assume the terms are independent of each other, which is clearly not the case [Van Rijsbergen 1979 page 120]. Indeed, when the co-occurrence among concepts is used to estimate the relationship between terms [Harper & Van Rijsbergen 1978], queries in a special form (i.e., mathematical formula representing dependency trees) can often be constructed which give better performance than do vector queries.

Secondly, the particular similarity computation mentioned above is essentially additive, ignoring, in weighted systems, how many matches are responsible for arriving at the final similarity. Thus, consider the terms in Table 1.1. The weights of "information", "retrieval", and "systems" sum to 7.13, which is less than 7.19, the weight of the single term "inefficiency". Hence, there is the undesirable situation that one term is preferred to three, while those three together define a much more specific context. This problem is avoided when binary weights are assigned to terms, but that causes other difficulties. Thus, when binary weights are used, a simple count of the number of matching words with the query "systems that use Boolean functions" would favor documents containing the vague pair of terms "system" and "use" over those just containing "Boolean". In spite of these exceptions linear models do tend to work well in average situations so these problematic cases with similarity functions can be ignored, especially if queries are long enough to provide a proper context by having at least a few terms in common with most relevant documents.

Finally, though some early studies showed the benefits of having bibliographic concepts included in the vectors [Salton 1971b], little in practice has been done about that generalization. This matter will be considered in more detail below.

1.3. Value of Bibliographic and Factual Information

In an early study by Salton [1963], the use of bibliographic information was explored in the context of extending vectors for associative retrieval methods. Benefits seemed likely on the basis of tests made with an automatically generated pseudo-collection.

In a later study [Salton 1971b], vectors were extended to include citations to documents. Some 42 documents were chosen from which to generate queries, so that citation

data would be available in the queries. Citation information did enhance retrieval, especially when the citations were relevant to the subject matter of the query.

A further study [Michelson et al. 1971] employed bibliographic data in the context of relevance feedback (i.e., after a short initial search, the user selects relevant documents, and then those are used to extend and improve the query), and also showed benefits from the use of author and citation information. Though the tests were made with the very small ADI collection, which had few citations, and though improved feedback techniques have since been developed, this use of the SMART system did suggest that further development of citation feedback would be worthwhile.

Another form of bibliographic information, that of the bibliographic coupling between articles, was initially investigated by Kessler [1962]. Though not in the context of vector methods, Kessler defined bibliographic coupling [Kessler 1963a]; i.e., when articles A and B both refer to an earlier article such as C, which is present in each of their bibliographies, then A and B have one unit of coupling. This measure was used by Kessler to group together documents into categories, and, accordingly, can be used to determine similarities between documents. Kessler even utilized some of this information in the TIP system developed at MIT [Kessler, Ivie & Mathews 1964 and Kessler 1965a]. Weinberg [1974] reviews these early studies of bibliographic coupling.

Since bibliographic coupling identifies connections between articles based solely on bibliographies, there may be similar articles in a collection, written in different time periods, which have little or no coupling. Further, even if subsequent developments cause researchers to view a pair of articles as similar and hence refer to both of them together, there will be no change in the bibliographic coupling. However, another measure, co-citation

strength, was defined later and does reflect the viewpoints of later authors.

Garfield [1979] launched the development of citation indexes, listing for each article (that has been cited) all the articles which cite it. At the Institute for Scientific Information©, (ISI©), Garfield, Small and others have developed large files with citation information. Small and Koenig [1977] used bibliographic coupling to cluster journals, but even earlier Small proposed the definition of a measure of document similarity based upon co-citations [Small 1973]. Thus, the similarity of two articles is computed based on the number of other articles that cite both of them together.

Many studies have utilized co-citation counts for clustering documents [Salton & Bergmark 1977], journals [Carpenter & Narin 1973], and authors [White 1981]. Bichteler and Eaton [1980] proposed the combined use of bibliographic coupling and co-citation for document retrieval, and gave preliminary evidence as to its value.

Thus, the early studies of incorporating bibliographic information, the tests of extending vectors with citations, and various results with bibliographic coupling and co-citation measures, all suggest that each of these types of information does provide data useful for retrieval. The question that remains, then, is whether they can be easily combined into a single document representation such that retrieval system performance with such an expanded form is better than performance of representations which use only the terms present in documents.

1.4. Extensions to Earlier Approaches

Extensions to the Boolean and vector space approaches are motivated by the desire to develop an integrated system providing the good features of existing methods while at the same time avoiding some of the limitations and disadvantages present. P-norm queries

allow a generalization of both the Boolean and vector forms of queries. A parameter p can be varied so that queries behave as Boolean, vector, or any of a continuum of in-between forms. Furthermore, Boolean logic structure can be expressed and term weighting methods can be employed. Retrieval yields a ranked output, so all of the key advantages of both Boolean and vector systems are present.

The use of multiple concept types to generalize the vector representation of documents provides a second method for performance improvement. By including more information in the document representation and by judiciously utilizing that information through the relevance feedback cycle, improved retrieval can result. Thus, in one system the benefits of both keyword and citation searching can be enjoyed.

1.5. Experimental Approach

The aim of this thesis is to demonstrate the value of various applications of the p-norm model, to empirically confirm ideas as to the best ways of using those techniques, and to show that the newly proposed multiple concept type document representation scheme improves upon regular vector methods. Consequently the work reported here has involved a good deal of (document) collection and program development followed by running of experiments with the aid of computers. Though some have claimed that experimental computer science is becoming less common [Feldman & Sutherland 1979], Cornell University with funding provided by the National Science Foundation has since 1981 acquired very effective computer resources which have been heavily utilized by the SMART group for information retrieval experiments.

In many respects, the SMART approach used since the 1960's [Salton 1980] has been adopted almost unchanged. However, new document collections, more generalized

programs, and more convenient summary statistics from test runs, all had to be developed to accomplish the aims of this research study.

Given some way of recording data about documents in a collection (i.e., a document representation scheme), a way of describing users' questions, and an algorithm for retrieving and possibly ranking documents for each query, the computer carries out the retrieval function. When judgments are available from subject experts as to the relevance of documents to each question, the effectiveness of the entire storage and retrieval system can be determined.

Though many measures have been proposed and a number have been computed for some of the test runs made, the recall and precision chart has been the primary tool used by SMART personnel for many years [Salton 1968]. Since so many tests have been made, and since common regression methods require a single dependent variable, a single statistic has been adopted in many cases to summarize the recall-precision behavior.[2]

Thus, hundreds of experimental tests have been made using the modernized and enhanced SMART system, and summary statistics as well as detailed behavior have been considered in evaluating results. All important ideas have been tried out using one of a number of new or adapted document collections. Procedures for determining settings of most important parameters have been demonstrated. Thus, the adaptability of techniques used and the soundness of conclusions reached are not likely to be in question.

---

[2]Recall and precision are defined and explained in Chapter 3. The single statistic mentioned is the average of the precision values for recall levels 0.25, .50, and .75.

1.6. Thesis Outline

Chapters 2 through 5 develop the p-norm model while Chapters 6 through 8 develop the extended vector model. Sections of Chapters 7 and 8 deal with the p-norm model in the context of extended vectors, and Chapter 9 ties everything together.

The appendices provide much useful information and clarify many points made in the thesis. The charts in Appendix A should certainly be glanced at, to provide some intuition regarding p-norm behavior. Appendix B, however, is only for the more mathematically inclined; it gives proofs for theorems stated in Chapter 2 about p-norm query ranking behavior. Appendix C highlights the characteristics of the various collections used.

The reader may wish to read only certain portions of this thesis. The theoretical contributions are concentrated in Chapters 2 and 6 but additional important discussions occur at the beginning of Chapters 4, 5, 7 and 8. Experiments are discussed toward the ends of chapters, especially in Chapters 3, 4, 5, 7 and 8. Clustering is discussed only in Chapter 7, and feedback only in Chapters 5 (for Boolean and p-norm queries) and 8 (for extended representations).

It is hoped that with this document description the reader will find and read those sections of particular interest and not be turned away by discussions considered irrelevant (having low similarity to the perceived information need).

CHAPTER 2

P-NORM QUERIES: THEORY

2.1. Wu's Early Development

2.1.1. Background

Searches in many Boolean systems result in sets of documents being identified at each step along the way. The Boolean operators AND, OR and NOT thus correspond to the set operators for intersection, union, and difference, respectively. Documents have membership in various sets; Boolean queries aim at describing precisely the desired set of relevant documents [Artandi 1971].

Fuzzy set theory [Zadeh 1965] generalizes the notion of set membership to be described by a real valued (in the range of zero to one) membership function. Proposals such as those of [Tahani 1976, 1977] suggest using fuzzy set theory in information retrieval; numerous theoretical and argumentative papers have dealt with this subject (see discussion in 2.1.2 below). Wu's p-norm model resolves many of the dilemmas discussed, and will be introduced after the original fuzzy set model is more fully explained.

According to fuzzy set theory, the concept of "tallness" would be described mathematically by a membership function mapping peoples' actual heights according to conventional opinion to the interval [0,1]. Assuming adults were being described, a height of four feet (122 cm.) might have an associated "tallness" value of 0.0 while eight feet (244 cm.) would probably be assigned 1.0. A height of five foot ten inches (178 cm.) would perhaps have a "tallness" value of 0.7.

This notion applies to document retrieval if we assume, as is done in weighted systems, that a value can be given to the usefulness of a term in being assigned to index a document. By considering document and collection characteristics, the document membership values of terms can be determined. Any term weighting method will work as long as the final values are normalized to be in the unit interval [0,1]. Probabilities, though technically inappropriate since the defining axioms differ from those of fuzzy set theory, or scaled values devised by other formulas, might then be used for document membership functions.

While membership functions simply formalize the automatic indexing notion of assigning weights to terms, without the added restriction of having to satisfy the axioms of probability theory, the related issue of the interpretation of Boolean connectives in a fuzzy set model is very controversial [Robertson 1978]. The standard definitions yield, in the notation introduced by Chapter 1, for document $\vec{D} = (\ <A,d_A>,\ <B,d_B>\ )$:

$$SIM\ (A\ \ OR\ \ B,\ \vec{D}) = \max\ (d_A,d_B) \tag{2-1}$$

$$SIM\ (A\ \ AND\ \ B,\ \vec{D}) = \min\ (d_A,d_B) \tag{2-2}$$

Note that these interpretations are at variance with those of probability theory. Thus, assuming term independence (which really is not warranted), if

$$P(A) = d_A$$

$$\text{and } P(B) = d_B$$

$$\text{then } P(A\ \ AND\ \ B) = P(A,B) = d_A \cdot d_B \tag{2-3}$$

which is very different from equation (2-2) above when binary weights are not required.

A second extension to standard Boolean queries is that of having relative weights assigned to terms and clauses. Thus, a searcher presented in a recent experiment with the

query statement "testing automated information systems," decided that in the ADI collection of articles on documentation, "testing" would provide the most discrimination (i.e., by its presence or absence it would best separate the relevant from the non-relevant articles) and should be weighted twice as much as the other terms. The resulting weighted query was then:

< testing, 2.0 >

AND

(automated OR information OR systems)

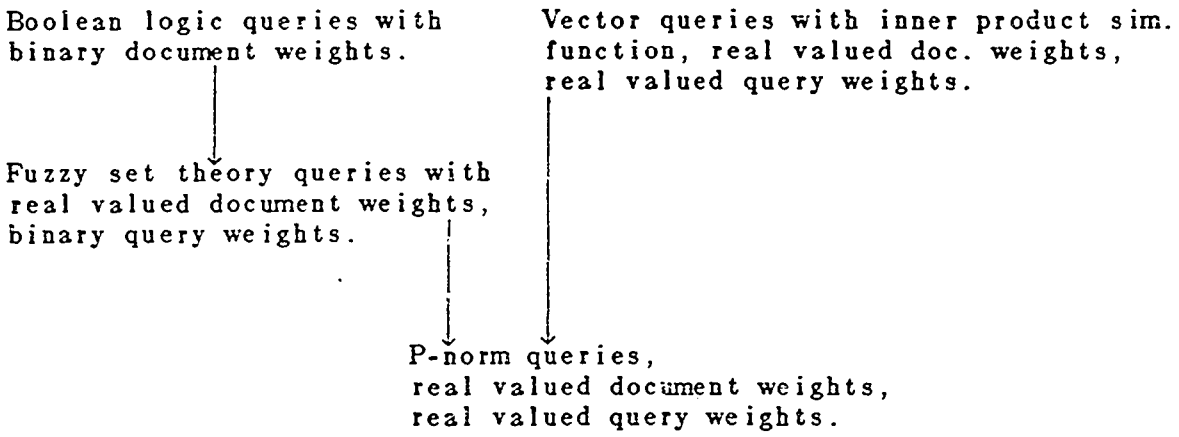
This may well not be the most appropriate formulation of the given query but at least serves to show one line of reasoning on using weighted queries.

The model proposed by Bookstein [1980] included provisions for handling both fuzzy set interpretation of operators and the incorporation of relative weights mentioned above. A more general model was proposed by Wu [1981] which goes beyond the fuzzy set interpretation and encompasses vector methods. Wu's p-norm model is explained in the next several sections.

2.1.2. Origins

Since the later part of the 1970's, a number of papers such as [Tahani 1976, 1977], [Radecki 1977, 1979], [Robertson 1978], [Waller & Kraft 1979], and [Bookstein 1978, 1980] have appeared, discussing the value of fuzzy set theory for information retrieval. Shortly thereafter, Wu devised the p-norm model which generalizes and unifies the Boolean, fuzzy set and vector space models. His perspective on the role of p-norm formulations is shown in Figure 2.1.

Figure 2.1: (Adapted from [Wu 1981])
Relationship of P-norm Queries to Other Schemes



The fundamental insight of p-norm theory is that of relating the norms employed in numerical analysis with the membership functions of fuzzy set theory and the similarity measures of information retrieval; the next section gives the relevant formula along with a brief explanation.

2.1.3. Definitions

In vector space theory, the concept of "norm" is introduced to formalize distance notions. A common class of norms is the l_p norm, defined [Ortega 1972 page 16] for

$p \in [1,\infty]$ and $\vec{X} = (x_1, \ldots, x_n)$ as

$$||\vec{X}||_p = (\sum_{i=1}^{n} |x_i|^p)^{1/p}. \qquad (2\text{-}4)$$

Wu defined the OR operator in terms of distance from the vector space point which indi-

cates absence of all terms, namely,

$$\vec{0} = (0, \ldots, 0).$$

Next, the AND operator was defined in terms of closeness to the point

$$\vec{1} = (1, \ldots, 1).$$

Finally, for Q_{NOT} = NOT (A), the definition is

$$SIM\ (Q_{NOT}, \vec{D}) = 1 - d_A. \tag{2-5}$$

Wu's general definitions allow queries with many terms as well as indications of the relative importance of each of those terms. With

t_i = the i^{th} query term (or more generally, expression)

q_i = the relative weight of t_i in the query.

then for queries of form

$$Q_{OP(p)} = OP^p\ (<t_1, q_1>, \ldots, <t_m, q_m>) \tag{2-6}$$

and documents of form

$$\vec{D} = (<t_1, d_1>, \ldots, <t_n, d_n>) \tag{2-7}$$

the key defining equations are

$$SIM\ (Q_{OR(p)}, \vec{D}) = \left[\frac{(q_1)^p\,(d_1)^p + \cdots + (q_m)^p\,(d_m)^p}{(q_1)^p + \cdots + (q_m)^p}\right]^{1/p} \tag{2-8}$$

$$SIM\ (Q_{AND(p)}, \vec{D}) = 1 - \left[\frac{(q_1)^p\,(1-d_1)^p + \cdots + (q_m)^p\,(1-d_m)^p}{(q_1)^p + \cdots + (q_m)^p}\right]^{1/p}. \tag{2-9}$$

2.2. Equal Similarity Contours

2.2.1. Examples

Since the p-norm definition is based upon the use of distance according to the l_p norm, it is appropriate to turn to graphical portrayals to provide insight into the various complexities of this new formulation. In the two-dimensional case, where documents and queries are here described solely by terms arbitrarily called A and B, the unit square completely includes all points in the vector space. Since documents are characterized by their membership function values, e.g. $\vec{D} = (d_A, d_B)$, each document is represented by a point on that unit square.

The values d_A and d_B can be determined in various ways. A simple definition based upon relative term frequencies in documents alone is

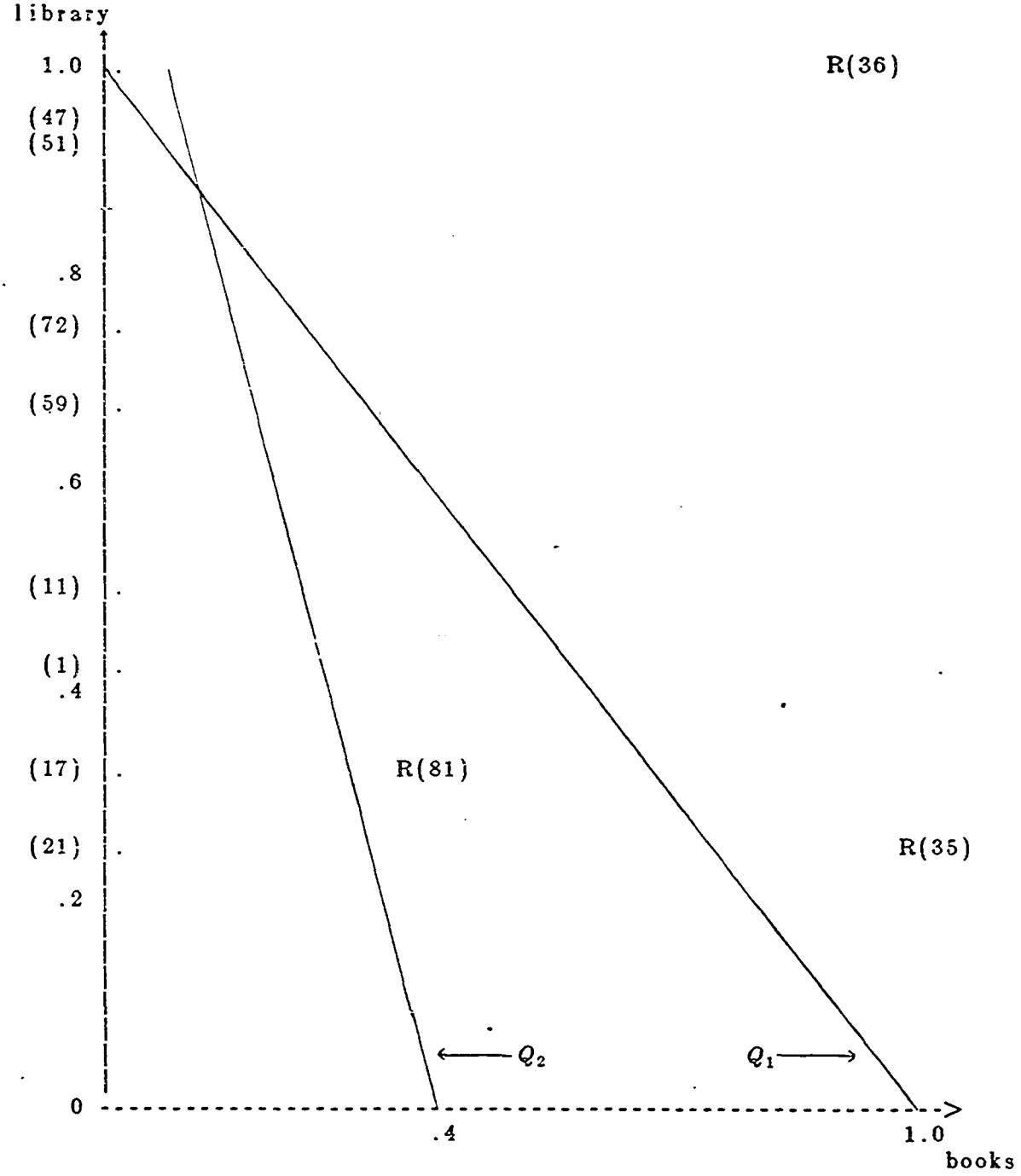
$$d_A = \frac{\text{frequency of term A in the document}}{\text{frequency of term that occurs most often in the document}} \qquad (2\text{-}10)$$

so that if some term occurs five times in the document and the only other term occurs three times, their membership function values will be 1.0 and 0.6, respectively.

To illustrate this situation, drawings were made for the two term query "library books". In Figure 2.2, the two dimensions correspond to terms "books" on the X-axis and "library" on the Y-axis. ADI documents relevant to the query are marked by an "R" adjacent to the document identifier. Non-relevant documents with positive similarity to the query all fall along the Y axis; their identifiers are shown near the proper location.

Two lines have been superimposed upon the diagram to illustrate the role of similarity functions in separating relevant from non-relevant documents. Assume that one ranking method retrieves documents above line Q_1 while another retrieves those above Q_2. Clearly,

Figure 2.2: Two Term Space for Query ''library books''



the rule utilizing Q_2 will be superior, since relevant document 81 is retrieved while non-relevant documents 47 and 51 are disregarded. The p-norm similarity functions can be thought of in terms of sets of lines like Q_1 and Q_2. Actually, a curve exists for each possible similarity value in the range [0,1] since a p-norm formula defines a family of iso-similarity contours.[1] Figures 2.3 and 2.4 illustrate this fact by giving a few contours for two different formulations of the query "library books".

Utilizing the relative weight capability of the p-norm model, the query "library books" can be stated so as to emphasize "books" much more than "library,", e.g., in the ratio 3:1; when $p = 1$ a query with those relative weights is given by

$$< \text{books}, 3 > \ AND^1 \ < \text{library}, 1 > .$$

Contours for this specially concocted illustrative query are shown in Figure 2.3 such that any point on each line has equal similarity to the query. Points above and to the right of a line have higher similarity values while points below and to the left have lower similarity. The example query separates the relevant from the non-relevant documents; utilizing the retrospectively chosen relative weights yields an optimal ranking.

The same performance achieved in this example can result without the use of relative weights if the p-value is varied instead. Clearly the query indicates that documents with the presence of both terms "library" and "books" should be preferred so the appropriate operator is AND^p with $p > 1$. Figure 2.4 shows contours for the AND^2 case. As with the relative weight formulation this retrospectively chosen p-norm formulation gives optimal performance.

---

[1]These contours connect points which could represent documents. All documents on one contour have the same similarity to the given p-norm query.

Figure 2.3: Contours for Query ''books$_3$ *AND*^{1} library$_1$''

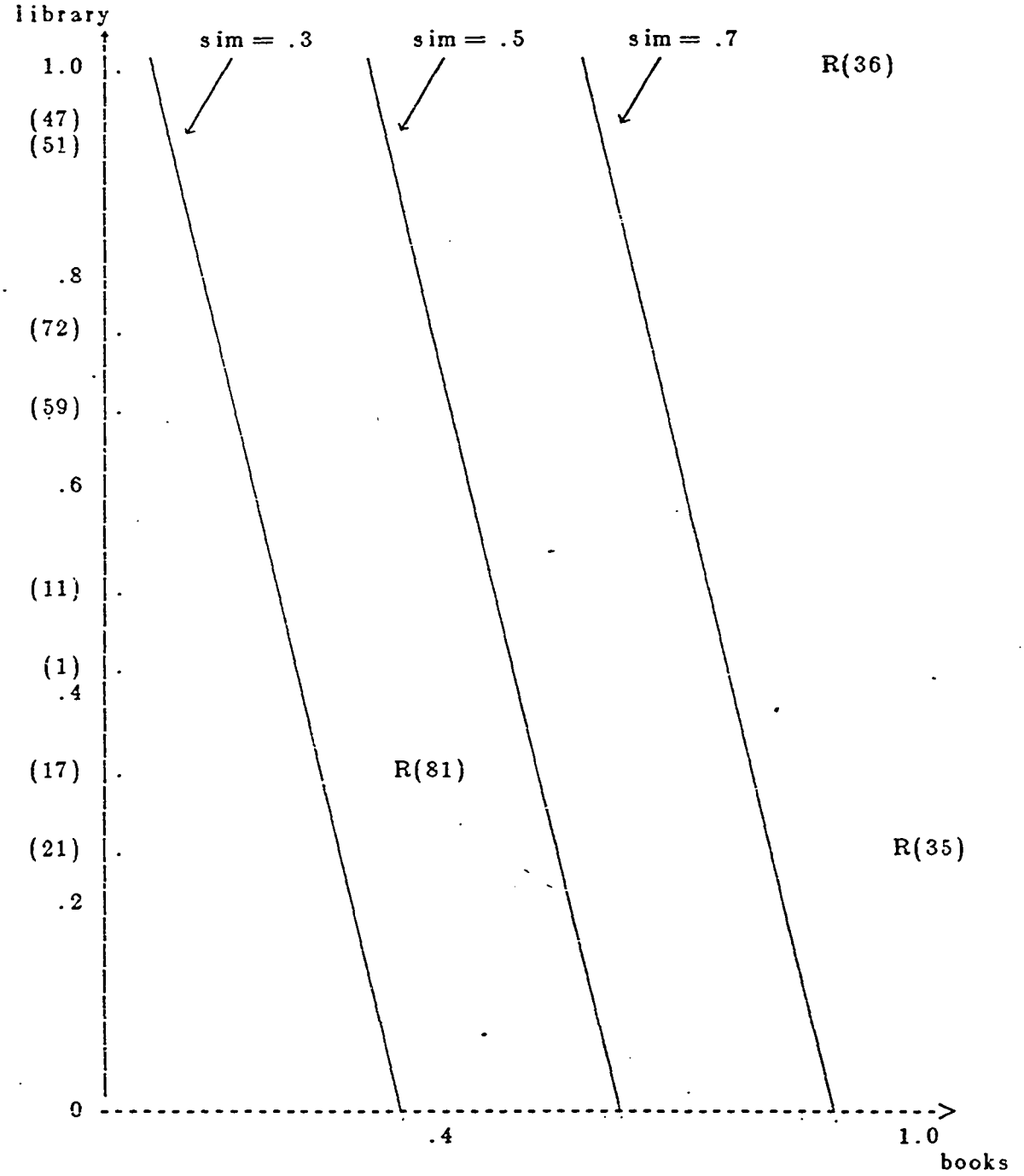
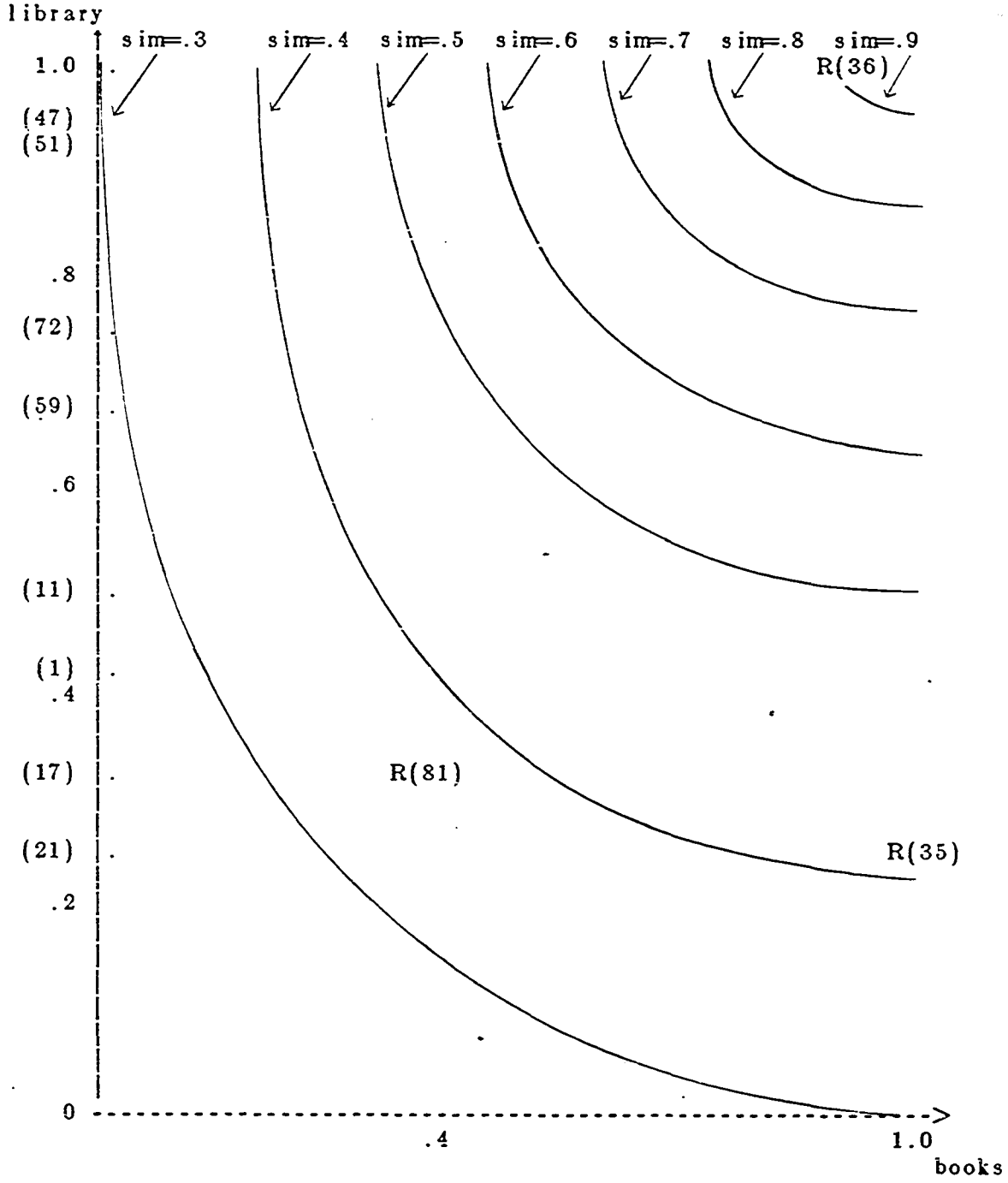


Figure 2.4: Contours for Query ''books *AND*^{2} library''



In normal situations, with many relevant documents and many non-relevant documents that each have some terms in common with the query, it is not possible to retrospectively specify simple p-norm queries that give an optimal ranking. In that sense Figures 2.3 and 2.4 are somewhat misleading. Nevertheless since p-norm expressions can be arbitrarily nested and since they allow variation of p-values and relative weights, predictive queries that give nearly optimal rankings are often possible. For more insight into the behavior of two term queries the reader is encouraged to carefully study the contours shown and explained in Appendix A.

2.2.2. Comments on P-Norm Contours

The following general observations apply to p-norm contours and their ranking of documents:

(1) When $p = 1$, one has a strict linear model where the contours are straight lines. The slopes of all lines are the same, equal to $-a/b$. By varying the relative weights, the overall similarity of a set of terms can be expressed as any of the possible linear combinations of those terms, as in the vector model.

(2) Small p-values give gentle curves; thus the Boolean operators are interpreted less strictly for low p-values. Once $p > 1.5$, there is a fair amount of curvature. At $p = 3$ the curves without relative weights are beginning to sharpen and by $p = 5$ they are rather angular, except near the 45° line. When $p = 10$ the fuzzy set definition of "L" shaped curves is fairly well approximated.

(3) The AND operator "favors" documents near the 45° line. If a document moved along a perpendicular toward the 45° line, its similarity would increase. For OR queries the reverse is true – the edges are favored.

(4) Relative weights interact strongly with p-values. This phenomenon of space distortion is quite pronounced when relative weights and p-values are not close to one. A 2:1 ratio of relative weights, coupled with as small a p-value as 2, causes the more highly weighted term to almost completely dominate the overall similarity computation. If possible relative weights close to unity should be utilized and low p-values should always be employed when differing relative weights are utilized.

(5) The effect of placement of documents in the space relates to how strictly AND and OR are interpreted (i.e., how close p is to ∞). Curvature near the 45° line is relatively gentle. As mentioned, small p-values generally give gentle curvature. For OR this is especially true near the (1,1) point while for AND it is true near the (0,0) point. Viewed another way, the ranking behavior is more sensitive to variations in p-values for an OR query when documents are closer to (0,0) and for an AND query when documents are closer to (1,1). Thus, a document weighting method that causes most documents to be close to the origin effectively causes OR operators to be interpreted more strictly than AND operators (i.e., the similarity contours for OR have more curvature than those for AND, for the same p-value, when documents are close to the origin). Stated another way, it is recommended that membership functions be utilized which spread documents appropriately (e.g., evenly) throughout the entire space.

Many other comments could be made, but would probably be confusing without the background of more experience with this new formalism. Further characteristics of the p-norm equations will be discussed where they relate to the experimental results of subsequent chapters.

2.3. Ranking Behavior of the P-Norm Similarity Function

2.3.1. Introduction

Though the contours of the previous section provide some insight as to the behavior of p-norm formula for two-term queries, they have many limitations for more general situations. Three or four dimensional plots might be of interest but would be difficult to properly interpret. Even more important than equal similarity plots, however, is having some means to consider bounds on and interactions among the various similarity functions. Analytical results are therefore given in the rest of Section 2.3.

2.3.2. Binary Query Weights in Two Term Queries

The case of two-term queries with binary query weights is easily understood. The basic result is

$$\begin{aligned} X\ AND^{\infty}\ Y &\leq X\ AND^{p}\ Y \\ &\leq X\ AND^{1}\ Y \\ &= X\ OR^{1}\ Y \\ &\leq X\ OR^{p}\ Y \quad \leq \quad X\ OR^{\infty}\ Y. \end{aligned} \tag{2-11}$$

where $1 < p < \infty,\ 0 \leq X, Y \leq 1$

The proof, using algebraic manipulations, is given in Appendix B, Section 1. When the membership functions for X and Y have the same value (i.e., $d_x = d_y$), equality holds throughout (2-11) and $X\ AND^{\infty}\ Y = X\ OR^{\infty}\ Y$. In all other cases, however, the difference between the X and Y membership functions causes inequalities to hold in (2-11). As p increases for the AND operator similarity decreases while as p increases for the OR operator similarity increases. In effect, higher p-values for AND clauses give greater

emphasis to the term with the lowest membership function while for OR clauses the term with the highest membership function becomes more important as p gets larger.

Figure 2.5 illustrates the ranking behavior predicted by (2-11). Part (a) gives a few sample points and p-values which are then shown in the diagrams of parts (b) and (d). Parts (c) and (e) demonstrate how the expected inequalities are satisfied.

2.3.3. Binary Document Weights

A second case of interest is when document terms are assigned binary weights. Actually it is sufficient to utilize any constant value, as long as all the weights on documents terms are the same, e.g., documents $\vec{D}_1$, $\vec{D}_2$, and $\vec{D}_3$ as follows

$$\vec{D}_1 = (.2, .2, \ldots, .2)$$

$$\vec{D}_2 = (.5, .5, \ldots, .5)$$

$$\vec{D}_3 = (1, 1, \ldots, 1).$$

Indeed, all that is really needed is that the same document weight be assigned to all terms present in the query.

The crucial fact is that for a given p-norm query Q and any properly weighted document $\vec{D}$, $SIM(Q, \vec{D})$ is constant regardless of the choice of p-values. Let $Q_{OP(p)}$ designate a query with single Boolean connector OP and p-value p. Then

$$\begin{aligned} d &= SIM(Q_{AND(p1)}, \vec{D}) = SIM(Q_{AND(p2)}, \vec{D}) \\ &= SIM(Q_{OR(p3)}, \vec{D}) = SIM(Q_{OR(p4)}, \vec{D}) \end{aligned} \tag{2-12}$$

where $1 \leq p1, p2, p3, p4 \leq \infty$

and $\vec{D} = (d, d, \ldots, d)$ for $0 \leq d \leq 1$.

Figure 2.5: Two Dimensional Example of Ranking Behavior

a) Example Points and P-Values

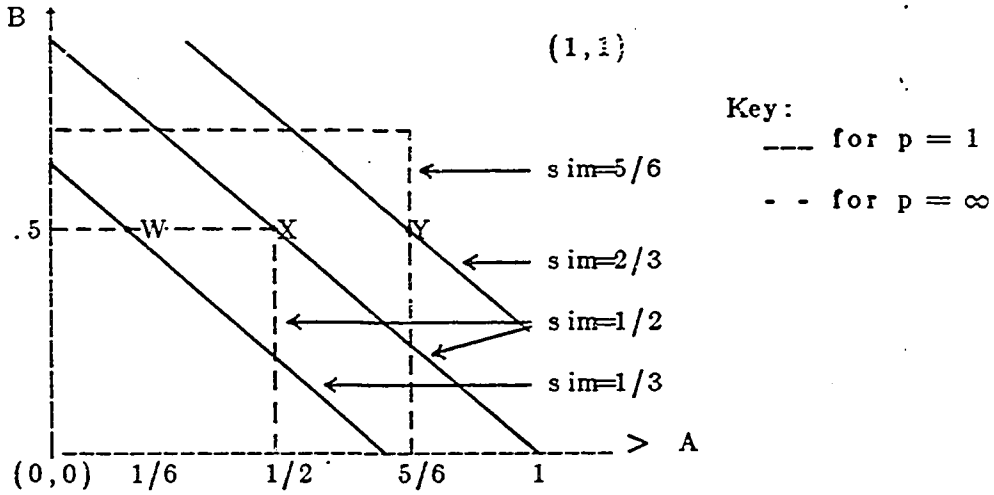
Choose $\vec{W} = (1/6, 1/2)$

$\vec{X} = (1/2, 1/2)$

$\vec{Y} = (5/6, 1/2)$

Select $p = p1 = 1$ or $p = p2 = \infty$ to use in $Q_{OR(p)}$

b) Iso-Similarity Contours for $Q_{OR(p)} = \text{A } OR^p \text{ B}$



c) Inequalities for $Q_{OR(p)}$

Regardless of p-value variations, at any point like X where $d0 = 1/2$,

$$SIM(Q_{OR(p)}, \vec{X}) = 1/2.$$

Comparing SIM($Q_{OR(p)}$, $\vec{D}$),

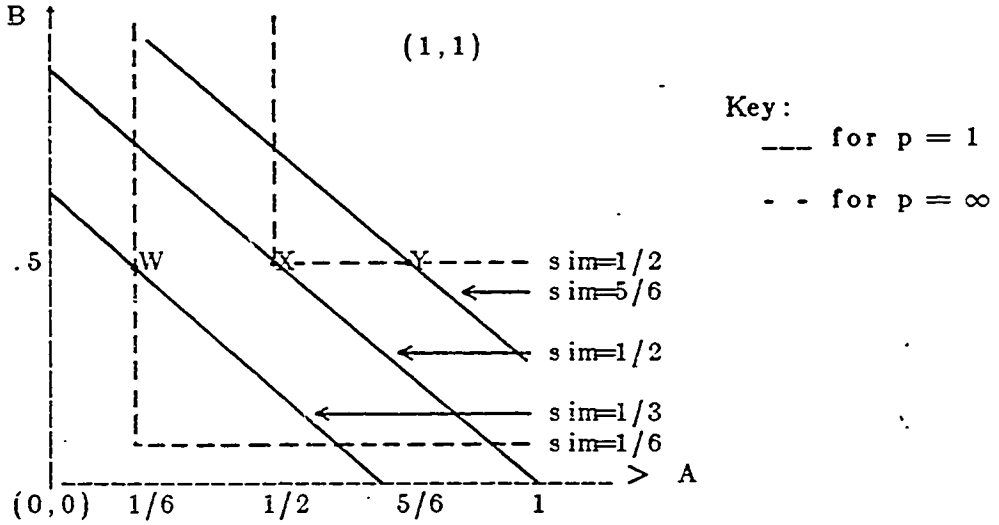
for p1 and p2 as described above, gives

$$1/3 = SIM(Q_{OR(p1)}, \vec{W}) < SIM(Q_{OR(p2)}, \vec{W}) = 1/2$$

$$2/3 = SIM(Q_{OR(p1)}, \vec{Y}) < SIM(Q_{OR(p2)}, \vec{Y}) = 5/6$$

Figure 2.5 cont'd: Two Dimensional Example of Ranking Behavior

d) Iso-Similarity Contours for $Q_{AND(p)}$ = A AND^p B



e) Inequalities for $Q_{AND(p)}$

When $d0 = 1/2$, i.e., at point X,

$$1/2 = SIM(Q_{AND(p)}, \vec{X})$$

regardless of p-value.

Comparing SIM($Q_{AND(p)}$, $\vec{D}$),

for p1 and p2 as described above, gives:

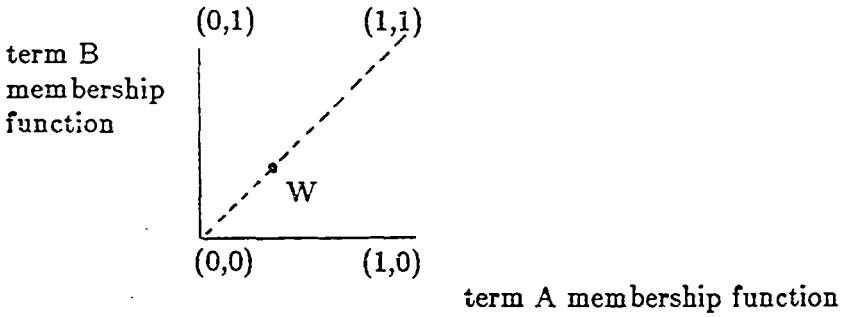
$$1/3 = SIM(Q_{AND(p1)}, \vec{W}) > SIM(Q_{AND(p2)}, \vec{W}) = 1/6$$

$$2/3 = SIM(Q_{AND(p1)}, \vec{Y}) > SIM(Q_{AND(p2)}, \vec{Y}) = 1/2.$$

A proof is given in Appendix B Section 2. Equation (2-12) corresponds to the observation that each document along the 45° line from (0, . . . , 0) to (1, . . . , 1) is assigned a similarity to a given query that is independent of p.

Thus, in the two dimensional case of Figure 2.6 below, note that document W would be given the same similarity value, for a given query, regardless of what p-value is used.

Figure 2.6: Points Along 45° Line



2.3.4. General Case

The third query form to consider is the totally general one where no constraints are placed upon document or query weights. Unfortunately, there is no simple variation of similarity with p-value. As p increases for a given query, the similarity of certain documents may decrease while for other documents the similarity may stay the same or even increase. Indeed, in a two-term space, points in that space can be easily identified that exhibit such seemingly strange behavior. For example, consider documents

$\vec{D}_1 = (.2, .5)$

$\vec{D}_2 = (.5, .5)$ and

$\vec{D}_3 = (.814, .493)$

where, for

$$Q'_{AND(p)} = <d_A, .5> AND^p <d_B, 1.0>$$

similarities follow the equalities

$$0.4 \approx SIM\ (Q'_{AND(1)}, \vec{D}_1) < SIM\ (Q'_{AND(10)}, \vec{D}_1) \approx 0.6$$

$$0.5 = SIM\ (Q'_{AND(1)}, \vec{D}_2) = SIM\ (Q'_{AND(10)}, \vec{D}_2) = 0.5$$

$$0.6 \approx SIM\ (Q'_{AND(1)}, \vec{D}_3) > SIM\ (Q'_{AND(10)}, \vec{D}_3) \approx 0.4.$$

Figure 2.7 is the familiar unit square with these points shown together with the equal similarity contours for AND^1 and AND^{10}. Appendix B Section 3 gives computations of similarities for the above equations.

2.3.5. Key Resuit for Binary Query Weights

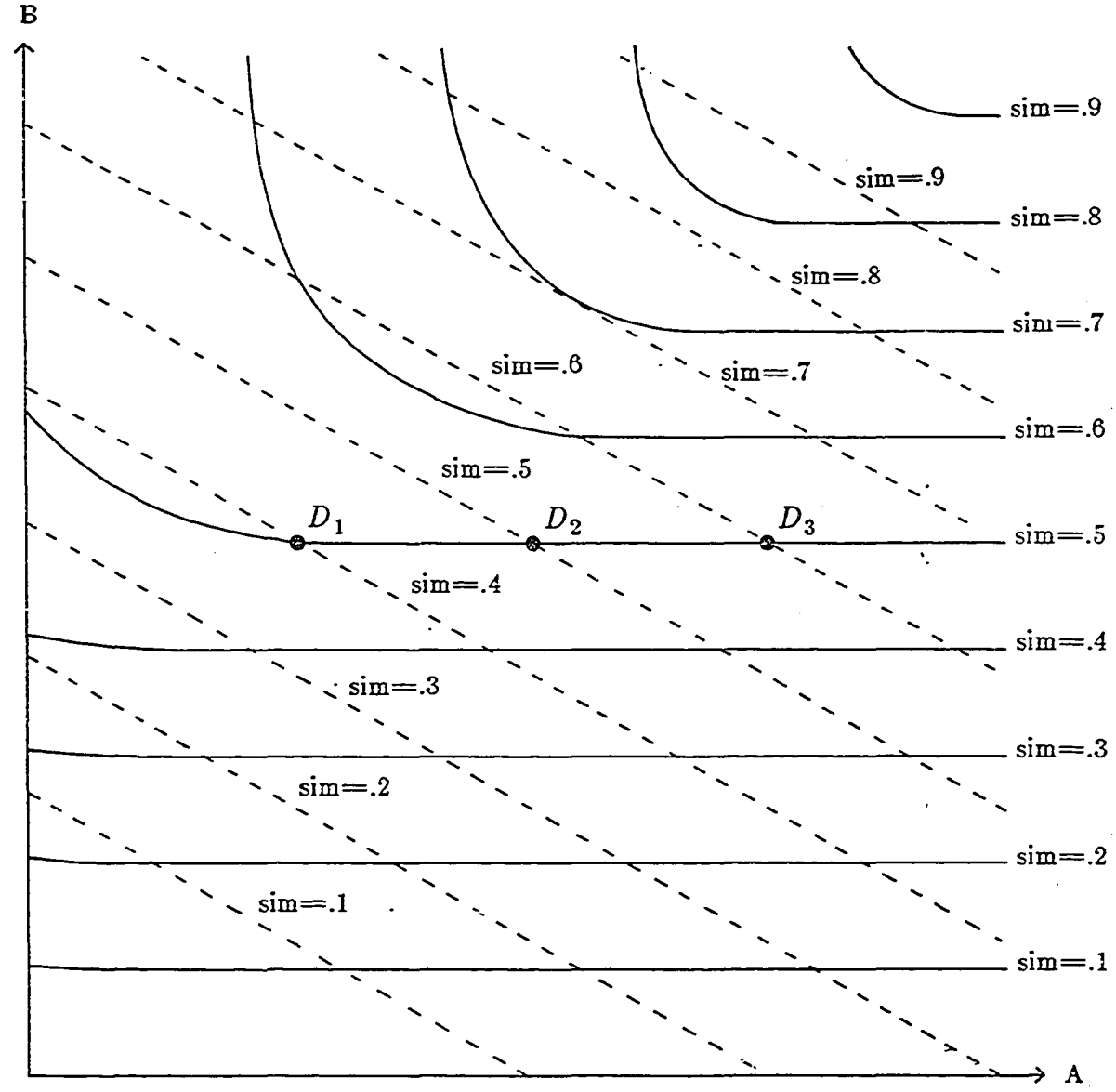
Since the ranking behavior in the general case is not amenable to simple analysis, as was shown by counterexample in the previous section, it is appropriate to consider the simpler and very common case of having binary query weights. Here the two term formula (2-11) of Section 2.3.2 is generalized to the n-term query case.

The key result is that, for binary query weights and queries with n-terms,

where $1 \leq p1, p2 \leq \infty$

that

Figure 2.7: Differing Ranking Effects Due to Varying P



Key:

(1) dashed lines - iso-similarity points for query

$Q'_1 = AND^1 (<d_A, .5>, <d_B, 1.0>)$

(2) solid lines - iso-similarity points for query

$Q'_{10} = AND^{10} (<d_A, .5>, <d_B, 1.0>)$

$$
\begin{aligned}
SIM\ (Q_{AND(\infty)}, \vec{D}) &\leq SIM\ (Q_{AND(p2)}, \vec{D}) \\
&\leq SIM\ (Q_{AND(p1)}, \vec{D}) \\
&\leq SIM\ (Q_{AND(1)}, \vec{D}) \\
&= SIM\ (Q_{OR(1)}, \vec{D}) \\
&\leq SIM\ (Q_{OR(p1)}, \vec{D}) \\
&\leq SIM\ (Q_{OR(p2)}, \vec{D}) \\
&\leq SIM\ (Q_{OR(\infty)}, \vec{D}).
\end{aligned}
\tag{2-13}
$$

Deriving (2-13) is somewhat difficult, so the complete proof is given in Appendix B. Both a long geometric version (Section 5) and a short version using calculus (Section 4) are presented there. Actually, the complete result is shown only for OR – the AND case can be demonstrated analogously, and parts of it are indeed given in Appendix B.

2.3.6. Multi-Level Vectors

It has been stated that when $p = 1$, the p-norm model reduces to the vector model. The meaning is clear for expressions with a single Boolean connector. For more complex nested constructions it is not as obvious how the original structure affects term weights.

First, consider the following query:

$$Q = (A\ AND^1\ B\ AND^1\ C)\ OR^1\ (D\ AND^1\ E)\ .$$

For document

$$\vec{D} = (\ <A, d_A>, <B, d_B>, <C, d_C>, <D, d_D>, <E, d_E>\ ),$$

the similarity equation, after substitution of definitions recursively, is

$$SIM\,(Q,\vec{D}) = \frac{\left[\frac{d_A + d_B + d_C}{3}\right] + \left[\frac{d_D + d_E}{2}\right]}{2}$$

$$= \frac{d_A}{6} + \frac{d_B}{6} + \frac{d_C}{6} + \frac{d_D}{4} + \frac{d_E}{4}$$

Hence, the clausal structure is flattened to a vector, but terms in longer clauses or very far down in the expression tree may have very low weights in the final result.

When relative weights are employed, the result is even more complex. Thus, for

$$Q = (\ \langle A,a\rangle\ AND^1\ \langle B,b\rangle\ AND^1\ \langle C,c\rangle\ )\ OR^1\ (\ \langle D,d\rangle\ AND^1\ \langle E,e\rangle\ )$$

the similarity with $\vec{D}$ is

$$\frac{\frac{a\cdot d_A + b\cdot d_B + c\cdot d_C}{a + b + c} + \frac{d\cdot d_D + e\cdot d_E}{d + e}}{2}$$

which is linear for fixed relative weights, but still reminiscent of the original structure, due to the weighting and normalization.

When $p \neq 1$, the similarity is no longer linear. Though the formulas provide a reliable means of determining similarities, they provide little intuition regarding the ranking behavior. That is in part why contours are given in Appendix A and why the discussion in Section 2.2 was included.

It is hoped that, with the aid of the graphical and analytical results of this chapter, the reader will now feel somewhat more confident about the meaning of p-norm expressions. In any case, the experimental tests described in later chapters should be easily understood even if the details of this chapter have been only casually considered.

CHAPTER 3

P-NORM IMPROVEMENTS OF EXISTING QUERIES

For a number of test collections, experiments are described concerning whether using the p-norm formalism to interpret Boolean queries will result in improvements in search effectiveness. Numerous tables, regression results, and summary charts provide empirical evidence as to the value of p-norm queries in extending standard Boolean operations.

3.1. Preliminaries

In order to determine whether the p-norm formulation merited development for possible practical use, experiments were conducted using the small ADI collection of 82 documents in the field of documentation. Based on recall-precision[1] tables produced for each of 35 manually constructed Boolean queries, the effectiveness of various query formulation methods were compared. Some of the most important conclusions reached are:

(1) Rather than having to code or scale the p-values, it was workable to use the actual numbers – e.g., $p = 1.5$ or 2.

---

[1]Define

$$\text{recall} = \frac{\text{number relevant retrieved}}{\text{number relevant}} \qquad \text{precision} = \frac{\text{number relevant retrieved}}{\text{number retrieved}}.$$

Recall-precision tables show the average over all queries of precision for each recall level, selected at 10% intervals. Thus, uniformly higher values indicate more effective queries. Though some may argue on technical grounds against use of such evaluation for Boolean queries, the requirement of comparison of vector and p-norm methods made this choice the most appropriate.

(2) Using document term weights based on normalized term frequency values, e.g.,

d_{ij} = weight of term no. j in document no. i

$$= \frac{n_{ij}}{\max_{t_k \in \vec{D}_i} n_{ik}}$$

where n_{ik} = number of occurrences of term t_k in document $\vec{D}_i$

queries performed more effectively than if document weights were binary.

(3) The performance for good p-norm queries was better than that of cosine runs made.

3.1.1. Document Weighting Method

To determine the most effective document weighting scheme, a set of 13 queries, each with two terms, were selected so as to cover a wide variety of combinations of frequency characteristics – e.g., to have pairs with frequency behavior (low, low), (low, medium), etc. For example, query 1, "government agencies", falls into category (low, low) while query 4, "library books", falls into category (high, medium).

Defining:

n_{ij} = number of occurrences of term t_j in document $\vec{D}_i$

n_{imax} = number of occurrences of the most frequent term in $\vec{D}_i$

f_j = number of postings of term t_j in the collection

f_{jmax} = number of postings of the term which occurs in the maximum (largest) number of documents of the collection

$idf_j = \log (f_{jmax}/f_j)$

$idf_{jmax} = idf_j$ for the term t_j which has fewest postings, and hence maximum (largest) idf value

the various schemes of document weighting can be described as

(1) normalized term frequency $= n_{ij}/n_{imax}$

(2) normalized inverse frequency $= idf_j/idf_{jmax}$

(3) average of schemes 1 and 2

(4) product of schemes 1 and 2.

Using the two term queries and the relevance judgments that were prepared especially for them, charts were produced with equal similarity contours, similar to those of Section 2.2. After analysis of the placement of documents in the unit square and the effects of that placement on ranking behavior, it was observed that scheme 4, essentially a normalized version of the weighting scheme utilized in vector runs (i.e., "tf*idf" $\equiv n_{ij} * idf_j$) was most appropriate. However, due to the shape of the normalized frequency distributions (schemes 1, 2 above), most documents were then placed near the origin, resulting in asymmetrical handling of the OR and AND operators (see discussion in Section 2.2, specifically point 5 of Section 2.2.2). The problem was to devise a version of (4) that would spread documents more widely over the space.

Standard statistical transformations did not possess the desired properties. Furthermore, it was observed that a straightforward utilization of (1) as one factor in formula (4) caused inappropriate down weighting of terms that occurred only once in a document. Such terms should have a membership value of, say, at least 0.5 out of 1.0, as regards the term frequency component. The final selected document weighting scheme, used in subsequent experiments as well, was designated as "tf*idf" (since it corresponds to the similarly named weighting scheme used in cosine runs). The definition is, for term t_j in $\bar{D}_i$,

$$\text{"tf*idf" wt} = \left[\frac{idf_j}{idf_{jmax}}\right] * \left[0.5 + 0.5 * \frac{n_{ij}}{n_{imax}}\right] \tag{3-1}$$

3.1.2. Query Weights

To try out the proposed weighting scheme of equation (3-1) and to see what the interaction is with relative weights on query terms, 19 new two clause queries were constructed. By way of example, consider the first query

$$\begin{aligned} &AND\ (\ OR\ (\text{catalogue, catalog}), \\ &\qquad OR\ (\text{mechanization, automation, computerization})\ ) \end{aligned} \tag{3-2}$$

where entries in each clause, especially the first, are all exact or near synonyms. It is clearly the query author's intention to treat occurrence of either "catalogue" or "catalog" as equivalent and so it is appropriate to include them in an OR clause. Elements of the second clause are also nearly synonymous with each other in the context of collection concepts, but each occurs in more documents than do the terms of the first clause. Since such occurrence statistics are likely to affect retrieval, it is sensible to examine how query weights might reflect such semantic and frequency considerations.

The query shown in (3-2) above is a standard Boolean query in which no p-values or weights are explicitly shown. However, the p-norm notation allows p-values and weights to be assigned in many places. To illustrate this, the real meaning of (3-2), where all default values are substituted in, is shown as equation (3-3).

$$\begin{aligned} &AND^{\infty}(\ <OR^{\infty}(<\text{catalogue}, 1>, <\text{catalog}, 1>), 1>, \\ &\qquad <OR^{\infty}(<\text{mechanization}, 1>, <\text{automation}, 1>, \\ &\qquad\qquad <\text{computerization}, 1>), 1>\ ) \end{aligned} \tag{3-3}$$

The most general expression possible with this query structure is:

$$AND^{p_0}(\ <OR^{p_{i1}}(\ <\text{catalogue}, w_{11}>, <\text{catalog}, w_{12}>\ ), w_{i1}>, \quad (3\text{-}4)$$
$$<OR^{p_{i2}}(\ <\text{mechanization}, w_{21}>, <\text{automation}, w_{22}>,$$
$$<\text{computerization}, w_{23}>\ ), w_{i2}>\ )$$

where p_0 is the outer p-value, p_{i1} and p_{i2} are inner p-values, w_{i1} is the first inner clause's relative weight, w_{21} is the term weight for the first term of the second clause, etc. Clearly, for complex queries, having so many parameters could become rather confusing, so systematic methods of determining parameter values become a necessity. A number of experiments were conducted where the inner and outer p-values were all varied independently, but similar results were obtained to those found when $p_0 = p_{i1} = p_{i2}$.

Thus, the following simplifying rules were followed in p-value and query weight assignment.

(1) All p-values in a query were assigned the same setting.

(2) Term weights such as w_{11}, w_{12}, and w_{23} were either all binary or all idf. That is to say, either 1.0 was uniformly assigned to all terms, or else the collection idf value for the term was used. Formally, the concept weight pair for concept c_i was either

$$<c_i, 1> \quad \text{or} \quad <c_i, idf_i>. \quad (3\text{-}5)$$

Since binary weights are commonplace, and since idf is a reasonable assignment for query weighting when other information is lacking, these forms seemed appropriate for initial testing.

(3) Clause weights such as w_{i1} and w_{i2} were either assigned binary or average idf weights. Thus, inner clause k with concepts $c_{k1}, \ldots, c_{kn}$ would have weight w_k computed as

either

$$w_{ik} = 1.0 \tag{3-6}$$

or

$$w_{ik} = \frac{1}{n}\left[idf_{k1} + idf_{k2} + \cdots + idf_{kn}\right].$$

3.1.3. Weighting Cases of Interest

Various combinations of p-value assignment and query weighting systems were candidates for experimentation. For the 2 clause query tests, the most important cases tested are described and illustrated below.

(1) Base case – Boolean statement

As shown in (3-2) or in the fully specified form (3-3), the typical Boolean query is easily described by using $p = \infty$ and binary weights throughout.

(2) Vector case – flat query

The other base case to compare with is the "flat" vector form, where no clauses exist, the outer operator has $p = 1$, and idf weights are used on query terms.

Table 3.1: Vector Formulation of Query 1

(<catalogue,6.358>, <catalog,5.358>, <mechanization,4.036>,
<automation,2.657>, <computerization,1.603>)

---

(3) Fully Weighted P-Norm Query

Here, idf weights are used on query terms and average idf values are used for clauses.

A single p-value is used throughout and can be varied from the vector case of $p = 1$ to the Boolean case of $p = \infty$.

Table 3.2: Fully Weighted P-Norm Formulation

AND^{p_0} ($< OR^{p_{11}}$(<catalogue, 6.358>, <catalog, 5.358>), 5.898 >,

$< OR^{p_{12}}$(<mechanization, 4.036>, <automation, 2.657>,

<computerization, 1.603>), 2.765>)

These three cases were those found to be effective for the ADI two clause queries and so were selected for testing with other larger collections.

3.2. Medlars Experiments

3.2.1. Collection

The ADI collection contains only 82 documents, so it is conceivable that results might hold for that size collection and not for larger ones. Hence the Medlars collection of 1033 documents and 30 queries was employed to further test the utility and behavior of p-norm queries. A description of the document and query collections employed is given in Appendix C. However, a few comments about the quality of the queries are in order here.

The queries were originally constructed by Medlars searchers, using text words and MESH[2] thesaurus terms. Since the document file lacked thesaurus terms, each occurrence of a MESH term in some query was replaced by the disjunction of the entries given under

---

[2]MESH "Medical Subject Headings" in [National Library of Medicine 1968].

that thesaurus class. Since this expansion replaced original terms with a group of terms having different specificity[3] and frequency characteristics, it is not surprising that the Boolean queries performed less well than those of the earlier published Medlars-SMART comparison [Salton 1972b]. Nevertheless, they were very useful for relative comparisons associated with experiments described in this and subsequent chapters.

3.2.2. Recall-Precision Charts

Throughout the rest of this thesis, recall-precision charts or summaries thereof will be given as the basic data supporting observations and conclusions stated. This section will illustrate the procedure and explain the rationale.

For each comparison chart such as those of Tables 3.3 and 3.4, a number of cases are considered. The first serves as the base case, so changes and percentage improvements can be given against a common standard. Often the usual Boolean situation (i.e., binary document weights, binary query weights, $p = \infty$ everywhere) serves as the basis for comparison and is assigned the first column for output.

The standard form of the chart includes a heading describing the collection and test being made followed by a list of descriptions of cases that follow. The body of the chart shows for each level of recall, the average precision value for each case. Thus, at each of 21 levels, the average precision over the 30 Medlars queries is presented. By comparing precision values it is possible to see which cases are uniformly better than others and which are better only in selected parts of the recall-precision curves.

---

[3]Indexing specificity measures the value of a term or other type of concept assigned; it refers to the generic level used to characterize content. A specific vocabulary uses narrowly define terms so that most of the non-relevant items will not be retrieved by typical queries [Salton & McGill 1983 page 55].

Table 3.3: Medlars 1033 Docs, 30 Queries Results.
Fix dwt=qwt=cwt=binary - Vary p.

Results for the following cases are shown below:
1) Base case: p-norm with dwt=bin, qwt=bin, cwt=bin, p=∞
2) Test case: p-norm with dwt=bin, qwt=bin, cwt=bin, p=1
3) Test case: p-norm with dwt=bin, qwt=bin, cwt=bin, p=1.5
4) Test case: p-norm with dwt=bin, qwt=bin, cwt=bin, p=2
5) Test case: p-norm with dwt=bin, qwt=bin, cwt=bin, p=3
6) Test case: p-norm with dwt=bin, qwt=bin, cwt=bin, p=5
7) Test case: p-norm with dwt=bin, qwt=bin, cwt=bin, p=9

recall	precision for cases:						
level	1	2	3	4	5	6	7
0.00	0.7327	0.8214	0.8438	0.8438	0.8660	0.8508	0.8686
0.05	0.6878	0.7924	0.8081	0.8081	0.8303	0.8152	0.8329
0.10	0.5606	0.7383	0.7619	0.7691	0.7883	0.7862	0.7782
0.15	0.5082	0.7005	0.7217	0.7291	0.7586	0.7545	0.7514
0.20	0.4364	0.6635	0.6852	0.6895	0.7109	0.6997	0.6982
0.25	0.3518	0.6200	0.6205	0.6259	0.6198	0.6213	0.6184
0.30	0.2917	0.5956	0.6125	0.6133	0.6038	0.6056	0.6090
0.35	0.2699	0.5519	0.5791	0.5840	0.5842	0.5824	0.5883
0.40	0.2387	0.5263	0.5520	0.5583	0.5441	0.5371	0.5442
0.45	0.1898	0.4939	0.5001	0.5147	0.5065	0.4937	0.4922
0.50	0.1871	0.4680	0.4726	0.4757	0.4638	0.4621	0.4574
0.55	0.1603	0.4399	0.4453	0.4425	0.4339	0.4366	0.4294
0.60	0.1562	0.4294	0.4344	0.4294	0.4238	0.4241	0.4214
0.65	0.1108	0.3666	0.3721	0.3750	0.3723	0.3742	0.3555
0.70	0.1073	0.3563	0.3564	0.3567	0.3563	0.3568	0.3367
0.75	0.0805	0.3249	0.3289	0.3284	0.3323	0.3326	0.3038
0.80	0.0774	0.2853	0.2899	0.2914	0.2967	0.2933	0.2655
0.85	0.0443	0.2147	0.2182	0.2069	0.2001	0.1980	0.1827
0.90	0.0385	0.1696	0.1695	0.1531	0.1470	0.1436	0.1413
0.95	0.0339	0.1222	0.1186	0.1094	0.1052	0.1072	0.1082
1.00	0.0324	0.1116	0.0996	0.0887	0.0862	0.0857	0.0859

average prec and % change vs. base case, for levels: .25, .50, .75

prec=	0.2065	0.4710	0.4740	0.4767	0.4720	0.4720	0.4599
% prec change=	128.1	129.6	130.9	128.6	128.6	122.7	

Table 3.4: Medlars 1033 Docs, 30 Queries Results.
Fix dwt=tf*idf, qwt=cwt=binary – Vary p.

Results for the following cases are shown below:
1) Base case: p-norm with dwt=bin, qwt=bin, cwt=bin, p=∞
2) Test case: p-norm with dwt=tf*idf, qwt=bin, cwt=bin, p=1
3) Test case: p-norm with dwt=tf*idf, qwt=bin, cwt=bin, p=1.5
4) Test case: p-norm with dwt=tf*idf, qwt=bin, cwt=bin, p=2
5) Test case: p-norm with dwt=tf*idf, qwt=bin, cwt=bin, p=3
6) Test case: p-norm with dwt=tf*idf, qwt=bin, cwt=bin, p=5
7) Test case: p-norm with dwt=tf*idf, qwt=bin, cwt=bin, p=9
8) Test case: p-norm with dwt=tf*idf, qwt=bin, cwt=bin, p=∞

recall	precision for cases:							
level	1	2	3	4	5	6	7	8
0.00	0.7327	0.9139	0.9138	0.9103	0.8644	0.8803	0.8800	0.7645
0.05	0.6878	0.8845	0.8722	0.8835	0.8501	0.8384	0.8352	0.6958
0.10	0.5606	0.8136	0.8256	0.8126	0.8158	0.8110	0.8150	0.6200
0.15	0.5082	0.7750	0.8032	0.7822	0.7825	0.7910	0.7876	0.5585
0.20	0.4364	0.7467	0.7927	0.7724	0.7603	0.7606	0.7654	0.4762
0.25	0.3518	0.7183	0.7557	0.7436	0.7293	0.7332	0.7357	0.3940
0.30	0.2917	0.6799	0.7062	0.7011	0.7027	0.6948	0.6984	0.3175
0.35	0.2699	0.6484	0.6659	0.6615	0.6670	0.6523	0.6638	0.2961
0.40	0.2387	0.6271	0.6236	0.6291	0.6376	0.6147	0.6143	0.2616
0.45	0.1898	0.5930	0.5940	0.5994	0.6029	0.5839	0.5835	0.2148
0.50	0.1871	0.5448	0.5520	0.5547	0.5555	0.5543	0.5434	0.2128
0.55	0.1603	0.5141	0.5053	0.5138	0.5145	0.5106	0.5044	0.1906
0.60	0.1562	0.4945	0.4818	0.4789	0.4844	0.4880	0.4806	0.1860
0.65	0.1108	0.4414	0.4306	0.4261	0.4154	0.4189	0.4112	0.1376
0.70	0.1073	0.4198	0.4130	0.4037	0.3942	0.3837	0.3941	0.1318
0.75	0.0805	0.3884	0.3857	0.3736	0.3567	0.3450	0.3460	0.1039
0.80	0.0774	0.3592	0.3573	0.3499	0.3301	0.3193	0.3128	0.0939
0.85	0.0443	0.3032	0.2967	0.2912	0.2758	0.2579	0.2538	0.0570
0.90	0.0385	0.2247	0.2196	0.2147	0.1997	0.1855	0.1807	0.0489
0.95	0.0339	0.1492	0.1437	0.1407	0.1306	0.1266	0.1201	0.0441
1.00	0.0324	0.1288	0.1219	0.1114	0.1024	0.0975	0.0954	0.0388

average prec and % change vs. base case, for levels: .25, .50, .75

prec=	0.2065	0.5505	0.5645	0.5573	0.5472	0.5442	0.5417	0.2369
% prec change=	166.6	173.4	169.9	165.0	163.6	162.4	14.7	

It should be noted that due to the random ranking procedures adopted for ties and non-retrieved documents the precision values given for high recall values may not always provide fair comparisons. Also, it has been argued that precision results for very low recall levels are not very realistic since usually at least ten documents would be retrieved and it is often unimportant where in that group the relevant occur as long as they are present somewhere in the group. Based in part on these considerations, and further to enable brevity in presentation of results, a simple average precision value is also shown, covering the middle portion of the chart. Thus, the average of the precision values at recall levels .25, .50, and .75 is shown for each case. Furthermore, the percentage improvement of those values against the base case are given to show relative performance.

Substantial improvements in performance are obtainable using p-norm queries in conjunction with conventional Boolean processing and unweighted documents and queries. Table 3.3 indicates that using any p-value other than ∞ seems wise. The best results are for p-values between 1 and 5. As p-value increases from 1, performance increases and then gradually decreases but even at relatively high p-values (e.g., 9) there has not been a substantial degradation in effectiveness. Strict Boolean queries thus do appear to be overly restrictive in retrieving relevant documents so using lower p-values is advised if possible.

Table 3.4 relates to use of the new "tf*idf" normalized document weighting scheme given in equation (3-1). Query weights are all still binary. For all values shown of p other than ∞ (i.e., the strict Boolean case), performance exceeds that in the previous table; using document weighting seems to be clearly better than using binary weights. As in the previous table, the best p-values are low ones but even after that there is little change in performance as p increases.

3.2.3. Comparisons

Table 3.5 highlights key results of Tables 3.3 and 3.4 in the context of related runs. In particular, the cosine run with tf*idf weighting applied to the results of automatically indexing the original natural language queries (NL terms – case 7, the basis for the second column of percentage improvements), and a similar run on the terms taken from a flattened form of the Boolean logic queries (BL terms – case 6) provide contrast with Boolean methods. For comparison purposes, and to reduce confounding of effects, the cosine run on BL terms is more realistic; like the p-norm cases, its performance suffers from the large set of BL terms employed. Hence, to determine the effectiveness of p-norm methods applied to the given Boolean queries (i.e., cases 1-5), it seems appropriate to compare with a cosine run on the same terms (i.e., case 6).

Table 3.5: Summary of Different Medlars Trials

case no.	p value	doc. wt.	query wt.	description of trial	aver. prec.	%improv. vs. Bool.	%improv. vs. cos.
1	∞	binary	binary	standard Boolean	.207	Base	-62.2
2	2	binary	binary	p-norm	.477	130	-12.8
3	1.5	tf*idf	binary	p-norm	.565	173	3.3
4	1	tf*idf	binary	p-norm	.550	166	0.5
5	1	tf*idf	idf	p-norm	.547	164	0
6	-	tf*idf	tf*idf	cos.BL-terms	.445	115	-18.6
7	-	tf*idf	tf*idf	cos.NL-terms	.547	164	Base
8	-	binary	probab.	retrospective indep.case BL terms	.661	219	20.8
9	-	binary	probab.	retrospective indep.case NL terms	.718	247	31.3

As a final point of reference, the probabilistic results of a retrospective run, made with complete knowledge of query-document relevance, are given as an upper bound on the possible performance of linear weighting schemes. Based on the assumption that weight assignment to each term should be done independently, the scheme employed assigns optimal weights to query terms [Van Risjbergen 1980]. Thus, other methods that perform almost as well as this case can not be improved much further. Cases 8 and 9 of Table 3.5 give these limiting values for linear weights on BL and NL terms, respectively.

Based on the average precision values summarized in Table 3.5, the following comments can be made:

(1) Using binary weights and $p<\infty$, the p-norm methods (e.g., Table 3.5 case 2) do better than the cosine run on the same terms (case 6), but not as well as the cosine run on the original natural language terms (case 7). Improvements for these three cases versus the original queries are 130%, 115% and 164%, respectively.

(2) When document weights and low p-values are combined, performance improvements surpass those of all but the upper bound case. The enhanced Boolean run yields an improvement of up to 173% (i.e., case 3 for $p = 1.5$, or 166% for case 4 with $p = 1$), which is slightly more than the 164% result for the usual cosine case (case 7).

(3) The use of p slightly larger than 1 (i.e., $p = 2$ for binary weight case number 2 or $p = 1.5$ for document term weighted case 3 seems advisable. This implies that the Boolean structure does convey useful information. With p-norm methods it is possible to employ just the right amount of strictness in interpreting the Boolean connectives, and when p-values are low, having a reasonably good document membership function can further improve retrieval system performance.

3.2.4. Detailed Analysis of Weighting and P-Norm Variations

Though the cases shown in Tables 3.3 and 3.4 convey the principal results of p-norm behavior for the Medlars experiments, other combinations of weighting methods and p-values were also tested.

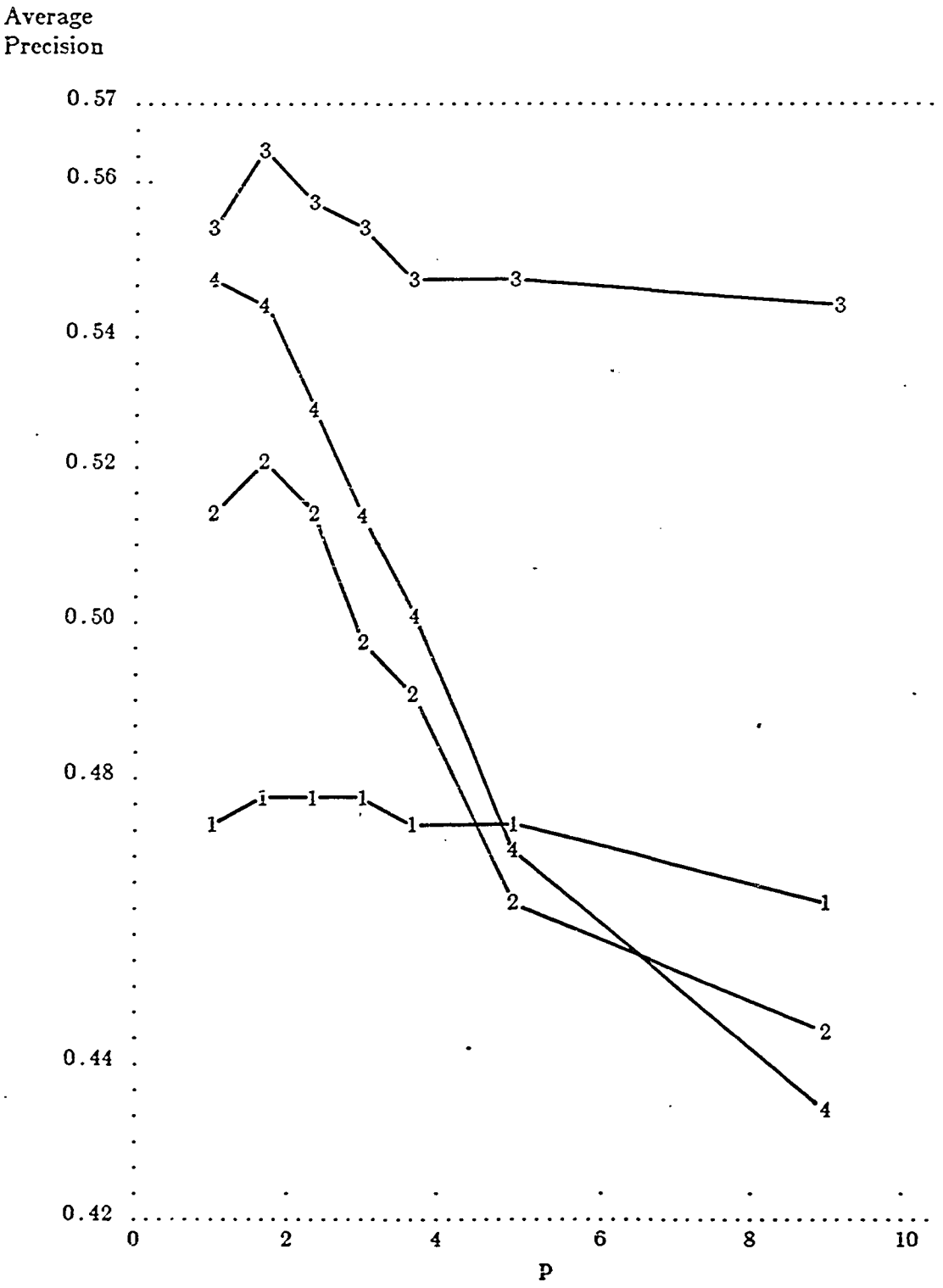
Figure 3.1 gives in graph form the above mentioned single valued measure (i.e., the average precision at levels of recall .25, .50, .75) for the various combinations of document and query weights considered, each for eight values of p. It shows the performance variation (here using values of the average precision measure) of each weighting case against p.

To further explore the interaction between weighting methods and p-value assignment, the results summarized in Figure 3.1 were used as data for regression analysis. Two categorical variables, namely document weight method (either binary or tf*idf) and query weight method (either binary or idf), and one quantitative variable (or covariate), the p-value, were the independent variables. A single measure of performance was chosen as the dependent variable – the average precision value. The four cases considered are designated as in Table 3.6.

Table 3.6: Weighting Methods for Regression

Label	Document Weighting	Query Weighting
1	binary	binary
2	binary	idf
3	tf*idf	binary
4	tf*idf	idf

Figure 3.1: Plot of Precision vs. P for 4 Weighting Cases Using Boolean Queries



A factorial design was adopted, so precision values were needed for each weighting case at a number of p-values (i.e., 1, 1.5, 2, 2.5, 3, 5, 9). Some obvious trends are immediately apparent from the data, and can be seen by inspection of Figure 3.1.

(1) The use of query term weights causes performance to fall off as p-value increases. Thus, cases 2 and 4 are different from cases 1 and 3.

(2) For the range of interest case 3 is very effective and as immune to the effects of increasing p-values as is case 1.

These trends are more precisely stated using the model sequence approach to regression analysis [Allen & Cady 1982].

Figures 3.2 through 3.5 present four such models aimed at determining the significance of each parameter and at identifying interactions among those parameters. The multiple R-square value indicates the quality of the fit for a given model, where 1.0 appears for a perfect fit. Each parameter is assigned a coefficient in the usual linear model, and the significance of that coefficient is determined by the value for the t-test shown. The "Results" part of these tables comes directly from output of the S statistical package [Becker & Chambers 1981].

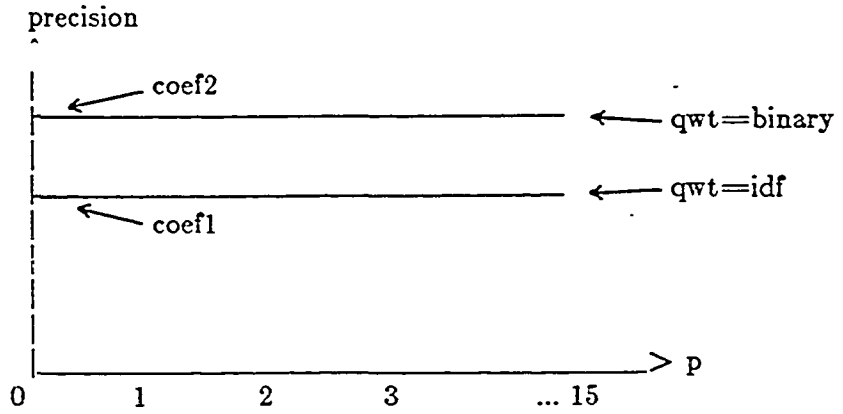
For Figure 3.2 the linear equation is

$$\text{precision} = coef_1 * x_1 + coef_2 * x_2$$

which tries to fit (i.e., predict precision) using only the criterion of whether the query terms are weighted. The high multiple R^2 value is close to 1.0 so the fit is fairly good, indicating that query weighting method is an important variable.

Figure 3.2: Regression of Precision vs. Query Wt.

Model:



Key to Results:

x1 = 1, x2 = 0 when qwt=idf
x1 = 0, x2 = 1 when qwt=binary

Results:

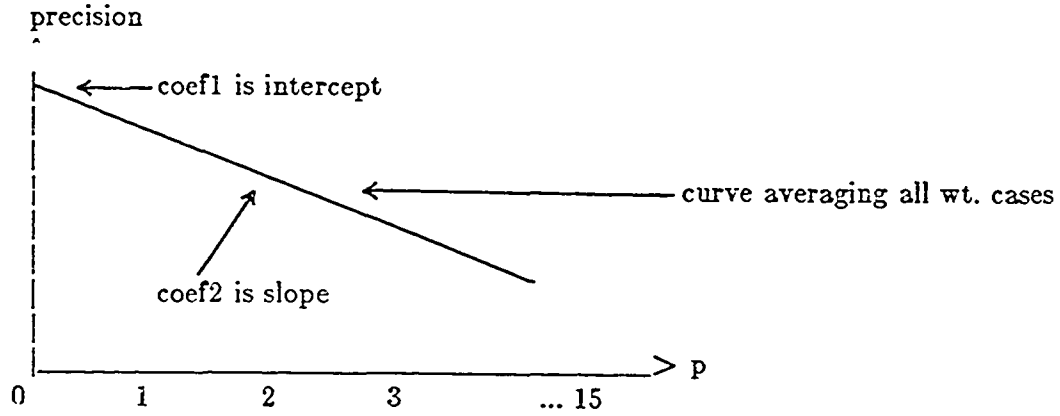
	Coef	Std Err	t Value
x1	0.4982143	0.01027741	48.47663
x2	0.5113572	0.01027741	49.75543

Residual Standard Error = 0.03845455 Multiple R-Square = 0.994641
N = 28 F Value = 2412.794 on 2, 26 df

To see whether precision decreases as p-value increases, regardless of what weighting scheme is chosen, the second model was devised, as shown in Figure 3.3. The poor fit indicates that although there is a general downward trend for precision versus p-value, it is not advisable to ignore the effects of weighting methods.

Figure 3.3: Regression of Precision vs. p-Value

Model:



Key to Results:

x2 = p - $\hat{p}$

where $\hat{p}$ = average of cases used

Results:

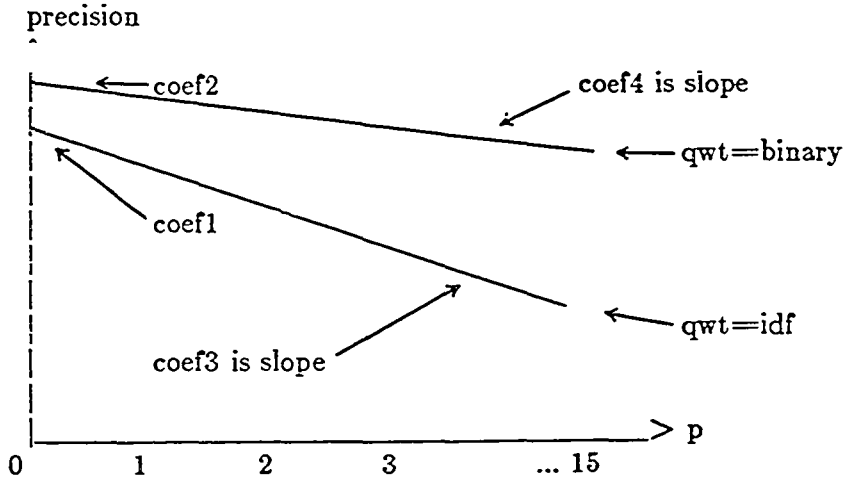
	Coef	Std Err	t Value
Intercept	0.5047857	0.00646385	78.09368
x2	-0.00706994	0.00251566	-2.81037

Residual Standard Error = 0.0342035 Multiple R-Square = 0.2329971
N = 28 F Value = 7.89818 on 1, 26 df

Figure 3.4 provides an excellent fit, based on considering the interaction between p-value and query weight method. The model is that depending on whether query terms are weighted or not, there is a different slope for each of the two situations. Thus cases 1 and 3 essentially share the same intercept and slope while cases 2 and 4 share another intercept-slope pair of values. The slope for cases 1 and 3 is not nearly as sharp as for the other cases, as can be seen from the rather low t-test value for x_4.

Figure 3.4: Regression of Precision vs. Query Wt. & P-Value

Model:



Key to Results:

x1 = 1, x2 = 0 when qwt=idf
x1 = 0, x2 = 1 when qwt=binary
x3 = x1 * (p - $\hat{p}$)
x4 = x2 * (p - $\hat{p}$)

Results:

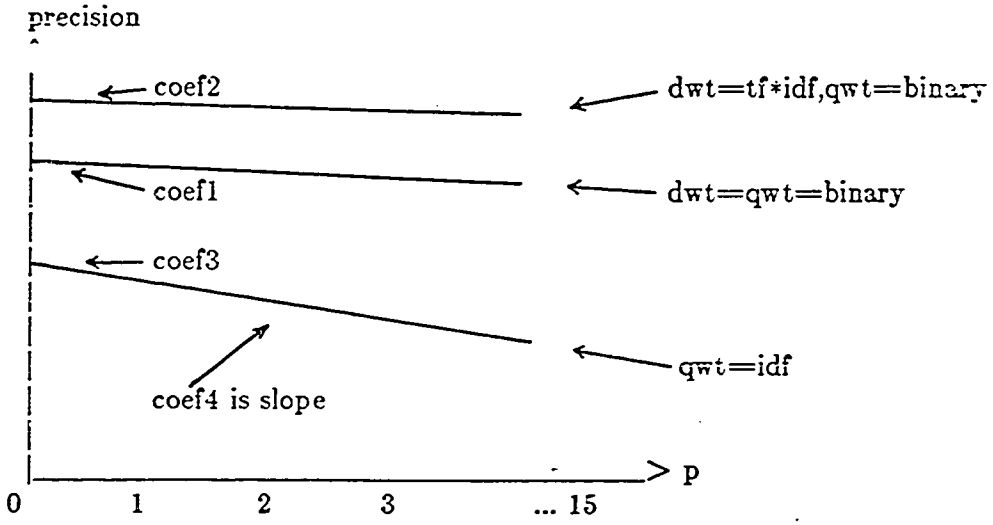
	Coef	Std Err	t Value
x1	0.4982142	0.008489406	58.68659
x2	0.5113571	0.008489406	60.23474
x3	-0.01226661	0.003303985	-3.712672
x4	-0.001873263	0.003303985	-0.566971

Residual Standard Error = 0.03176444 Multiple R-Square = 0.996625
N = 28 F Value = 1771.612 on 4, 24 df

The final model given in Figure 3.5 is clearly the best, judging from the multiple R^2 value. The assumption here is that cases 1 and 2 (both of which have binary query weights) can be represented by two distinct horizontal lines, and that cases 3 and 4 (both of

Figure 3.5: Regression of Precision vs. Query,Doc.Wt. & P-Value

Model:



Key to Results:

x1 = 1, x2 = 0 when dwt=qwt=binary
x1 = 0, x2 = 1 when dwt=tf*idf, qwt=binary
x3 = 1 for qwt=idf
x4 = x3 * (p - $\hat{p}$)
x4 = x2 * (p - $\hat{p}$)

Results:

	Coef	Std Err	t Value
x1	0.4718571	0.004006215	117.7812
x2	0.5508571	0.004006215	137.5006
x3	0.4982142	0.002832821	175.8721
x4	-0.01226661	0.001102504	-11.12614

Residual Standard Error = 0.01059945 Multiple R-Square = 0.999624
N = 28 F Value = 15958.41 on 4, 24 df

which have idf weights on query terms) can be fit by a common sloped line. More elaborate models, with slopes on the lines for cases 1 and 2, do not seem warranted.

The main conclusion, then, is that query weight and p-values are significant factors in predicting the performance of basic p-norm runs. Especially when query term weights are employed, a downward trend in precision is observed for increasing values of p. When no query term weights are used, p-value has little effect. A second conclusion is that when binary query weights are employed, better performance results from using weighted instead of binary document terms.

Interpreting these effects in terms of the p-norm retrieval environment being studied, the following observations are in order.

(1) As expected, overly high p-value degrades performance.

(2) Using binary weights for query terms gives some immunity from the fall off due to high p-values.

(3) Using weighted document terms and binary query terms gives better performance than the pure binary weight case.

(4) The cases with query weights all behave alike, with performance degrading as p increases.

For further proof of the value of p-norm queries, additional experimental results are presented in subsequent sections.

3.3. INSPEC Experiments

3.3.1. Collection

The INSPEC collection contains 12,684 documents together with 77 queries. The subject matter is electrical engineering and computer science, and queries were supplied by students, staff and faculty in those fields at Syracuse University.

The Boolean queries used with the INSPEC documents were based upon traces of searches made using the Diatom system at Syracuse. Since the current version of the Cornell SMART system uses stemming instead of truncation, and does not allow metrical operators (e.g., "within 5 words of" instead of AND), the performance of those queries was not very good, when interpreted in the usual environment of having binary weights on terms and associating $p = \infty$ with the operators.[4] Nevertheless, the various results demonstrating relative improvements provide additional evidence to that already presented for other collections.

3.3.2. Comparisons

Table 3.7 summarizes the INSPEC results. The two base cases are the conventional Boolean one and a cosine run on the original natural language queries. Since the Boolean queries use different terms than those of the natural language (NL terms) statements (due to searchers selecting good terms and supplementing them with others), an additional cosine run was made (case 5 of Table 3.7) with the Boolean logic (BL) query terms. In contrast with Medlars though, the cosine run using Boolean query terms did better than when all of the natural language terms were indexed. This is due in part to the fact that Medlars natural language queries were very precise and short, whereas the INSPEC ones were usually at least a few sentences long. Thus cosine as well as Boolean methods are sensitive to the quality of the query used as input to processing.

Another difference is that in this collection there is no appreciable difference in performance in the upper bound runs – cases 7 and 8 of Table 3.7 – depending on whether BL or

[4]For further discussion of the query characteristics and their retrieval performance, refer to Appendix C Section C.5.

Table 3.7: Summary of Different INSPEC Trials

case no.	p value	doc. wt.	query wt.	description of trial	aver. prec.	%improv. vs. Bool.	%improv. vs. cos.
1	∞	binary	binary	standard Boolean	.116	Base	-50.0
2	2	binary	binary	p-norm	.207	78	-10.8
3	1	tf*idf	binary	p-norm	.275	137	18.5
4	1	tf*idf	idf	p-norm	.265	128	14.0
5	-	tf*idf	tf*idf	cos.BL-terms	.243	109	4.7
6	-	tf*idf	tf*idf	cos.NL-terms	.232	100	Base
7	-	binary	probab.	retrospective indep.case BL terms	.314	171	35.3
8	-	binary	probab.	retrospective indep.case NL terms	.313	170	34.9

NL terms are used. A fairly significant improvement over other methods does occur as expected, but even in these probabilistic based tests, the query performance is rather poor. P-norm relative improvements thus take place both in collections where queries are reasonably good (e.g., ADI) or rather bad (e.g., INSPEC).

The base case (entry 1 in Table 3.7), using strict Boolean queries with binary weights, performed badly; the cosine run of natural language terms (case 6) showed 100% improvement while the cosine run on Boolean query terms (case 5) gave 109% improvement. Simply using low p-values (case 2) gave roughly 78% improvement; using p-values and document term weighting (case 3) resulted in 137% improvement. Thus, the p-norm methods were clearly superior to both Boolean and cosine techniques. These results give strong sup-

port to the hypothesis that p-norm methods would be of value to use in operational retrieval systems. If, given a user's query, on average the system could give much better performance than the usual Boolean approach and slightly better performance than that of vector methods, it would be reasonable to simply generalize the Boolean system to employ p-norm techniques.

A word of caution is in order, however. Since the original Boolean queries performed so badly, and did not employ truncation or metrical operators as are now available on many systems, it could be claimed that p-norm methods would not give as much of an improvement over that situation. In lieu of such a test, however, other collections were selected to further validate the above assertions; the results for the ACM and ISI collections are given in the next sections.

3.4. ACM Experiments

3.4.1. Collection

The ACM collection contains 3204 documents together with 52 queries. The documents are based on titles, and where available, abstracts, of all articles published in the *Communications of the ACM* from the first issue of 1958 through the last one in 1979. Queries were supplied by faculty, staff, and students at Cornell University and at other computer science departments around the U.S.A. during the spring of 1982. Compared to the INSPEC collection, there are fewer documents and the queries are more specific, so precision, especially at low recall levels, was much higher. Two different searchers formulated each Boolean query, so it is possible to see if the relative improvements are confounded with searcher variation.

3.4.2. Comparisons

As can be seen in Table 3.8 the retrieval results are similar to those for Medlars. The two searchers' queries perform fairly similarly and so comparisons between them do not seem appropriate. On the other hand, when relative comparisons are made for each of the searchers, the same trends hold as did for Medlars.

Table 3.8: Summary of Different ACM Trials

case no.	p value	doc. wt.	query wt.	description of trial	aver. prec.	%improv. vs. cos.
1	∞	binary	binary	standard Boolean-searcher 1	.141	-53.5
2	∞	binary	binary	standard Boolean-searcher 2	.115	-62.0
3	2	binary	binary	p-norm- searcher 1	.255	-15.8
4	2	binary	binary	p-norm- searcher 2	.232	-23.4
5	1	tf*idf	binary	p-norm- searcher 1	.354	16.8
6	1	tf*idf	binary	p-norm- searcher 2	.359	18.5
7	1	tf*idf	idf	p-norm- searcher 1	.343	13.2
8	1	tf*idf	idf	p-norm- searcher 2	.351	15.8
9	-	tf*idf	tf*idf	cos.NL-terms	.303	Base
10	-	binary	probab.	retrospective indep.case NL terms	.419	38.3

3.5. ISI Experiments

3.5.1. Collection

The ISI collection (see Appendix C Section C.2.4 for more details) consists of 1460 documents in the field of information science. Bibliographic data supplied by ISI© helped

identify this set, chosen from among the most highly cited articles in the discipline.

Since the ADI collection covered similar subject matter, it was decided to use the queries already available from the ADI collection. Specifically, 35 queries were recorded in their original natural language form and were available for automatic indexing and vector runs. An experiment had been conducted earlier in which Boolean queries were constructed to go along with the vector forms. Three different searchers, referred to as searchers 1, 2, and 3, each carefully considered the original natural language queries, and provided a Boolean expression for each of those queries. Thus, three full sets of Boolean queries were available enabling contrasts between their performance. Finally, exhaustive relevance judgments were made for each of the 35 queries.

3.5.2. Comparisons

Numerous comparisons were made, examining weighting cases, p-value settings, and searcher variations. Results are consistent with those in previous collections and so further comment is held until the summary in Section 3.6.

3.5.3. Lexical Relation Experiment

In the Medlars experiments described in Section 3.2, it was pointed out that query performance was relatively poor due to the replacement of thesaurus terms by all the entries in the respective thesaurus classes.

To explore the effects of term expansion (i.e., replacing a single term by a clause of related terms), it was decided to appeal to the concept of lexical relations of the form described in [Fox 1980]. Based upon the work of Mel'chuk et al. ([Apresyan, Mel'chuk & Zholkovsky 1969] and [Mel'chuk 1973]) on lexical functions and a later study by Evens and

Smith [1979] a grouping had been made of the types of word relationships present in the lexicon. Table 3.9 shows the scheme used here and in [Fox 1980]. Revisions had been suggested[5] but it was expected that their effect on retrieval would be minimal. Further, changes in the groupings or list of relations might have required a considerable amount of time and effort to modify already expanded queries.

The first column of Table 3.9 specifies the 11 groupings and gives the abbreviated name of each relation. The last column gives a brief explanation, following the terminology of Mel'chuk. Finally, the middle two columns provide an example – column two indicates a word being considered and column three provides the word that could be stored in a lexicon as being lexically related to it according to the relationship being considered.

With this list of word relationships, instances of which could be captured in a good lexicon, the following experiment was conducted.

(1) Words lexically related to the terms of the ADI collection were selected. Only terms of low or medium frequency were considered in deciding which words should be expanded.

(2) The ADI queries were expanded to include the list of lexically related words. In this process, words related by antonyms (group B of Table 3.9) were excluded. As an example, consider the word "communication." Lexically related words are shown in Table 3.10; actually they are for the corresponding verb form "communicate."

[5]Personal communications with Joseph Grimes and Martha Evens relating to adding in other relationships that were not included and on reorganizing the groups according to a more coherent and consistent scheme.

Table 3.9: Lexical Relation Groups

RELATION ABBREVIATION	ARGUMENT TO REL.	RELATED WORD	EXPLANATION OF RELATION
A. CLASSICAL			
1.Taxonomy	lion	animal	(GENER) is a kind of
2.Synonymy	amusing	funny	interchangeable word
B. ANTONYMS			
1.Comp	married	single	binary opposition
2.Anti	hot	cold	one denies the other
3.Conv	to buy	to sell	conversiveness
4.Reck	husband	wife	reciprocal kinship
C. GRADING			
1.Queue	Monday	Tuesday	adjacent in a list
2.Set	sheep	flock	(MULT) aggregate
3.Stage	ice	water	(CHILD+)manifestation of
4.Compare	wolf	coyote	typically compared with
D. ATTRIBUTE			
1.Male	duck	drake	unmarked → male
2.Female	lion	lioness	unmarked → female
3.Child	cow	calf	parent → juvenile
4.Home	lion	Africa	origin, habitat
5.Son	dog	bark	characteristic sound
6.Madeof	tire	rubber	substance, made of
7.Color	tomato	red	usual color
8.Time	breakfast	morning	usual time
9.Location	toilet	bathroom	usual place
10.Size	giraffe	tall	usual height or volume
11.Quality	saint	holy	characteristic attribute
E. PARTS-WHOLES			
1.Part	tusk	elephant	has part
2.Cap	tribe	chief	head of organization
3.Equip	gun	crew	name of staff (personnel)
4.Piece	sugar	lump	(SING) item of
5.Comes-from	milk	cow	provenience
6.Poss	rich-man	money	(QUAL1) possesses
F. CASE			
1.Tagent	conquer	conqueror	(usually is S1) agent
2.Tobject	dine	dinner	(S2) direct object
3.Tresult	dig	hole	(Sres) result
4.Tcagent	beat	loser	(can be S3) counter-agent
5.Tinst	sew	needle	(Sinstr) instrument
6.Tsource	sprout	earth	source
7.Texper	love	lover	experiencer
8.Tloc	bake	kitchen	(Sloc) location
9.Tsubject	sell	seller	(S1) subject

Table 3.9 continued: Lexical Relation Groups

RELATION ABBREVIATION	ARGUMENT TO REL.	RELATED WORD	EXPLANATION OF RELATION
F. CASE – cont'd			
10.Tinobj	sell	price	(S4) indirect object
G. PREDICATES			
1.Perm	fall	drop	permit, make possible
2.Incep	difficulty	run into	begin (func. oper.)
3.Cont	peace	maintain	continue
4.Fin	patience	lose	cease, stop
5.Perf	study	mastered	perfective, to have
6.Result	died	dead	resulting state
7.Fact	dream	come true	to become a fact
8.Real	attempt	succeed	make real, fulfill
H. COLLOCATION			
1.Copul	victim	fall	copula, to be
2.Liqu	mistake	correct	destroy,liquidate (verb)
3.Prepar	table	set	prepare (verb)
4.Degrad	teeth	decay	deteriorate
5.Inc	tension	mount	increase (verb)
6.Dec	cloth	shrink	decrease (verb)
7.Bon	conditions	favorable	attribute for "good"
8.Centr	life	prime	culmination
I. MORPHOLOGY			
1.Past	go	went	past tense form
2.PP	go	gone	past participle
3.Plural	man	men	plural form
4.others	fun	funny	any irregular form
J. PARADIGMATIC			
1.Cause	go	send	(CAUS) effect
2.Become	red	redden	verb to get that result
3.Be	near	neighbor	(inv.PRED)that which is
4.Nomv	die	death	(V0,inv.S0)process noun
5.Adjn	sun	solar	(inv.A0)adj. form
6.Able	burn	combustible	(ABLEi)able to
7.Imper	talk	go ahead!	irregular imperative
8.Magn	cold	freezing	very, intensely
9.Mode	style	write	(Smod)mode of action
10.Figur	flame	passion	figurative designation
K. SITUATIONAL			(participants in sit.
1.Ai	fire	burn	generic attribute
2.Operi	sacrifice	make	verb connecting with sit.
3.Funci	silence	reign	verb for subject
4.Labori	torture	put to	verb for action
5.Si	sell	goods	name of participants

Table 3.10: Words Lexically Related to "communicate"

Relationship	Word
Cont	speech
T source	speaker
T inst	word
Part	phoneme

The scheme adopted was to replace all occurrences of selected original words with an appropriate clause, where the related terms were down weighted. Thus, every query containing "communication" had instead, in its place, the clause:

communication
OR
<(speech OR speaker OR word OR phoneme), 0.5>

(3) Since the relative weights would have different effect depending upon how query weights were determined, several schemes for query weighting were employed. In the usual binary scheme, the .5 relative weight was ignored and all terms present have a weight of one; however, the effect of placing the related terms in a separate clause, coupled with the way that p-norm formula interpret this construction, would result in the equivalent of a slight down weighting of related terms. The second scheme, "rw", indicates use of the relative weights (e.g., the .5) as given and uses a weight of 1 when none is shown. The third scheme, "rw*idf", multiplies either the default weight of 1.0 or a supplied relative weight (e.g., .5) by the idf weight of the query term.

Given the above design, the results in Tables 3.11 through 3.13 can be explained as follows:

(1) In Table 3.11 the case of strict Boolean queries with binary weights is given. The down weighting of related terms is essentially ignored due to p = ∞ and binary weights. For searcher 1, with long clauses already in the query, the lexically related words cause a degradation in performance. For the other searchers, with shorter queries, the lexically related terms are of positive value; a slight improvement is seen. At the very low recall levels there is no or hardly any improvement, but improvement occurs at medium and high recall ranges – lexical relations certainly should aid recall.

Table 3.11: ISI Expansion with Lexical Relations – binary wts., $p = \infty$

Searcher No.	Average Precision Original Query	Expanded Query	Percent Change vs. Original
1	0.1118	0.1041	-6.9
2	0.0549	0.0654	+ 19.1
3	0.0653	0.0832	+ 27.4

(2) Table 3.12 covers the case of weighted document and query terms. The scheme "rw" means that the relative weights in the query (e.g., the 0.5 down weighting) are utilized. For all three searchers a slight improvement results. For searcher 3, the improvement is 9%. Searcher 3's queries are fairly deeply nested – i.e., the expression trees are tall and thin instead of wide and bushy like those of searcher 1. Hence lexical relations, which cause expansion or widening of nodes, affect short clauses more than long ones (since long ones have already been expanded somewhat by the searcher), a possible reason why the improvement for searcher 3 was so much more.

Table 3.12: ISI Expansion with Lexical Relations - dwt=tf*idf, qwt=rw, p=1

Searcher No.	Average Precision Original Query	Expanded Query	Percent Change vs. Original
1	0.1835	0.1866	+ 1.7
2	0.1604	0.1662	+ 3.6
3	0.1802	0.1963	+ 8.9

(3) Table 3.13 shows results for each of the three searchers. Document weight is fixed at tf*idf and $p = 1$ is used. For a particular searcher, a pair including the regular and expanded results are given for each of the three types of query weights mentioned above.

As was mentioned earlier for the other weight cases, the queries prepared by searchers

Table 3.13: ISI Expansion with Lexical Relations - dwt=tf*idf, $p = 1$

Searcher No.	Query Weight Method	Average Precision Original Query	Expanded Query	Percent Change vs. Original
1	bin	0.1835	0.1869	+ 1.9
1	rw	0.1835	0.1866	+ 1.7
1	rw*idf	0.1726	0.1788	+ 3.6
2	bin	0.1604	0.1694	+ 5.6
2	rw	0.1604	0.1662	+ 3.6
2	rw*idf	0.1575	0.1652	+ 4.9
3	bin	0.1802	0.1949	+ 8.2
3	rw	0.1802	0.1963	+ 8.9
3	rw*idf	0.1728	0.1823	+ 5.5

1 and 2 show slight improvements in all of the paired comparisons. Searcher 3's queries show even greater improvements.

In conclusion, then, it is clear that the effects of adding lexically related terms is not a very strong one. A mild improvement generally results when the new terms are added in the form used in this experiment. For certain searchers or types of queries, the improvement is more dramatic. It appears to be the case that when Boolean queries are relatively short and of fairly good quality – a situation that only occurred for searcher 3 – then lexical relations are of most benefit.

Regarding further applications, it should be noted that the use of lexical relations in this context is a well defined concept that could be automated if a suitable machine readable lexicon were available. Indeed, Martha Evens[6] has made similar tests (without weights, however) to these with recent experiments, using automatically identified lexically related terms.

3.6. Summary and Conclusions

3.6.1. Principal Results for All Collections

For ease of reference, the key results from the various experiments are summarized in Tables 3.14 through 3.16. That is, the essential ADI, Medlars, INSPEC, ACM, and ISI results are all charted below.

[6]Personal communication, July 1982.

Table 3.14: Average Precision for Key Cases, All Collections

Collection Name	Searcher Identifier	NL Cos tf*idf	Strict Boolean wt=bin	p=1 Boolean wt=bin	p=1 qwt=bin dwt=tf*idf
ADI 2 clause	author	.7901	.6354	.7512	.8321
Medlars	NLM	.5473	.2065	.4710	.5505
INSPEC	Syracuse 7	.2325	.1159	.1911	.2747
ACM	1)student a	.303	.1414	.2495	.3542
	2)student b	.303	.1156	.2273	.3594
ISI	1)author	.1589	.1118	.1687	.1835
	2)librarian a	.1569	.0549	.1502	.1604
	3)librarian b	.1569	.0653	.1727	.1802

Table 3.15: Average % Change for Key Cases vs. Boolean Queries

Collection Name	Searcher Identifier	NL Cos tf*idf	p=1 Boolean wt=bin	p=1 qwt=bin dwt=tf*idf
ADI 2 clause	author	24.4	18.3	31.0
Medlars	NLM	165.0	128.1	166.6
INSPEC	Syracuse 7	100.6	64.9	137.0
ACM	1)student a	114.9	76.6	151.1
	2)student b	163.5	98.3	212.2
ISI	1)author	40.3	50.9	64.1
	2)librarian a	185.8	173.6	192.2
	3)librarian b	140.3	164.5	176.0

Table 3.16: Average % Change for Key Cases vs. Cosine

Collection Name	Searcher Identifier	Strict Boolean wt=bin	p=1 Boolean wt=bin	p=1 qwt=bin dwt=tf*idf
ADI 2 clause	author	-19.6	-4.9	5.3
Medlars	NLM	-62.3	-13.9	0.6
INSPEC	Syracuse 7	-50.2	-17.8	18.2
ACM	1)student a	-53.5	-17.8	16.8
	2)student b	-62.0	-24.8	18.5
ISI	1)author	-28.7	7.5	17.0
	2)librarian a	-65.0	-4.3	2.2
	3)librarian b	-53.8	10.1	14.9

Table 3.14 shows a single measured value of the merit of each run – the average of the precision values at recall levels .25, .50, .75 – for various collections and retrieval methods.

Natural language (NL) queries were available for each collection as were a corresponding set of Boolean logic (BL) queries; for CACM two students reformulated questions into Boolean expressions while for the ISI collection, three different searchers, namely the author and two librarians, constructed Boolean queries.

The key cases shown in those tables allow comparison between the standard vector, the standard Boolean, and two of the new p-norm trials. Table 3.14 shows the actual precision values. Table 3.15 shows percent changes, where the original Boolean queries are used as a base for comparison. However, in many of the collections, the Boolean queries did very badly. Hence, the cosine run, made on the automatically indexed NL queries, is used as the base for percentage changes given in Table 3.16.

3.6.2. Discussion

Five different collections, of differing sizes and composed of documents in different subject areas, were used to compare the performance of p-norm methods with those of the standard Boolean and cosine techniques. In all cases the Boolean queries for $p = 1$ gave significant improvements (i.e., 24–186%) over the original Boolean queries. Using document weights also, according to a scheme similar to the tf*idf formula used for cosine runs gave further improvement. The p-norm scheme with weights and $p = 1$ did at least slightly better than the standard cosine scheme in all cases. In the two cases where the initial Boolean queries were very much worse than the cosine runs, the weighted p-norm scheme was only marginally better than the cosine case; for other sets of queries the improvement over the cosine run ranged from 5% to 18%.

Though these improvements over cosine behavior are not as substantial as those resulting from the use of feedback techniques, they are of interest for a number of practical and theoretical reasons:

(1) Existing Boolean systems could be changed to use p-norm techniques in a stepwise fashion. Thus, using $p = \infty$ and document weights is a reasonable initial change, requiring no alteration in the Boolean logic operations of union and intersection of inverted lists. Since for binary query weights the effect of high p-values is not as noticeable as when query weights are used, some improvement would result and the document weights could initially be based on a scheme like idf using data available from the postings file. A similar scheme was used successfully in the Syracuse SIRE system [McGill & Noreault 1977].

After such an initial change, the relaxing of p-values could be partially implemented.

First, computing the similarity using lower p-values could be done simply, for document ranking purposes only, after retrieval was performed based on $p = \infty$. Second, the retrieval logic could be modified so as to retrieve documents beyond those which fully satisfy the $p = \infty$ Boolean logic - e.g., also retrieve documents matched by any low frequency term in the query, or any pair of high frequency terms, or perhaps, any combination of two or more terms with total idf weight above a certain threshold. Algorithms could be devised which would result in retrieving no more than a specified upper limit on the number of documents, L, and which would initially select documents with highest similarity to the p-norm equation. Thus, the efficiency of an inverted file system would be partially preserved. Alternatively, clustered document schemes could be employed as have been proposed for use with the vector model (see Chapter 7 for more details on clustering).

(2) Some experienced searchers prefer to use Boolean queries rather than vector methods, feeling that they have more control and can be more precise in stating an information need through Boolean logic. For such searchers, their favorite form of query would still be handled, and yet the p-norm methods would improve the effectiveness of retrieval. Similarly, for untrained users of such a system, a list of key words could be automatically converted into a query with a single outer operator and $p = 1$; using weights on the terms, a run like the cosine case would result.

(3) As will be discussed in later chapters, the p-norm model allows queries which are a mixture of two parts - one which would probably utilize $p = \infty$ since it requires exact matching of certain factual information (e.g., AUTHOR = Smith), along with a second part needing less strict matching - e.g., lists of terms ORed together with low

p-values. Though a certain amount of training would be required to form such queries, the p-norm notation is clearly adequate to express such information needs. An interesting idea for future research would be to train searchers to prepare such complex, mixed queries and see how well they perform.

(4) Finally, the p-norm model enables one to better understand various phenomena about Boolean and vector systems. Thus, the graph and regression analysis of Medlars results pictorially and quantitatively show the interaction between p-values and weighting schemes.

The above discussion brings out several points that lead into the subsequent chapters. Specifically,

(1) Since some might feel that p-norm methods further complicate the process of Boolean query construction, which is already fairly difficult, it would be desirable to automatically construct p-norm queries. Techniques for doing so are given in Chapter 4.

(2) Since many Boolean queries are not optimal, and p-norm performance is reduced if the basic query is poorly formed, a means for reformulating Boolean queries would be useful. Chapter 5 therefore deals with automatic feedback methods used to improve Boolean and p-norm queries.

Building upon the p-norm model described in Chapter 2 and the explanation and validation of the basic approach given in this chapter, further chapters will further develop the methodology for practical use of p-norm techniques.

CHAPTER 4

AUTOMATIC P-NORM QUERY CONSTRUCTION

4.1. Introduction

Boolean logic based search systems require that users describe queries in the form of Boolean expressions. The computer retrieves documents satisfing those formulas. Since the retrieval quality of the original query greatly affects search performance, even when p-norm enhancements are made (cf. Chapter 3), it is of interest to examine strategies recommended for constructing good queries. This topic is covered in Section 4.2.

Since constructing a good query is a relatively difficult task, the question arises whether such a process could be automated, thereby relieving the user of that burden. If Boolean queries could be automatically constructed with performance approaching that of manually constructed queries, and p-norm methods could be automatically used to improve performance to exceed that of manually constructed queries, a simpler and more effective retrieval scheme could be implemented. In addition, users would not have to learn the intricacies of manual p-norm query construction, since that would be done for them by the computer. A two step process is therefore suggested to arrive at mechanistically generated p-norm queries: 1) automatically construct regular Boolean queries from a list of terms, and 2) interpret those queries using experimentally tested types of assignments of p-values and weights. Alternatively, the two steps could be collapsed into one, where suitable p-norm expressions are directly produced. In any case, the starting point should simply be a list of keywords or key phrases.

Two possible methods proposed for automatic query construction are described in Sections 4.3 and 4.4. Though both are in accord with the assumptions of the theory of frequency based retrieval, these techniques approach the process from slightly different perspectives. Section 4.5 therefore compares and summarizes the relevant results.

Since automatic query construction techniques here described ignore any information fed back by the user as part of an interactive search process, that topic will be taken up in Chapter 5, which focuses on automatic query reconstruction using feedback methods.

4.2. Background – Strategies for Searching

In describing "An Expert System for Document Retrieval," Yip [1979] explains the guidance given by EXPERT-1, a experimental system aimed at aiding or replacing search intermediaries through the use of interactive computer dialogues. Essentially, the approach is to state the topic of interest and then decompose it into meaningful concepts. One formulates each concept as the union of search terms, and then constructs the query as the intersection of the expressions representing those concepts.

Meadow and Cochrane [1981] devote an entire chapter to strategies for searching concept groups and terms. Though the process they suggest for negotiating with users to identify information needs is not germane to the discussion, the various approaches they recommend for query construction may be of interest. Exploring those alternative approaches, described below, should help explain why Boolean query formulation is difficult, and should illustrate the roles played by term frequencies and Boolean operator selection.

(1) Most Specific Facet (i.e., term or phrase)

Select a facet that has few postings and is a good indicator of the desired result. Combine it with variant spellings using the OR connective.

(2) Lowest Postings Facet

Select the term or phrase estimated to have the fewest postings. Add other facets using the OR connector until enough useful documents are retrieved.

(3) Building Blocks

This is essentially the approach taken in EXPERT-1. Key concepts become blocks, and facets relating to those are combined using the OR operator. Figure 4.1 shows three building blocks for a hypothetical query, "Estimation problems in probabilistic IR". Finally, the AND operator connects all of the blocks.

Figure 4.1: Building Blocks Strategy

Blocks:

Block 1	Block 2	Block 3
estimation parameters **Jeffrey's prior**	probabilistic **BLE**	IR information retrieval **feedback retrieval**

Key to Forming Query from Blocks:

Bold faced words or phrases in blocks identify those entries which have fewest postings.

Resulting Query:

("estimation" OR "parameters" OR "Jeffrey's prior")
AND
("probabilistic" OR "BLE")
AND
("IR" OR "information retrieval" OR "feedback retrieval")

---

(4) Citation "Pearls" (i.e., grow around a small beginning)

Given a building block formulation, select the most specific facet of each block. The bold face entries in Figure 4.1 identify those expected to have fewest postings. Assuming that they are the most specific terms, the query would then be:

("Jeffrey's prior" AND "BLE" AND "feedback retrieval")

After a few documents are retrieved, they are examined to supply additional terms for an improved query. Essentially, feedback is being performed manually.

(5) Successive Fractions / Divide and Conquer

Beginning with a broad initial formulation, repeatedly make revisions that limit the result to some, possibly large, fraction of the previously identified set. One might first select a range of years, secondly apply some general subdivision to isolate a field of interest, and then continue adding in additional restrictions as conjuncts.

Based on the above descriptions of query formulation techniques, one can see that Boolean queries are constructed:

(1) using appropriate elements so as to produce a retrieved set of manageable size,

(2) using the number of postings for each term, and the size of the retrieved set for each query subexpression,

(3) using the OR operator to combine terms that are almost synonymous,

(4) using the AND operator to achieve greater specificity, by combining OR clauses,

(5) using retrieved documents as the basis for new query forms, as in feedback.

It is not surprising that Boolean queries can achieve good performance, when term combinations are properly organized. It is also not surprising, though, that some relevant documents can be missed, especially if many AND operators are utilized. By way of example, consider Figure 4.1 again, and note that a relevant document with facets "Jeffrey's prior" and "feedback retrieval" would be missed if neither "probabilistic" nor "BLE" appeared in that document, even though it would no doubt be relevant in any case.

While these strategies are easily understood, they are difficult to put into practice. Hence, it is desirable either to have automatic routines for them or to provide alternative implementations of the underlying principles. The next two sections explore this question, and describe two such possible alternative schemes for automatic query construction.

4.3. Frequency Range Based Construction

The first automatic query formulation scheme tested is one based on the frequency approach to retrieval, where the number of postings for each term is the only determinant of how it will be used in a query. In particular, term discrimination theory provides a number of insights that can be applied to forming p-norm queries.

4.3.1. Term Discrimination Theory Application

Mention has already been made of the simple idea of using AND clauses for connecting elements of phrases and OR clauses to link synonyms or lexically or semantically related words. Lacking adequate tools for easily identifying automatically such relationships in large document collections by linguistic analysis, it has been suggested that statistical information might be used instead [Salton 1975].

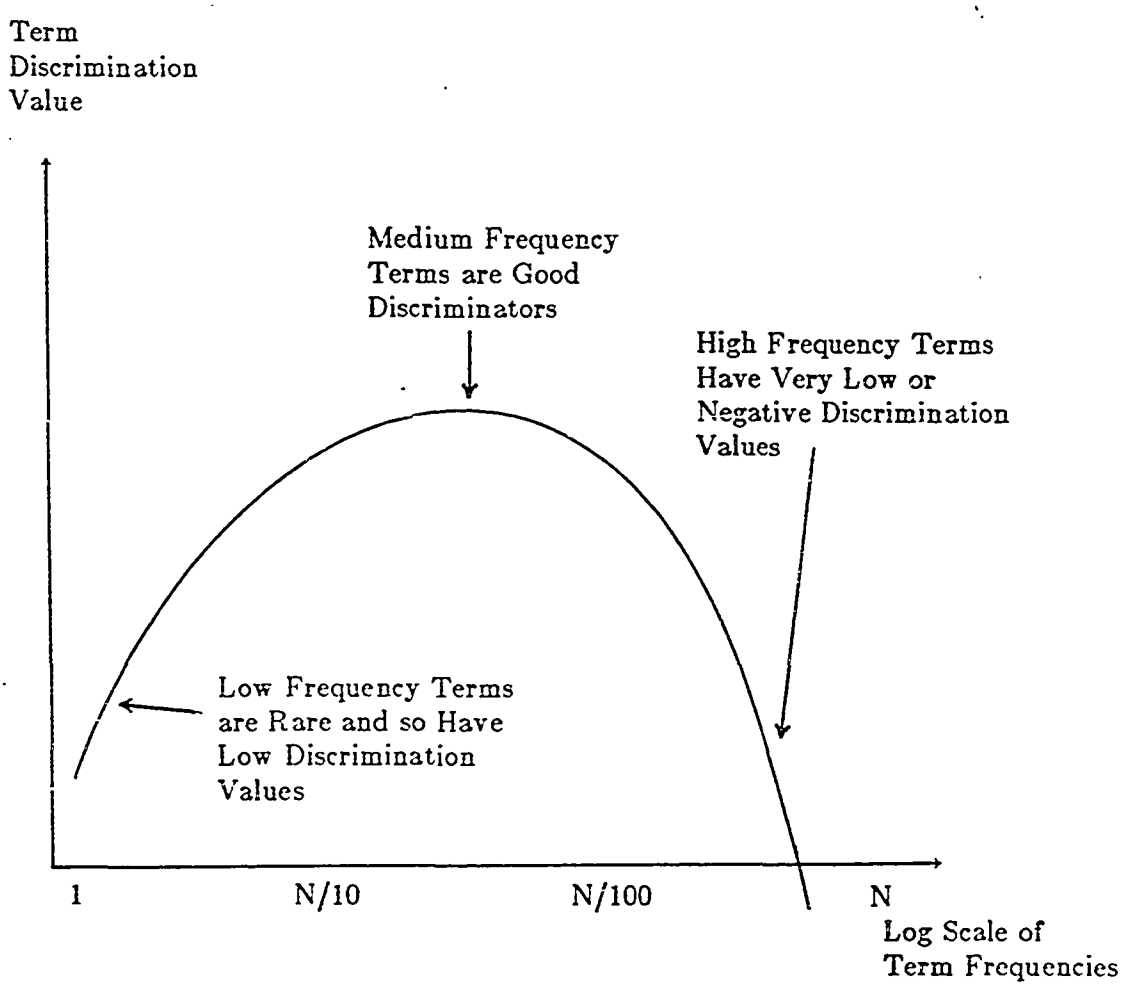
Several such schemes rely for theoretical and experimental support on work done with term discrimination values [Salton, Yang & Yu 1975]. Each term in the dictionary (of all terms) for a collection can be analyzed to see how well it alone can be used as a discriminator. Adding a good term to a space where it is not present should make the space less compact since good discriminators separate things well. Discrimination value measures how much adding a term causes the "density" of the vector space to be decreased (by good discriminators); for bad discriminators the discrimination value may be negative, reflecting an increase in density if those terms are added. If one assumes that discrimination values of terms can be accurately determined by positing independent assignment of those terms to documents, and furthermore that these values are reliable indicators of term quality, then discrimination values can help identify which terms should be utilized and which should be eliminated from queries.

High frequency terms are poor discriminators since they are usually assigned to many documents scattered all over the vector space. Hence, they should be omitted from queries. Alternatively, they could be used only when in a phrase, such as when ANDed together with other medium or high frequency terms.

As expected, terms with middle frequency values tend to occur in a moderate number of documents reasonably close together in the vector space, and so typically exhibit positive discrimination values. Terms with low document frequency are so rare and specific that they cannot retrieve very many of the documents relevant to given queries – they exhibit low discrimination values. However, a clause formed as the disjunction of such suitably chosen low frequency terms could exhibit fairly high discrimination ability.

Based on term discrimination tests, Figure 4.2 illustrates the variation of term discrimination values versus term frequency (postings). Since term frequency values (postings) are commonly available in retrieval systems whereas discrimination values require a good deal of computation to arrive at, it is suggested that following the correspondence shown between the two values term frequencies be appropriately utilized instead of discrimination values for determination of phrases or thesaurus classes. That correspondence is summarized as follows:

Figure 4.2: Term Discrimination Value versus Term Frequency



(1) Terms with very low document frequency exhibit low discrimination values.

(2) As the document frequency of terms rise, the discrimination value of those terms improve up to a maximum point that is reached for medium frequency terms.

(3) As the document frequency increases further, discrimination values rapidly become worse.

(4) Very high frequency terms have the worst discrimination values.

The vector space model typically uses a simple list of weighted terms identified from an original natural language query after applying a stop word list and after stemming each remaining word. Better performance could be expected if besides using simple word stems, thesaurus classes and phrases could be included according to the following principles:

(1) Low frequency terms should be broadened by incorporating several related low-frequency terms into a single OR clause, i.e., prefix the OR connective to a list of low-frequency terms to make a clause or subexpression of a Boolean or p-norm query.

(2) Medium frequency terms should be used as single terms without further ado.

(3) High frequency terms should be transformed into more specific entities by forming AND clauses, where the AND connective is used to prefix a list of high frequency terms to make a clause or subexpression that is sufficiently narrow in scope.

An excellent example occurs in query 1 of the ADI two clause collection, where "catalog" and "catalogue" both have very low frequency and so should be combined into a thesaurus class

"catalog" OR "catalogue".

Likewise, "computerization" and "automation" both have high frequency and so could be combined into the phrase

"computerization" AND "automation".

In the vector model, using phrases and thesaurus classes does not naturally fit in, and requires special static collection-wide processing and possible partial re-indexing. Certainly the Boolean model seems a more appropriate vehicle since each of the relevant phrases and thesaurus classes can be directly and dynamically specified for a given query.

However, in the Boolean model it is not obvious how to link the various clauses, or how to include the middle frequency terms. In the above mentioned query it is unclear how to combine the three elements:

low frequency class:	"catalog" OR "catalogue"
medium freq. term:	"mechanization"
high freq. phrase:	"computerization" AND "automation".

Since it appears that an acceptable solution might have to combine both Boolean logic and vector notions, the sensible approach would be to use a p-norm query such as

OR^1(OR^2(catalog, catalogue),
mechanization,
AND^2(computerization, automation)).

Here, an outer operator with $p = 1$ is used to link the various query components – OR clauses for low frequency terms, individual entries for medium frequency terms, and AND clauses for high frequency terms. Assuming that five frequency ranges are used instead of the four shown above, and defining outer operator $OP^{p\text{-}outer}$ (where OP is either AND or OR), one might have the following general query format:

$$OP^{\text{p-outer}}[\qquad (4\text{-}1)$$

$$< OR^{p1}(< c_{11}, w_{11} >, \cdots, < c_{1m}, w_{1m} >), w_1 >, \qquad \text{(very low freq.)}$$

$$< OR^{p2}(< c_{21}, w_{21} >, \cdots, < c_{2m}, w_{2m} >), w_2 >, \qquad \text{(med. low freq.)}$$

$$< OR^{p3}(< c_{31}, w_{31} >, \cdots, < c_{3m}, w_{3m} >), w_3 >, \qquad \text{(med. freq.)}$$

$$< AND^{p4}(< c_{41}, w_{41} >, \cdots, < c_{4m}, w_{4m} >), w_4 >, \qquad \text{(med. high freq.)}$$

$$< AND^{p5}(< c_{51}, w_{51} >, \cdots, < c_{5m}, w_{5m} >), w_5 >] \qquad \text{(very high freq.)}$$

where typical parameter assignments might be

p-outer = p3 = 1
p1 = p5 = 2
p2 = p4 = 1.5

and weights could either be binary values (i.e., 0 or 1), or else based on idf values.

To implement these ideas, it is necessary to have appropriate techniques to:

(1) Determine the correct number of frequency ranges to use.

(2) Identify term frequency values that delimit the frequency ranges – e.g.. determine what is the highest frequency for "very low frequency" terms and what is the highest frequency for the next range, "med. low frequency".

(3) Select suitable p-values for each range.

(4) Determine weights w_{ij} for term j of clause i.

(5) Determine weights w_i for each clause i.

(6) Select the appropriate connective for OP and the correct value for p-outer.

Preliminary exploration of theory-based ideas for making these decisions was done with the ADI two clause queries. Further experiments were carried out with the Medlars collection.

4.3.2. Frequency Range Medlars Experiments

Since there were only 82 documents in the ADI collection, the frequencies of terms necessarily covered a fairly narrow range. In order to explore the applicability of frequency range based query construction methods, the larger collection of 1033 Medlars documents was employed. To demonstrate what such queries actually look like an example is described in detail in Section 4.3.2.1. That example is then followed by a discussion of the design and results from experiments using automatic queries constructed with the frequency range method.

4.3.2.1. Example

The natural language form of the second Medlars query is shown in Figure 4.3.

Figure 4.3: Natural Language Form of Medlars Query 2

The relationship of blood and cerebrospinal fluid oxygen concentrations or partial pressures. A method of interest is polarography.

After automatic indexing the terms are assigned concept numbers and weights are computed according to the idf scheme. Table 4.1 shows the results.

The frequency range method separates terms into clauses based on the frequency limits set for each range. In one experiment the eight ranges shown in Table 4.2 were chosen, to ensure that the number of terms in classes were roughly the same. Every term of query 2 is listed under the proper range and a p-value for each range is suggested. The following sections will explain the basis for these assignments in more detail.

Table 4.1: Frequency and IDF Weights for Terms in Medlars Query 2

Concept Number	Actual Term	Collection Frequency	Idf Weight
6633	relationship	55	2.698
872	blood	148	1.27
1168	cerebrospinal	17	4.392
2878	fluid	45	2.988
5565	oxygen	30	3.573
1485	concentration	95	1.91
5662	partial	96	1.895
6197	pressure	339	0.075
4762	method	121	1.561
3969	interest	15	4.573
6009	polarography	2	7.48

Table 4.2: Frequency Ranges Used in Example of Query Clauses

Upper Frequency Value Allowed in Range	Boolean Connective Chosen	Possible P-Values	Terms from Example in Range
2	OR	2	polarography
3	OR	1.7	-
6	OR	1.5	-
10	OR	1	-
20	AND	1	cerebrospinal
50	AND	1.3	fluid, oxygen
100	AND	1.5	concentration, partial, relationship
1033	AND	2	blood, method, pressure

4.3.2.2. Experimental Design

The frequency range method requires that many parameter settings be determined to specify how queries should be formulated and how query-document similarities should be computed. Decisions must be made regarding:

(1) Document weighting – e.g., binary or tf*idf

(2) Query term weighting – e.g., binary or idf

(3) Query clause weighting – e.g., binary or average of idf values of sub-clause entries

(4) Outer p-value – e.g., $p = 1$ or $p = 2$

(5) Terms used to form the query from – e.g., the original natural language query or the Boolean logic query after expansion of MESH thesaurus classes

(6) Frequency ranges chosen (see 8 cases described later)

(7) Assignment of operators and p-values to each frequency range.

Since options 4, 6, and 7 could lead to infinite numbers of possible settings and since there are 16 combinations of the other four variables listed above, a factorial design is impossible. Almost 50 cases were selected to try to cover the space of reasonable assignments to parameters and to enable a number of contrasts to be made. That is, a number of pairs of selected cases differed only in a single setting, so if all such pairs showed preference for one particular setting, at least a tentative conclusion could be reported favoring that better setting. Statistical methods could be applied later to formalize these findings, if such validation seemed appropriate.

The essential feature of the frequency range theory that distinguishes it from other methods is that of subdividing the set of terms into ranges based on their frequency of occurrence in documents. Therefore, eight different ways of splitting the Medlars frequency range were suggested.

(1) Since the ADI results were fairly good the idf ranges used for dividing between successive ranges were scaled to match the wider set of possible Medlars values and corresponding frequencies were selected. This first set of ranges was 1-3, 4-21, 22-90, 91+ ; the four ranges are then analogous to the ADI ranges.

(2) According to findings with the Medlars collection made during term discrimination value studies [Salton, Yang & Yu 1975], discrimination rank falls rapidly as frequency increases, reaching a low point at frequency=10, coming up part way to a plateau at frequency 20, and reaching a maximum around frequency=50. Two guidelines often used are the points n/10 and n/100 which identify the low and high ends of middle frequency terms; when n=collection size=1033 as in Medlars, those values are roughly 100 and 10. Using these guidelines and desiring to have at least four ranges, five of the remaining seven sets of ranges were chosen. Set number two, for example, aimed at having a wide range for low frequency terms, and using points 20, 50, and 100 mentioned above. Hence, the four ranges were 1-20, 21-50, 51-100, and 100+ .

(3) Based on the previous case, but separating out terms occurring in only a few documents, gave five ranges: 1-2, 3-20, 21-50, 51-100, 100+ .

(4) Splitting the low frequency ranges further gives five ranges: 1-3, 4-8, 9-20, 21-100, 100+ .

(5) Using the cutoff point, frequency=10, gives five ranges: 1-10, 11-20, 21-50, 51-100, 100+ .

(6) Of interest is whether more ranges would help or hurt. Dividing the low and middle frequency ranges further gives eight ranges: 1-2, 3, 4-6, 7-10, 11-20, 20-50, 51-100, 100+ .

(7) Going to nine ranges and using upper limits of form

$$1 + 2^{k-1} \qquad \text{for } k = 1, 2, ..., 8$$

yields 1-2, 3, 4-5, 6-9, 10-17, 18-33, 34-65, 66-129, 129+ .

(8) Again using nine ranges, but selecting upper points 2^k, results in 1-2, 3-4, 5-8, 9-16, 17-32, 33-64, 65-128, 129-256, 256+ .

With ranges identified, a methodology for assigning operations and p-values to each range was needed. The obvious first approach was to use constant $p = 1$ everywhere. Alternatively, one could assign operators and p-values so that OR^z was used for lowest frequency terms, AND^y for highest frequency terms, and $OR^{z'}$ with $z' < z$ or $AND^{y'}$ with $y' < y$ for clauses in between. Typically, then, $OR^1 = AND^1$ would be used for clauses of medium frequency terms.

Actually, p-value assignments fell into 3 classes:

(1) $x = y = 1$, i.e. constant p-value

(2) $x = y = 2$, i.e. values of p ranging from 2 down to 1 and then up to 2

(3) other cases of handling x and y, such as $x = 2$ and $y = 3$.

From the above discussion, it can be seen which values were chosen from, in order to arrive at each of the test cases selected. Results of those tests, as given in the next section, should be easily understood.

4.3.2.3. Results

Tables 4.2, 4.3, and 4.4 summarize the results of the 47 test runs made using frequency range methods on the Medlars collection. Initial contrasts were made as shown in

Table 4.2 while more detailed tests for two particular sets of ranges are summarized in the other two tables.

The first test made is that comparing the use of terms from the natural language statement (NL) against the use of terms in the expanded Boolean logic query (BL). In Table 4.2, cases 1-4 use NL while 5-8 use BQ; pairwise contrasts show in each of the four situations that NL terms are preferable, in keeping with similar tests made earlier. This conclusion is further supported in Table 4.3 by contrasts between cases 1-4 and 9-12. In Table 4.4, contrasts supporting the above evidence are cases 1-2 versus 3-4 and 5-9 versus 15-19. Hence, attention will hereafter be focused on the use of NL terms.

The second test made compares the use of tf*idf versus binary weights, on document terms. Evidence that tf*idf is better is given by the following contrasts: Table 4.2 cases 3-4 versus 1-2, cases 7-8 versus 5-6; Table 4.3 cases 1-4 versus 5-8; Table 4.4 cases 2 versus 1, 4 versus 3, and cases 5-9 versus 10-14. Clearly, using tf*idf document weights is preferred.

There seems not to be any problem of interaction between the two parameters studied above; using NL terms and tf*idf documents is encouraged as can be seen in Table 4.2 by contrasting cases 3-4 with either 1-2 or 5-6 or 7-8. Similarly, in Table 4.3, cases 1-4 are better than 5-8 or 9-12. Finally, in Table 4.4, case 2 surpasses 1, 3, and 4 and cases 5-9 are better than 10-14 or 15-19. In summary, then, NL terms and tf*idf documents are a good combination.

A third question deals with how many frequency ranges are appropriate, and how the best breakdown of the frequency spectrum into ranges can be made. Fixing attention for the moment on when all p-values are set to 1.0, and assuming the NL terms and tf*idf document weights are employed, then if idf weights are used on query terms and clauses,

Table 4.2: Medlars Freq. Range Queries – Initial Contrasts

Case No.	Doc. Wt.	Query Wt.	Clause Wt.	Outer p	Term Set	Ranges	Frequency Ranges and Operators	Aver. Prec.
1	Bin.	Bin.	Bin.	∞	NL	5	(1-10):OR^{∞} (11-20):AND^{∞} (50-100+):AND^{∞}	.1904
2	Bin.	Bin.	Bin.	∞	NL	4	(1-20):AND^{∞} (50-100+):AND^{∞}	.1302 .1302
3	Tf*idf	Bin.	Bin.	∞	NL	5	(1-10):OR^{∞} (11-20):AND^{∞} (50-100+):AND^{∞}	.3558
4	Tf*idf	Bin.	Bin.	∞	NL	4	(1-20):AND^{∞} (50-100+):AND^{∞}	.2498 .2498
5	Bin.	Bin.	Bin.	∞	BL	5	(1-10):OR^{∞} (11-20):AND^{∞} (50-100+):AND^{∞}	.1342
6	Bin.	Bin.	Bin.	∞	BL	4	(1-20):AND^{∞} (50-100+):AND^{∞}	.0960
7	Tf*idf	Bin.	Bin.	∞	BL	5	(1-10):OR^{∞} (11-20):AND^{∞} (50-100+):AND^{∞}	.2753
8	Tf*idf	Bin.	Bin.	∞	BL	4	(1-20):AND^{∞} (50-100+):AND^{∞}	.1678
9	Tf*idf	Idf	Av.Idf	1	NL	4	(1-3-21):OR^{1} (22-90+):AND^{1}	.5055
10	Tf*idf	Idf	Av.Idf	1	NL	4	(1-3):$OR^{1.5}$, (4-21):$OR^{1.2}$ (22-90):$AND^{1.2}$ 90+ :$AND^{1.5}$	.5148
11	Tf*idf	Idf	Av.Idf	1	NL	4	(1-2-20):OR^{1} (21-100+):AND^{1}	.5170

(Continued on Next Page)

Table 4.2 cont'd: Medlars Freq. Range Queries – Initial Contrasts
(with Doc.Wt=tf*idf, Query-Wt=Idf, Clause-Wt=Av.idf,
Outer-P=1, Term-Set=NL)

Case No.	No.of Ranges	Frequency Ranges and Operators	Aver. Prec.
12	4	(1-2):$OR^{1.5}$, (3-20):$OR^{1.2}$ (21-100):$AND^{1.2}$ 100+ :$AND^{1.5}$	.4990
13	10	(1-2-3):OR^{1} (5-9):OR^{1} (17-33):OR^{1} (34-65):AND^{1} (129-257+):AND^{1}	.5164
14	10	(1-2):OR^{2}, 3:$OR^{1.7}$ (4-5):$OR^{1.5}$, (6-9):$OR^{1.3}$ (10-17):$OR^{1.2}$, (18-33):OR^{1} (34-65):$AND^{1.3}$, (66-129):$AND^{1.5}$ (130-257):$AND^{1.7}$, 258+ :AND^{2}	.5135
15	9	(1-2-4):OR^{1} (8-16-32):OR^{1} (33-64):AND^{1} (128-256+):AND^{1}	.5071
16	9	(1-2):OR^{2}, (3-4):$OR^{1.7}$ (5-8):$OR^{1.5}$ (9-16):$OR^{1.3}$ (17-32):OR^{1}, (33-64):$AND^{1.3}$ (65-128):$AND^{1.5}$, (129-256):$AND^{1.7}$ 257+ :AND^{2}	.5135

Table 4.3: Medlars 5 Freq. Range Queries – Contrasts

Operators are Assigned to Ranges:

Start of Range	Operator	P-Value Variable
1	OR	p1
4	AND	p2
9	AND	p3
21	AND	p4
101	AND	p5

Results for Trials: (clause wt.=av.idf)

Case	Group	Doc.	Query	Outer	Term	P-Values for Ranges					Aver.
No.	Entry	Wt.	Wt.	p	Set	p1	p2	p3	p4	p5	Prec.
1	A1	Tf*idf	Idf	2	NL	2	1	1.5	2	3	.4883
2	A2	Tf*idf	Idf	1	NL	2	1.5	1	1.5	2	.4966
3	A3	Tf*idf	Bin	1	NL	2	1.5	1	1.5	2	.4759
4	A4	Tf*idf	Idf	1	NL	1	1	1	1	1	.5025
5	B1	Binary	Idf	2	NL	2	1	1.5	2	3	.4518
6	B2	Binary	Idf	1	NL	2	1.5	1	1.5	2	.4398
7	B3	Binary	Bin	1	NL	2	1.5	1	1.5	2	.4219
8	B4	Binary	Idf	1	NL	1	1	1	1	1	.4430
9	C1	Tf*idf	Idf	2	BQ	2	1	1.5	2	3	.3453
10	C2	Tf*idf	Idf	1	BQ	2	1.5	1	1.5	2	.3590
11	C3	Tf*idf	Bin	1	BQ	2	1.5	1	1.5	2	.3613
12	C4	Tf*idf	Idf	1	BQ	1	1	1	1	1	.3756

Ordering of Cases in Each Group:

A: $4 > 2 > 1 > 3$

B: $1 > 4 > 2 > 3$

C: $4 > 3 > 2 > 1$

Table 4.4: Medlars 8 Freq. Range Queries – Contrasts

Operators are Assigned to Ranges:

Start of Range	Operator	P-Value Variable
1	OR	p1
3	OR	p2
4	OR	p3
7	OR	p4
11	OR	p5
21	AND	p6
51	AND	p7
101	AND	p8

Results for Trials:

Group A: Q.Wt=idf, Outer-P=1												
Case No.	Group Entry	Doc. Wt.	Term Set	P-Values for Ranges p1	p2	p3	p4	p5	p6	p7	p8	Aver. Prec.
1	A1	Binary	NL	1	1	1	1	1	1	1	1	.4863
2	A2	Tf*idf	NL	1	1	1	1	1	1	1	1	.5242
3	A3	Binary	BQ	1	1	1	1	1	1	1	1	.3549
4	A4	Tf*idf	BQ	1	1	1	1	1	1	1	1	.3767
Group B: D.Wt=tf*idf, Terms=NL												
Case No.	Group Entry	Query Wt.	Outer p	P-Values for Ranges p1	p2	p3	p4	p5	p6	p7	p8	Aver. Prec.
5	B1	Binary	1	1	1	1	1	1	1	1	1	.5218
6	B2	Idf	1	1	1	1	1	1	1	1	1	.5242
7	B3	Binary	1	2	1.7	1.5	1.3	1	1.3	1.5	1.7	.5193
8	B4	Idf	1	2	1.7	1.5	1.3	1	1.3	1.5	1.7	.5239
9	B5	Idf	2	2	2	1.5	1.3	1	1.3	1.5	2	.5104

(Continued on Next Page)

Table 4.4 continued: Medlars 8 Freq. Range Queries – Contrasts

Results for Trials (continued):

Group C: D.Wt=binary, Terms=NL											
Case No.	Group Entry	Query Wt.	Outer p	P-Values for Ranges p1	p2	p3	p4	p5	p6	p7	p8	Aver. Prec.
10	C1	Binary	1	1	1	1	1	1	1	1	1	.4928
11	C2	Idf	1	1	1	1	1	1	1	1	1	.4863
12	C3	Binary	1	2	1.7	1.5	1.3	1	1.3	1.5	1.7	.4839
13	C4	Idf	1	2	1.7	1.5	1.3	1	1.3	1.5	1.7	.4913
14	C5	Idf	2	2	2	1.5	1.3	1	1.3	1.5	2	.4788
Group D: D.Wt=tf*idf, Terms=BQ												
Case No.	Group Entry	Query Wt.	Outer p	P-Values for Ranges p1	p2	p3	p4	p5	p6	p7	p8	Aver. Prec.
15	D1	Binary	1	1	1	1	1	1	1	1	1	.3789
16	D2	Idf	1	1	1	1	1	1	1	1	1	.3767
17	D3	Binary	1	2	1.7	1.5	1.3	1	1.3	1.5	1.7	.3656
18	D4	Idf	1	2	1.7	1.5	1.3	1	1.3	1.5	1.7	.3644
19	D5	Idf	2	2	2	1.5	1.3	1	1.3	1.5	2	.3495

Ordering of Cases in Each Group:

A: $2 > 1 > 4 > 3$

B: $2 > 4 > 1 > 3 > 5$

C: $1 > 4 > 2 > 3 > 5$

D: $1 > 2 > 3 > 4 > 5$

---

the cases of interest are 9, 11, 13, and 15 of Table 4.2; 4 of Table 4.3; and 2 (or equivalently 6) of Table 4.4. Summarizing the ranges chosen and resulting average precision. Table 4.5 shows how the various options compare.

Table 4.5: Comparison of Effects of Different Ranges
When Doc.Wt.=Tf*idf, Query&Clause wt.=Idf,
and Terms=NL, $p = 1$

Original Table	Original Case	No. of Ranges	Limits of Ranges	Average Precision
4.5	9	4	1,3,21,90	.5055
	11	4	1,2,20,100	.5170
	13	10	1,2,3,5,9,17,33,65,129,257	.5164
	15	9	1,2,4,8,16,32,64,128,256	.5071
4.6	4	5	1,3,8,20,100	.5025
4.7	2	8	1,2,3,6,10,20,50,100	.5242

The entire range of variation of average precision values is only 0.5055 to .5242, so it seems unlikely that there is much significance of difference between one range assignment method and the next, as long as they are each fairly systematic. It does seem wise to have a separate class for frequency 2 terms but little else conclusive can be said about this data.

Fourth, consider the assignment of p-values to the outer AND operator. Using an outer p-value of 2 was tried in a number of cases, to contrast with the usual choice of AND^1 as the outermost connective; in all but one situation $p = 1$ is the best choice. Only when binary document weights are also used (i.e., case 5 of Table 4.3 which contrasts with the $p = 1$ case 6), and another confounding change in inner p-values was also introduced, is there an exception, caused by the interaction of these three parameter settings. However, since binary document weight is not encouraged, the exceptional situation is not really germane to the present discussion, and the conclusion stated above can be generally followed.

Regarding p-value assignments on interior clauses, the question is whether $p = 1$ should be applied throughout or whether the p-values should be high for the extremes of

frequency, and low for medium frequencies. Unfortunately, there is no clear answer; in a few cases the best results occurred when p-values varied while in the other cases using $p = 1$ was best or there was no significant difference. It is possible that only certain frequency ranges along with particular p-value settings give consistently better results and so should be utilized, but there is no algorithm apparent here for identifying such situations a priori.

Sixth, the question is whether query weights are advisable. Once again, the contrasts give conflicting or unclear results. As shown in Chapter 3, it is far more important if document term weights are used.

Finally, the question is whether $p = \infty$ gives tolerable results when the frequency range method is adopted. The abysmal performance shown in cases 1-8 of Table 4.2 should clearly dispel that illusion.

To put these comments on the results shown in perspective, a summary of the conclusions based on these exploratory tests is given in the next section.

4.3.2.4. Conclusions

The frequency range approach to automate query construction is a viable technique, especially when weighted terms and p-values can be employed. Based on experimentation with the Medlars collection, tentative settings for the basic parameters have been identified and could probably be used for other collections. Though the best selection of frequency ranges is not obvious for Medlars, any reasonable frequency theory based partitioning seems to give fairly good performance. The follow up study reported in [Salton, Buckley & Fox 1983] further verifies these two observations through testing of other ranges on both the

Medlars and INSPEC collections.

The use of varying p-values on different clauses, where the value of p is determined by the frequency range covered by a clause needs further study. Though certain assignments yield better results than the uniform use of $p = 1$ everywhere, further tests would be needed to pinpoint the appropriate combinations in all cases.

The use of weighted document terms is clearly indicated whereas weighting of query terms does not give consistent improvement. However, it is clear that using $p = 1$ on the outer connector (i.e., the AND which combines the clauses into a single query) is preferable.

Finally, the evidence for Medlars is that using the natural language query terms is far superior to using the Boolean query terms. The important point is that the auto frequency method relies heavily upon the fact that the set of terms provided be reasonably specific.

4.3.3. Frequency Range Summary

Boolean or p-norm queries can be automatically constructed from a set of terms based upon the theory of term discrimination values and a correspondence of those values with term frequencies. Initial experiments with the ADI two clause queries showed the technique to be a viable one. Terms were grouped into classes depending on their postings values, and a number of the runs gave performance improvements beyond both the original Boolean and the automatic vector methods.

Further studies were made using the Medlars collection. Once again, results were better than the original Boolean query, when $p < \infty$, but here they did not surpass vector methods. Relative contrasts demonstrated that a reasonable setting of parameters was to have all p-values $=1$, to weight document terms, and to make a good selection of terms (i.e., NL not BL query).

A follow up study is described in [Salton, Buckley & Fox 1983]. Additional tests were made for the Medlars and INSPEC collections, with similar results to earlier trials. The main additional conclusion was that using a small number of classes (e.g., 3-5) seems best.

All in all, then, the frequency range method seems viable and could be used in Boolean systems that have been extended to utilize document weights and low p-values. If vector methods are not implemented, automatically constructed p-norm queries could result in almost the same level of performance. Alternatively, a user of a Boolean system might call for automatic query construction and improve the result either via feedback techniques (as shown in Chapter 5) or by manual reformulation.

The auto-frequency method does have some disadvantages, however. First, since it utilizes $p = 1$ and weights, it is not designed for use in a Boolean query environment where $p=\infty$ and binary weights are required. Indeed, the performance under those conditions is poor.

Secondly, the method has many parameters and they are difficult to tune except through experimental processing; that makes use of the method somewhat troublesome. For example, determining the number and composition of frequency classes is still not thoroughly understood.

Finally, it is difficult to control, for the $p=\infty$ case, the number of documents that will be retrieved. Since the clause definition scheme is a static one based on fixed frequency ranges, there is no adjustment on how many query terms will be in a given clause. In the environment of Boolean systems with binary weights, this lack of control could be somewhat awkward. When weights are allowed, however, ranking and thresholding solve the control problem.

In part to overcome the limitations of the frequency range method, another automatic query construction technique, creating queries in disjunctive normal form (DNF), was developed. The DNF method provides control over the desired number retrieved and works well in a pure Boolean environment even with binary weights. This procedure is described in detail in the next section.

4.4. Disjunctive Normal Form Automatic Queries

The previous section focused on queries formed automatically using the frequency range approach, where typically an outer AND connective relates a number of clauses, each containing terms with similar frequencies. Though the recommended use of $p = 1$ on the outer operator, for p-norm interpretations, weakens the argument that strict AND is appropriate, the general query form is nonetheless close to conjunctive normal form. That is to say, an outer AND relates clauses which usually are made up of terms in the same frequency range, connected by OR, except when the frequency range is a high one (and AND is used inside a clause too).

When $p = \infty$, the frequency range approach gives poor performance, since the outer AND is strictly enforced, and since it is unusual for most of the relevant documents to match with all of the clauses identified. When $p = 1$, the similarity measure reflects the number of clauses which have matches. This has the unfortunate consequence that if there are many matches which occur in the first clause, of low frequency terms, which is enclosed by say OR^2, and few matches elsewhere, then in spite of the high idf values associated with those low frequency matches, an improperly low similarity would be computed. Perhaps this explains why the simpler vector matching scheme, where no clauses are formed, seems often to do much better. Apparently, without the aid of either semantic or co-occurrence

data to help identify the proper makeup of each clause, it is unlikely that a Boolean query with AND as the outer operator could do even tolerably well.

Using OR as the outer operator comes to mind as the proper approach, especially if $p = \infty$ is required. But that still leaves the problem of selecting operators for the inner clauses. A similar scheme to that of the frequency range system might be appropriate, but since there is an outer OR, it seems pointless to group low frequency terms into clauses with another useless surrounding OR connective. Inner clauses, then, could be single terms or conjuncts of two, three, or more terms – resulting in a query that is one simple type of disjunctive normal form.

While the frequency range method placed an upper bound on the number of clauses, by making that a fixed parameter specified as part of the set of chosen classes, the DNF scheme allows any number of inner clauses. Since these are ORed together, each clause should only be included if a high probability exists that when that clause is satisfied, a relevant document has been identified. The detailed procedure described later for selecting clauses therefore attempts to utilize any available data to give the best results. As will be seen in the next chapter, then, the DNF method can easily be applied to situations where feedback information is available to improve upon a priori collection-wide probabilities. Furthermore, the expected number of documents retrieved by a query can be easily controlled with a moderate degree of accuracy – an advantage over the lack of control commonly taken for granted in strict Boolean systems.

The question then remains, however, about how inner clauses should be formed. The situation is somewhat simplified by the fact that NOT is rarely called for – though defined in the p-norm system the negation operator seems so rarely useful that consideration of its

automatic inclusion should best be ignored. Hence, with inner clauses necessarily specific, using the AND connective seems the proper approach there. For test collections of the size used at Cornell, it seems unwarranted to have any more than three terms ANDed together. Therefore, the approach described here will truly be disjunctive normal form, with interior clauses being either single terms, ANDed pairs, or ANDed triples.

This section, therefore, explains the theory and methods of automatic DNF query construction. In addition, experimental results are given of tests of this method, and comparisons are made with the performance achieved by other Boolean query formulation methods. The discussion supplied gives a firm foundation for understanding the more generalized situation of automatic DNF Boolean or p-norm feedback, which is the central theme of the next chapter.

4.4.1. Theory and Method

Constructing DNF queries of the form alluded to above, namely as the disjunction of conjuncts, seems fairly simple. However, whereas the frequency range approach directly specifies the final query structure once terms and their frequencies are given, the DNF approach requires more sophisticated techniques for selecting the appropriate set of inner clauses.

The one essential new user supplied single parameter is termed "desired retrieved"; it is a target estimate of the expected number of documents to be retrieved by the query. Some sophisticated users might particularly enjoy being able to specify such a value, allowing them to indicate whether the query should be narrow, medium, or broad in scope as measured by size of retrieved set. Other users could simply ignore such a parameter, and perhaps rely on the system to utilize a medium value as the default setting. Such a default

could easily be selected after some initial tests are made on each new collection.

Given "desired retrieved", DNF methods can construct the appropriate queries when given a list of concepts and their collection frequencies. The general steps are:

(1) Delete all concepts that are useless for retrieval – i.e., stop words or words with very high collection frequency values.

(2) Let n be the number of initial terms provided. Construct a list of all possible clauses, including:

n single terms –
$s_1, s_2, \ldots, s_n$

$p = n*(n-1)/2$ pairs –
$p_1, p_2, \ldots, p_p$

$t = n*(n-1)*(n-2)/6$ triples –
$t_1, t_2, \ldots, t_t$.

(3) Use frequency information to assign a weight to each such clause, and to estimate the number retrieved by that clause.

(4) Select the best collection of singles, pairs, and triples such that the expected number retrieved by the whole query is roughly equal to "desired retrieved".

In performing these steps, certain assumptions make the task much simpler:

(1) Terms occur independently – i.e., P(A)*P(B)=P(A∩B).

(2) There is little overlap between the retrieved sets of each of the possible clauses, except when one clause is "covered" by another. Such covering is carefully avoided – i.e., one should not include clause c_i (e.g., 2 or 3 terms), if in the Boolean logic formalism some other c_0 (e.g., 1 or 2 terms) is present, and $c_1 \Rightarrow c_0$. For example, it is pointless to

include "A AND B" when "A" is already a chosen clause.

In order to assign an estimate of the number retrieved by each clause, it is necessary to make the following calculations. Assume that

N is the collection size,
n_i is the number of documents containing term i,
n_{ij} estimates the number containing terms i and j, and
n_{ijk} estimates the number containing terms i, j, and k.

Following the guideline of retaining useful information from old queries, it was decided that the n values above should count the query as a document, and so all query terms have frequency of a least one. This simplification makes terms seem better than they really are, a possible disadvantage for large collections. In any case, the calculations follow from definitions above:

(1) Since terms occur independently,

$$P(s_i \cap s_j) = P(s_i) * P(s_j).$$

(2) Since frequency values for terms are given,

$$P(s_i) = \frac{n_i}{N}.$$

(3) In general, for clause c (either single, pair, or triple),

$$P(c) = \frac{n_c}{N}.$$

(4) Solving for n_c, the derivations and results are:

$$n_{ij} = N * P(s_i) * P(s_j)$$

$$= \frac{n_i * n_j}{N}$$

$$n_{ijk} = N * P(s_i) * P(s_j) * P(s_k)$$

$$= \frac{n_i * n_j * n_k}{N^2}.$$

In general, for the m terms of clause

$$s_1 \; AND \; s_2 \; AND \; \cdots \; AND \; s_m$$

the estimated document frequency for the clause is

$$= \frac{n_1 * n_2 * \cdots * n_m}{N^{m-1}}.$$

Thus, for each clause one can easily compute the expected number retrieved.

Given the above estimates for each clause, and assuming as stated above that there is little overlap, one can estimate the expected number retrieved by the complete query simply by adding together the expected retrieved for each component clause. Since all possible 1, 2, or 3 term clauses have been included except for those eliminated by the covering rule, the grand total will undoubtedly far exceed "desired retrieved". If not, the query is simply the disjunction of all the non-overlapping singles, pairs, or triples. But if the total is too large, methods must be devised to select an appropriate smaller set of such clauses.

Selecting a subset of the possible clauses is most easily done if each clause is assigned some value reflecting its quality to retrieve relevant documents. Lacking feedback information, then, one must utilize frequency values and the most likely value to choose is the inverse document frequency (idf). In actuality, a formula monotonic with idf was adopted, one with the added quality that any available feedback information could be entered into the computation as well. Described in [Porter 1982], the formula assigns weight to term s_i

as

$$w_i = P(s_i \mid \text{relevance}) - P(s_i).$$

Since feedback information is lacking, one can fix

$$1 = P(s_i \mid \text{relevance})$$

and get consistent results. The final formula,

$$w_i = 1 - P(s_i) = 1 - \frac{n_i}{N},$$

is clearly monotonic with idf. For pairs and triples, the weights are

$$w_{ij} = 1 - n_i * n_j / N^2$$
$$w_{ijk} = 1 - n_i * n_j * n_k / N^3.$$

The only missing detail still not specified in the above algorithm outline is how to select an appropriate subset of clauses. Two different procedures have been suggested, and results are given for both later on. Hence, explanation of each is in order. These procedures are referred to as "sort" or "narrow" in the following discussion.

4.4.1.1. Sort

The original procedure was the "sort" one. Several versions were tried, with slight variations in settings of particular parameters, but there were only minor differences among results. The general idea and some of the variations should therefore be explained.

First, all initial terms were screened so that stop words and very high frequency terms were removed. Each remaining term n_i then had both $E_{s_i}(\text{retrieved})$[1] and w_i computed.

[1] $E()$ designates the expected value function.

The set of terms was sorted in descending order of w_i value and if there were too many left (i.e., more than "type_limit"=70), the worst ones were deleted. In any case, all with very low weight (i.e., $w_i < .1$) were removed from consideration.

Next, the remaining single terms were used to generate all possible distinct pairs. Those whose $E_{p_{ij}}$(retrieved) was too low (ex., $< .5$) were removed since it is unlikely that they would retrieve documents. Weights were computed for each pair, and if too many were left, ones with lowest weight were dropped.

Finally, using the above two sets of singles and pairs, all possible distinct triples were produced. Once again, those with lowest weights were eliminated to conform to a limit on the set size.

With singles, pairs, and triples listed, the entire group was sorted in decreasing order of clause weight. The query was then produced by simply adding in clauses from that sorted list, one by one, until the value "desired retrieved" for the query was suitably approximated by the total number expected by retrieval of the clauses added. Of course, as each clause was added, it was removed if covered by existing clauses, and if it covered clauses added earlier, they were removed.

Since the selection of clauses is based on the sort order of the list which gives weights for candidate clauses, it seems appropriate to call this the "sort" method. Sorting did not reorder entries unnecessarily so with many ties among the weights it is useful to note that the sorted list was added to in lexicographic ordering – i.e., single concepts are numbered alphabetically, and pairs or triples are added as

$$s_2 s_1,\ s_3 s_1,\ s_3 s_2,\ \ldots,\ s_n s_{n-1}$$

and

$\delta_1\delta_3\delta_2,\ \delta_1\delta_4\delta_2,\ \ldots,\ \delta_1\delta_n\delta_{n-1}\ldots,\ \delta_{n-2}\delta_n\delta_{n-1}$

so the "sort" method could be overly strongly influenced by the actual details of construction to give consistent and optimal results.

4.4.1.2. Narrow

Because of the sensitivity of the "sort" method to the details and special parameters of the construction algorithm, another method was devised and is reported upon in [Salton, Buckley & Fox 1983]. It will be referred to as "narrow" since it narrows down an initial query of all single terms ORed together until the estimated number retrieved is close to "desired retrieved". Some of the experimental results will be included in the summary charts and discussion of Section 4.5.

4.4.2. Sort Method Results

4.4.2.1. Medlars Collection

As in previous experiments, all parameters involved in the tests were identified, and suitable values for each selected. Document weights could be either binary or tf*idf. Query term and clause weights were assigned as both either binary or else based on idf values[2]. The third parameters, p-values, were consistently used on the outer and inner operators, and could be 1, 2, 5, 9, or ∞.

Regarding special DNF features, the hoped for number retrieved, called "desired retrieved", was set at one of: 20, 30, 50, 100, or 500. Note that in a collection of 1033 docu-

[2]Note that the weights computed for rating terms and clauses are appropriate to use as query weights. Indeed, since they are typically fairly close to 1.0, there should be less space distortion when using them than with simple normalized idf values.

ments it was thought that a goal of 500 would cause the algorithm to return almost all reasonable clauses – facilitating a check on the quality of the limiting method of obtaining a narrower query.

Regarding the clause makeup, two options were considered. One was to follow normal vector methods and only allow single terms. The other was to follow already described techniques for selecting singles, pairs, and triples. The notation adopted was thus to have "spt50" stand for "singles, pairs, and triples with desired retrieved=50" while "s500" essentially calls for all singles.

Tables 4.6 through 4.8 summarize the results of these initial Medlars tests. The first two tables illustrate effects of varying clause construction method and of p-value settings, respectively. Table 4.8 summarizes results for the best cases and for other trials that should be contrasted with.

The first part of Table 4.6 is concerned with behavior in the conventional retrieval environment, where $p = \infty$ and binary weights are employed. Since the DNF procedure was aimed specifically at this situation, it was hoped that behavior similar to that of manually constructed queries could be attained. Indeed, all but the control case of employing all single terms, that is "s500", were at least as good as the original manual query. Using "spt50" gave the best performance, a rather significant 47.5% improvement over the manual queries. Part (a) thus shows the success of the DNF procedure in this instance. It should be noted, however, that a key component in that success is the fact that the given NL terms were much better than the larger, less specific set of BL terms employed in the manual queries. In any case, NL terms is the sensible starting place for automatic query formulation, and the average precision values shown seem good enough to satisfy users

Table 4.6: Initial Medlars DNF Runs
Vary Clauses Using "Sort" on NL Terms

a) Strict Boolean Queries with Binary Weights

Case No.	Test Case Description	Aver. Prec.	%Improvement vs. Base
1	Base: Manual original strict Boolean query	.2065	-
2	spt20	.2111	2.2
3	spt30	.2506	21.4
4	spt50	.3046	47.5
5	spt100	.2719	31.7
6	s 30	.2705	31.0
7	s 500	.1599	-22.6

b) P-Norm Interpretation: $p = 1$, weights everywhere

Case No.	Test Case Description	Aver. Prec.	%Improvement vs. Base
1	Base: Manual original strict Boolean query	.2065	-
2	spt20	.4888	136.7
3	spt30	.5235	153.5
4	spt50	.5327	158.0
5	spt100	.5143	149.1
6	s 30	.2938	42.3
7	s 500	.5584	170.5

Table 4.7: Initial Medlars DNF Runs
Vary P-Values Using "Sort" on NL Terms

a) spt30 with various p-values

Case No.	Test Case Description	Aver. Prec.	%Improvement vs. Base
1	Base: Manual original strict Boolean query	.2065	-
2	$p = 1$	.5235	153.5
3	$p = 2$	.5073	145.7
4	$p = 5$	.4844	134.6
5	$p = 9$	.4639	124.7
6	$p = \infty$	.2506	21.4

b) spt50 with various p-values

Case No.	Test Case Description	Aver. Prec.	%Improvement vs. Base
1	Base: Manual original strict Boolean query	.2065	-
2	$p = 1$	.5327	158.0
3	$p = 2$	.5385	160.8
4	$p = 5$	.5090	146.5
5	$p = 9$	.4903	137.5
6	$p = \infty$	.3046	47.5

c) spt100 with various p-values

Case No.	Test Case Description	Aver. Prec.	%Improvement vs. Base
1	Base: Manual original strict Boolean query	.2065	-
2	$p = 1$	.5143	149.1
3	$p = 2$	.5109	147.5
4	$p = 5$	.4882	136.5
5	$p = 9$	.4735	129.3
6	$p = \infty$	.2719	31.7

Table 4.8: Initial Medlars DNF Runs
Compare Best Runs of "Sort" on NL Terms

Case No.	Test Case Description	Aver. Prec.	%Improvement vs.Base	vs. Cos
1	Base: Manual original strict Boolean query,bin.wts.	.2065	-	-62.3
2	Base: Vector using cos sim., tf*idf wts.	.5473	165.1	-
3	Manual query, $p = 2$, dwt=tf*idf,qwt-cwt=binary	.5573	169.9	1.8
4	DNF spt50, $p = \infty$, binary wts.	.3046	47.5	-44.3
5	DNF spt50, $p = 1$, idf based wts.	.5327	158.0	-2.7
6	DNF s500, $p = 1$, idf based wts.	.5584	170.5	2.0
7	upper bound retrospective, indep. assumption	.7675	271.7	40.2

when obtained as the first phase of an automatic feedback procedure (see more discussion in the next chapter).

A further conclusion obtained from Table 4.6a is that a medium value of "desired retrieved", in this case 50, is best. Indeed, performance follows the ranking:

spt50 > spt100 > s30 > spt30 > spt20 > s500.

This ranking differs from that based on results of part (b):

s500 > spt50 > spt30 > spt100 > spt20 > s30

which is based on weighted p-norm retrieval methods.

When p-norm methods are used, there are conflicting effects of query structure and utilization of good weights. With almost all singles and other possible clauses included, each with fairly good weights assigned, "s500" is like an extended vector technique and so does very well. Almost as good is "spt50", which has good specificity caused by selecting a

smaller number of clauses (or having longer clauses), but which also benefits from the choice of good term and clause weights. Other methods are similar – only "s30" where very few clauses are present shows especially bad performance. In any case, the good p-norm DNF queries are roughly 135-170% better than the initial manual, strict Boolean query.

Table 4.7 illustrates the effect of p-value on precision, for the three moderate cases spt30, spt50, and spt100 where weights are employed. In spt30 and spt100 cases, $p = 1$ again gives best results while for spt50, using $p = 2$ is more effective. It is always the case that higher p-values, namely $p = 5$, 9, and especially $p = \infty$, give continuously decreasing performance. Thus, while it is theoretically pleasing that the best case of all is spt50 with $p = 2$, the practical conclusion must be that for simpler implementation and best performance in most cases, that $p = 1$ should be employed.

Table 4.8 gives contrasts suitable to gauge the above remarks in the context of other methods. Considering cases 1 and 4 one sees that if strict Boolean logic with binary weights are used, DNF with spt50 outdoes manual queries. When p-values and weights are used, the manual queries with $p = 2$ and DNF s500 or spt50 are all about the same and closely match normal vector methods given in case 2. Further improvement is still achievable, as shown by the upper bound case 7.

4.4.2.2. INSPEC Collection

Based on the results of the Medlars test described above, similar trials were made with the larger INSPEC collection. The small number of trials made are reported in Table 4.9, and are sufficient to illustrate basic trends.

First, it should be noted that in general, average precision is rather low. This is due in part to the size of the collection, the poor quality of the queries, and finally to the

Table 4.9: Initial INSPEC DNF Runs
Compare Clauses,Terms for "Sort" with Binary Wts.,$P = \infty$

Case No.	Test Case Description	Terms Used	Aver. Prec.	%Improv. vs. Base
1	Base: Manual original	BL	.1159	-
2	spt30	BL	.0890	-23.3
3	spt30	NL	.0315	-72.8
4	spt50	BL	.1037	-10.6
5	spt50	NL	.0335	-71.1
6	spt100	BL	.1082	-6.7
7	s500	NL	.0333	-71.3

restriction to only use conventional Boolean retrieval methods.

Second, it should be noted that BL terms are better than NL terms. Indeed, except in a few cases where searchers did an exceptionally bad job of query construction, the Boolean queries selected out the main useful terms from the long, vague, paragraph form query statements. Since DNF methods rely so heavily upon proper utilization of the initial set of query terms, it is sensible that for the INSPEC tests, BL terms are advised.

A final observation concerns the best clause selection method. "spt100" is slightly better than "spt50" which is again better than "spt30", for the BL term set. Indeed, even in the very large INSPEC collection, if a good set of initial terms is utilized (i.e., BL terms), a reasonable DNF technique like "spt50" or 100 does almost as well as manually formulated Boolean queries.

These key tests using strict Boolean methods on INSPEC show that with a good set of initial terms, DNF techniques can work reasonably well.

4.5. Summary and Conclusions

Tables 4.10 and 4.11 summarize the best cases of each type of query formulation run made for Medlars and INSPEC collections, respectively. Table 4.10 includes a few more tests made in an early stage of investigating automatic query construction, but otherwise the content and format of the two charts are identical. The DNF "narrow" results are from [Salton, Buckley & Fox 1983] and are included for comparison purposes.

Table 4.10: (Medlars 1033 Docs., 30 Queries Collection)
Overall Summary of Initial Search Results

Case No.	Case Description	Aver. Prec.	% Difference vs.Boolean	vs. Cos
1	Base: Manual strict Boolean	.2065	-	-62.3
2	Base: Vector cosine	.5473	165.1	-
3	Manual query;$p = 2$;weighted	.5573	169.9	1.8
4	Best initial freq. range; 8 ranges;$p = 1$; NL terms; cutoffs:2,3,6,10,20,50,100	.5170	150.4	-5.5
5	Best strict binary initial freq.range; NL terms; 5 ranges; cutoffs:10,20,50,100	.1904	-7.8	-65.2
6	Best later wt'd freq.range; 3 ranges; $p = 1.5,1,1.5$; NL terms; cutoffs:1,2,5	.5229	153.2	-4.5
7	Best later freq.range; strict binary; 3,5,or 8 ranges; NL terms; bin.wts.	.0371	-82.0	-93.2
8	DNF "sort";spt50,$p = \infty$;bin.wts.	.3046	47.5	-44.3
9	DNF "sort";spt50,$p = 1$;wt'd	.5327	158.0	-2.7
10	DNF "sort";s500,$p = 1$;wt'd	.5584	170.5	2.0
11	DNF "narrow",$p = \infty$,bin.wts.,spt50	.2899	40.4	-47.0
12	DNF "narrow",$p = \infty$,bin.wts.,s100	.3009	45.7	-45.0
13	DNF "narrow",$p = 1$,wt'd,spt100	.5597	171.1	2.3
14	DNF "narrow",$p = 1$,wt'd,s500	.5550	168.8	1.4
15	Upper bound retrospective;term relev.	.7074	242.6	29.3

Table 4.11: (INSPEC 12684 Docs., 77 Queries Collection)
Overall Summary of Initial Search Results

Case No.	Case Description	Aver. Prec.	% Difference vs.Boolean	vs. Cos
1	Base: Manual strict Boolean	.1159	-	-50.2
2	Base: Vector cosine	.2325	100.6	-
3	Manual query;$p = 1$; terms: doc=wt'd,query=bin.	.2747	137.0	18.2
4	DNF "narrow",$p = \infty$,bin.wts.,spt50 NL terms	.0503	-56.6	-78.4
5	DNF "narrow",$p = \infty$,bin.wts.,s30 NL terms	.0547	-52.8	-76.5
6	DNF "narrow",$p = 5$,wt'd,spt50 NL terms	.1567	35.1	-32.6
7	DNF "narrow",$p = 1$,wt'd,s500 NL terms	.1110	-4.2	-52.2
8	Freq. range; 5 ranges;$p = 1$; weighted; cutoffs:2,3,4,7,21	..1592	37.3	-31.5
9	Upper bound retrospect.;term relev.	.2909	151.0	25.1

The first lines of each chart are for the base cases while the last case serves as an upper bound limiting possible effectiveness when linear weighting schemes are employed. Two goals were aimed at in suggesting use of automatic query methods: devising strict Boolean ($p = \infty$,binary weights) methods comparable in effectiveness to manually formed queries, and devising weighted schemes that do better than cosine vector methods.

Cases 8, 12, and 13 of Table 4.10 show that for Medlars, the first goal can be achieved; unfortunately, possibly because the initial NL terms submitted for INSPEC were so numerous and of such low specificity, no similar cases are found for that larger collection. For Medlars, it is convincing to see that three different runs using disjunctive normal form

queries all achieved performance exceeding that of manual queries by over 40%. In particular, the "sort" and "narrow" DNF algorithms, for the average case of having singles, pairs, and triples selected in order to retrieve 50 documents, give fairly good results. Even when the "sort" form for locating 100 documents by a query made up only of single terms is employed the performance is still satisfactory.

When weights are employed, all but one poorly chosen INSPEC case (number 7, where using "desired retrieved" of 500 is too low) do better than the strict Boolean manual queries. This is quite a success, since it seems likely that a conventional Boolean retrieval system can be adapted to perform the appropriate operations. Thus it is hoped that users need not have to always devise a manual Boolean query in order to begin doing feedback searching.

For Medlars, some of the weighted cases are slightly better than the cosine run but that is not the case with INSPEC. Thus, due to the relative simplicity of vector methods, if one had to devise a system to automatically process a user's NL query and conduct a search, it is clear that vector methods are the obvious choice. There is the potential, as in Medlars, that if good terms are initially provided, the automatic p-norm query construction methods can do better than vector techniques, but additional testing is needed to truly decide in what circumstances such techniques should be employed.

Aside from the above practical comments there are a number of very interesting observations that can be based on the available data. Automatic query techniques are useful in certain cases - the DNF form for either weighted or unweighted situations and the frequency range approach for weighted runs.

Frequency range techniques are almost as good as vector cases, which makes sense since: all terms are used, $p = 1$ is encouraged, and weights are similarly derived. When clauses are structured in a better fashion – i.e., when manual queries are interpreted with $p = 1$ and document weights, there is an improvement in both collections, even beyond that of cosine vectors. Thus, clause structure based on frequency theory is not quite as good as when based on semantics, but is not much worse when compared to a flat clausal vector form.

DNF techniques are needed if one hopes to use any of these automatic approachs to strict Boolean query formulation. There is not a great deal of difference between the "sort" and "narrow" methods but the later is be preferred because it is less sensitive to wide variations in performance due to slight parameter changes. In conclusion, the DNF scheme seems the most sensible approach to adopt for doing automatic query construction – whether of weighted or unweighted collections. Further tests using it for automatic feedback query construction are therefore given in the next chapter.

CHAPTER 5

AUTOMATIC BOOLEAN AND P-NORM FEEDBACK

5.1. Introduction

The principle of feeding back information from a previous attempt in order to accomplish a more effective subsequent trial, introduced to problems of automatic control by such pioneers as Norbert Weiner [1965], has been applied in many forms to various problems. One of the most intriguing and successful ideas advanced by researchers investigating automatic retrieval systems is that of using feedback information to improve the effectiveness of searches.

For retrieval, Rocchio [1966, 1971] did the first significant work, demonstrating that with the vector space model, feedback could improve performance by moving a new, automatically formed query from its original location in the space to another, more appropriate place, closer to relevant documents and further from non-relevant ones. The only user involvement in this entire process is that of simply inspecting a small list of say 10 initially retrieved documents, deciding which are relevant, and asking the system to form a new search taking that information into account.

Since the time of Rocchio's early work, a number of other researchers have adopted this basic technique, applied it to different situations, and devised theories and term weighting schemes that give ever increasing performance improvements. Their work, especially that relating to the use of feedback in Boolean systems, is discussed in Section 5.2 below.

Section 5.3 focuses on a new method for using feedback information, either in conventional Boolean contexts or in the extended or p-norm system elaborated on in previous chapters. Included are some early notions, the basic techniques, and comments on the method of evaluation.

Section 5.4 briefly reports on experiments conducted using the Medlars collection, showing what parameter settings are appropriate and what results from each of the several changes in variables that are made. Consistent and substantial effectiveness gains are clearly demonstrated for the Medlars tests.

Finally, Section 5.5 summarizes the methods and results specified, suggesting practical uses for these feedback algorithms and mentioning possible further investigations that might lead to additional performance enhancements.

5.2. Previous Research

After Rocchio's basic works [1966, 1971], a number of others extended the technique, developed suitable evaluation methods, and conducted tests with various collections of documents (e.g., [Ide 1971], [Ide & Salton 1971], [Chang, Cirillo, Razon 1971], [Salton 1971b, 1971d], [Salton 1975]). For additional information see discussion below and in Chapter 8.

5.2.1. Mixing Vectors and Boolean Queries

An interesting preliminary exploration of feedback in the context of vector and Boolean queries is [Fisher & Siegel 1972]. The approach was to have both a vector query $\vec{V}$ and a Boolean query B, and use the two together for retrieval and ranking. This predecessor of the p-norm technique was a bit cumbersome but some interesting ideas were suggested. The modification of the vector portion $\vec{V}$ was performed using SMART techniques,

following ideas of Rocchio, Salton, and Ide. Modifying B was harder to formalize and is more of interest here. Some of the suggestions on how to modify the use of a concept in the Boolean expression are:

(1) If a concept is in an AND clause, but not all relevant documents contain it, change to OR: to increase recall.

(2) Conversely, if a concept is in an OR clause and all relevant documents contain it, use AND: to increase precision.

(3) For a concept not in B but appearing in retrieved documents, add it, using AND if it occurs in all relevant documents, using AND NOT if it occurs in many non-relevant but no relevant documents, or using OR NOT if the concept appears in many non-relevant but only a few relevant documents.

Though somewhat ad hoc, a procedure following these suggestions seemed to result in queries with better performance than that of the original queries. An interesting side note was the observation made then, one which has been proved and dealt with by p-norm tests and methods, that "the choice of an AND relationship over an OR should not be made lightly, since the AND operand is very restrictive."

5.2.2. Feedback with Single Terms Only

In March 1972, Barker, Veal, and Wyatt [1972] described semi-automatic methods aimed at improving the Boolean queries stored as search profiles for users of the United Kingdom Chemical Information Service (UKSIS). After analyzing 31 profiles, obvious defects were identified in about 40% of the cases. Through an examination of relevant items, the individual terms present in the query and documents could be evaluated as to specificity and a revised query developed using only terms of proper specificity.

Unfortunately, no method was proposed for evaluating clauses made of pairs or triples of terms.

A group of users were asked to construct the original 31 profiles without the aid of intermediaries, thus mirroring the desired situation of having a system destined for casual access by many. Of those profiles, 61% had an inadequate initial formulation. 32% were overly restrictive and 19% had bad term fragmentation – both of these weaknesses could be ameliorated by using p-norm forms. 13% did not expand concepts adequately, something that could be aided by using lexical relations or feedback. 10% had spelling errors, which nowadays could be automatically identified.

Other sets of profiles were examined to identify the causes of failures in retrieval due to factors affecting precision (e.g., terms in wrong context – 68%, term fragmentation – 61%, homonyms – 13%) or factors affecting recall (e.g., inadequate expansion – 76%, over-restrictiveness – 35%, NOT terms – 15%). Dealing with these failures, while a meaningful task for a search intermediary, is especially difficult for an automatic procedure. Hence, the actual experimental methods described by Barker et al. simply constructed queries as the disjunction of single terms, each of proper specificity.

To measure term specificity, feedback data was used in the simple formula

$$\text{specificity} = \frac{\text{number of relevant documents in which word appears}}{\text{total number of documents in which word appears}}. \qquad (5\text{-}1)$$

After an initial search, these automatic feedback queries were formed, searching was done again, and the process continued until no new relevant documents appeared. Eventually a complete list giving term specificity for original and feedback terms was presented to the user to aid in constructing a final manual profile.

New profiles, created using the above single term feedback system, were then compared with original ones. Generally, precision remained fairly level or showed a slight increase, but there was a consistent recall improvement of a noticeable degree. It was hoped that if term pairs could be rated as well, then even greater improvements would result.

5.2.3. Automatic Query Adjustment

The vector methods of incorporating feedback information follow the simple strategy of changing the set of terms in the vector and/or of revising weights on terms. For Boolean queries, the task, in its most general formulation, is the very hard one of constructing the best possible Boolean expression for retrieval using the original query, appropriate terms, and all available collection and feedback information as input to a suitable algorithm.

A related but simpler problem is that of efficiently implementing a Boolean search, namely of constructing a tree that specifies when and where to do each of the various merge operations. The work of Liu [1976] illustrates the complexity of this process; she points out that "the problem of finding optimal merge trees that realize truth functions determined by Boolean expressions in which not all variables are distinct is a difficult one" [Liu 1976 page 314]. Forming a new Boolean query, given an initial manually or automatically formed one plus information supplied by relevance feedback, is certainly a great deal more difficult. As such, it suggests, instead of formal methods like Liu's, appealing to sensibly based but otherwise ad hoc techniques, like the following.

Carlo Vernimb, working with the European Nuclear Documentation System (ENDS), reported upon one such methodology [Vernimb 1977]. At the time of article submission, the procedure was already in routine use for several years.

The approach followed is a rather curious one, employing many heuristics for query construction and following a number of ad hoc rules for stopping various stages of the process. Doubtless there would be many exceptions where the procedure would give poor results.

Nevertheless, it is interesting to see how some of the formulation problems were faced. First, the basic algorithm really used available feedback information twice in a complete search cycle rather than just once. Specifically, a search cycle had six steps:

(1) Construct partial queries using relevance information.

(2) Combine query parts.

(3) Search and get feedback information (user interaction 1).

(4) Select effective partial queries.

(5) Form a new loosened query.

(6) Search and get feedback information (user interaction 2).

Steps 1-3 aim at improving precision while 4-6 focus on recall. Clearly, performance of this approach could be somewhat oscillatory, and, because of these and other complications, hard to evaluate or compare with other methods.

Step one was accomplished after forming a matrix of term occurrences in relevant documents. A clause was formed for each relevant document as the conjunction of all terms occurring in that document, except for terms that occurred in too few relevant documents. For example, one clause for a sample query "high temperature thermocouples" was

thermocouples *AND* wires *AND* high temperature. (5-2)

Step two simply combined all clauses (called partial queries) devised in step one, using the

OR connector between clauses.

After the search and feedback of step three, step four computes a quality factor for the query as a whole and for each of its clauses, and discards clauses less productive than the old query. The quality factor is relatively crude, i.e.

$$q = \frac{H}{N} \tag{5-3}$$

where H = number relevant retrieved

N = number irrelevant retrieved.

Ostensibly, steps 4-6 improve recall; however, step 4 obviously must **reduce** recall. Hence it is step 5 that aims at improving recall. According to an ad hoc set of rules of how to "loosen" various common Boolean sub-expressions, the remaining query clauses are loosened. Thus, the form (5-2) becomes

thermocouples *AND* high temperature. (5-4)

It is interesting to note that each of nine documentalists asked to construct a query came up with form (5-4). Successive steps of processing this example result in queries or parts as follows:

(1) thermocouples

(2) thermocouples AND (rhenium alloys OR tungsten alloys OR tungsten)

One key comment about this procedure concerns stopping rules. In different steps of the algorithm, retrieval proceeds until stopping is dictated by a particular condition such as: at least 10 retrieved, or some number (say x) of irrelevant documents retrieved in a row. In most retrieval systems, one must retrieve all documents identified by any query submit-

ted, so a fair evaluation, as distinct from the one described in the article, could not blatantly ignore documents retrieved but thrown away.

In terms of evaluation, Vernimb did claim this automatic, self-stopping, iterative method performed better than manual formulation schemes. He reported a ratio, comparing the number of relevant documents retrieved by the automatic method to the number retrieved by the initial manually constructed query. For easy, medium, and hard searches, these ratios were: 1.08, 3.69, and 13.33, respectively. Apparently, people do worse with more difficult search problems, and so in relative terms the automatic algorithm does better.

Vernimb's method showed the potential of Boolean feedback; it also pointed out the difficulty of devising a simple, consistent, successful approach to the task.

5.2.4. Improving the Set of Terms Selected

As mentioned earlier, Vernimb pointed out that a key component of the query formulation problem is that of selecting the proper terms to include in the query. Obviously, in a feedback situation, one could consider both terms in the original query and terms in all retrieved documents. Since the work of forming a query must increase with the number of terms that are candidates for inclusion in that query, a key question is whether any terms in the above mentioned set can be excluded. An obvious simplification is to exclude all terms that occur only in non-relevant retrieved documents. Furthermore, one can assign term weights or consider other characteristics of the remaining terms, and eliminate even more.

An interesting approach to Boolean feedback query formulation, employing sophisticated methods for selecting the proper terms to consider, is that tested as part of the

Responsa research project [Attar & Fraenkel 1977, 1981]. Though an essential technique employed in that project, using metrical constraints (ex., two terms must occur within n words of each other) to give greater precision than the normal AND operator, is not available in many Boolean logic systems, their general approach to the term selection problem is nevertheless of interest.

For each term s in the original query, a ranked list of related terms,

$$R(s) = (x_1, ..., x_n),$$

is constructed, where the ordering of the x_i terms is in descending order of the values $b(x_i)$, which are themselves computed by one of various methods. In usefulness, the list $R(s)$ is similar to a set of lexically related terms (recall Chapter 3), but while the lexically chosen set was unranked and based on general linguistic information, subsets of this Responsa "reference vector" can be easily selected. Thus, one may choose the k best ones, namely x_1 through x_k, or only those suitably good, namely x_1 through x_j, where $b(x_i) \geq T$ for $1 \leq i \leq j$ and for a specified threshold value T. Note that the reference vector is based upon behavior of terms only in the "local" set of retrieved documents. For each original term s, the set $R(s)$ can be presented to a user for inclusion in subsequent manual query reformulations; alternatively, automatic methods can be employed, say to replace term s with the disjunction of it and a suitable number of related terms.

In the context of the Responsa project, these techniques are especially useful, since in the Hebrew language collections of interest there are an enormous number of morphologically or lexically related words for each original word or word stem. By employing the reference vector approach, a dramatic reduction in the number of related terms can be effected and resulting Boolean queries can have proper specificity.

The key contribution of the Responsa project in the context of Boolean retrieval is the concept of reference vectors and the methods proposed to formulate such vectors. However, metrical information is not provided in many Boolean logic systems, and since such data has therefore been excluded from the experimental work being described in this thesis, further discussion of the metrical approach to reference vector construction is beyond the scope of this work.

Regarding automatic utilization of selected terms in the feedback query, little exciting was discovered by the Responsa work. Manual reformulations using reference vectors did give improvements, but the simplistic automatic reformulation method employed typically led to performance degradation instead.

Hence, it seems appropriate to focus next on the findings of a work that more adequately treats the query construction process. An initial study by Dillon and Desper and subsequent derivative work, all described in the next section, are thus particularly germane to solving the difficult problem left unaddressed by Responsa, namely that of automatically formulating a proper Boolean query from a given set of terms.

5.2.5. Prevalence Weights and Threshold Based Clauses

Dillon and Desper [1980] proposed a theoretically pleasing Boolean feedback methodology, though like the work of Vernimb, it too was plagued by methodological and conceptual errors, problems with ad hoc formulas, and arbitrary cutoffs. Unfortunately, they adopted somewhat unusual evaluation measures, and failed to provide experimental results that were at all convincing. Nevertheless, their scheme did serve as catalyst for development of algorithms described later in this chapter. Problems and characteristics of their method are as listed below.

First, they erroneously assumed that feedback information alone was adequate for constructing new queries. That is, they ignored the original user-supplied query, its terms and formulation, in all subsequent processing, and also did not consider term postings or other collection wide data.

Second, they computed prevalence weights for each term, using formula

$$\text{positive prevalence} = \frac{\text{no. of relevant documents with term}}{\text{no. of relevant documents retrieved}} \tag{5-5}$$

which is the same as (5-1), and

$$\text{negative prevalence} = \frac{\text{no. of non-relevant documents with term}}{\text{no. of non-relevant documents retrieved}} \tag{5-6}$$

which are present in the final combined measure

$$\text{prevalence} = \alpha \cdot \text{pos.-prev.} + \beta \cdot \text{neg.-prev.} \tag{5-7}$$

Though aware of term relevance and other schemes for assigning weights to terms, they nonetheless employed the ad hoc and problematic weighting scheme expressed in (5-7), with parameters α and β arbitrarily set to 1.0. They admitted that there were a number of problems with (5-7) but for some reason used it nonetheless.

Third, following the work of others such as Barker et al. [1972], they only evaluated single terms and failed to consider actual information about pairs or other groupings of terms. However, they did consider how to combine terms into clauses, using the above mentioned prevalence weights, and that methodology is the backbone of their approach.

Their notion, similar in certain ways to the idea behind the frequency range method discussed in Chapter 4, was to combine certain sets of terms using the AND connective and other sets of terms using the OR operator. Instead of using frequency information, how-

ever, they used prevalence values. Like the frequency range approach, they used cutoffs to group terms, so that all terms in a given prevalence range would be treated similarly. However, the only suggestion for determination of cutoffs was to try various values out empirically. Also, the treatment of terms in each group was ad hoc and problematic.

Terms with high prevalence values were ORed together, a sensible first step. Terms with moderate prevalence values were grouped in pairs, ANDed together, and all such possible pairs ORed together. Lower prevalence values would lead to triples, and so on; experiments, however, did not consider triples or other more numerous sets. Rather, negative prevalence values were treated in a negative but symmetrical fashion to that of positive prevalence.

Very low prevalence weights led to single terms ORed together; the whole clause then was prefixed with the NOT unary operator since documents with those terms were to be avoided. Prevalence weights that were not quite as negative led to a sequence of NOT clauses each made up of pairs of terms ANDed together. An example given in their initial report [Dillon & Desper 1980 page 202] is

((academic *OR* cataloguing) *OR* (5-8)
(acquisitions *AND* periodicals))

AND

((*NOT* (chronology *AND* interviews) *AND*
(*NOT* methods))

where

all positive prevalence clauses are connected by OR,
all negative prevalence clauses are connected by AND, and
all the positive and negative portions are linked by an outer AND.

Though elegant, this scheme had several problems beyond those mentioned.

The fourth difficulty with the proposed technique relates to the use of negative prevalence values and NOT clauses. Just as negative feedback was complicated to work with in a vector based scheme [Ide 1971], so also is it in a Boolean system. Semantically, use of NOT is problematic - a query "physics AND NOT chemistry" would not retrieve a document containing the statement "This article is about physics but not about chemistry"! While it might be wise to exclude most documents containing terms with very low prevalence, the clever notion of excluding documents with several fairly bad terms may or may not turn out to be useful. Indeed, the easiest and most reliable (in terms of recall) way to handle bad terms might be to simply eliminate them from the query altogether.

Fifth, the identification of cutoffs for the prevalence value ranges is problematic. Fixed limits could yield non-productive queries with few "positive" clauses and many negative ones. On the other hand, since no frequency information is considered, queries with many positive clauses and few negative clauses could retrieve far too many documents. Having floating cutoffs, so as to adjust a query form to the range actually present for that query, could erroneously lead to identical handling in cases where all terms are fairly good (ex., all terms have prevalence values in the range 0.8 to 1.0) or all are fairly bad (ex., in range 0.0 to 0.2).

A final difficulty with the reported results is that of evaluation. As Dillon and Desper admit, the number of queries and documents they employed is clearly inadequate to yield significant results. Further, the evaluation measures reported are unconventional and seem biased toward the recall measure; one might question the appropriateness of such a choice since the proposed query formulation method has so many recall-enhancing features (e.g.,

ORing in a number of single terms and pairs of terms).

Though the Dillon and Desper scheme had many problems, they considered it wise to continue the line of research and resolve the more blatant errors and inconsistencies. They suggested: using different term weighting methods, exploring better ways to manipulate cutoff values, ignoring NOT clauses, and testing with bigger collections.

A Cornell student, Adam Fleisher, conducted a brief study of the feasibility of some possible improvements to Dillon and Desper's work. Following general suggestions of Gerard Salton, Chris Buckley, and this author, Adam used more queries, tried several weighting schemes, and proposed other revisions to consider. Though results were inconclusive due to a number of problems, one useful suggestion was to use "tempered" term weights – i.e., multiply the relevance (here prevalence) weight by a collection-based merit factor, such as the idf value. That idea was later applied to the disjunctive normal form scheme discussed in subsequent sections of this chapter.

Thus, in spite of the numerous limitations and problems of the Dillon and Desper approach, they did prompt a re-opening of investigation of the important problem of finding effective methods for automatic Boolean feedback query construction. The next section addresses that problem in general form and then zeroes in on a straightforward solution extending the technique described in the last chapter.

5.3. Theory and Method

Based on previous studies, it is clear that the problem of automatic Boolean feedback relates to a number of facets of the retrieval process. Seven of these issues are considered.

5.3.1. Obtaining an Initial Query

5.3.1.1. Problem

In many cases a searcher will provide an initial Boolean formulation. When that is not available an automatically constructed query from a user-provided list of terms, using a method such as one of those described in Chapter 4, can be prepared. The question then is, which of these should be used for feedback tests and how exactly should they be obtained.

5.3.1.2. Solution

For experimental purposes both the manual and the automatic queries are used in separate families of trials. To maintain consistency, the automatic queries were composed of terms from user-supplied natural language queries and were constructed using DNF methods out of singles, pairs, and triples with desired number retrieved 50 (i.e., spt50).

5.3.2. Utilization of the Initial Query in Feedback

5.3.2.1. Problem

Given an initial query, Q_0, the issue is how it will appear in subsequent feedback queries Q_1, Q_2, ... Note that each query Q_i will locate or retrieve a set of documents L_i, of which some are relevant, R_i, and others are irrelevant, I_i. Let

$$Q_{i+1} = f(Q_i, L_i, R_i)$$

where function f employs the specified feedback information and other generally available collection information.

Focusing on the use of the previous query Q_i, it is clear that Q_{i+1} :

a) may explicitly contain Q_i -- e.g., $Q_{i+1} = Q_i \ OR \ Q_i'$

where "derived" query Q_i' is not a modified form of Q_i,
b) may contain a modified form of Q_i – e.g.,
a "loosened" form like that suggested by Vernimb,
c) may contain terms present in Q_i, or
d) may completely ignore Q_i.

Based on insights from vector feedback experiments [Wu & Salton 1981], the wisest course seems to be to possibly include a (down weighted) copy of Q_i and to also encourage selection of good terms from Q_i. Obvious choices to consider are, using p-norm notation,

$$Q_{i+1} = < Q_i, w_i > OR^p < Q_i', 1-w_i > \quad (5\text{-}9)$$

$$Q_{i+1} = < Q_i, w_i > AND^p < Q_i', 1-w_i > \quad (5\text{-}10)$$

$$Q_{i+1} = Q_i' \quad (5\text{-}11)$$

where the weights and p-values can be dropped if strict Boolean forms are required.

5.3.2.2. Solution

Two of these approaches were tested experimentally. One uses the derived feedback query alone as in (5-11). The other is that of (5-9), or in actuality, the simpler form $Q_{i+1} = Q_i \; OR \; Q_i'$. The first scheme presumes that a new query Q_i' includes all important components of Q_i, or that Q_i' has better recall or precision behavior than Q_i. The second scheme presumes that keeping Q_i will not degrade retrieval results unduly but that omitting it could be harmful, such as by reducing recall.

These two schemes should be contrasted to see which is better in various situations. The third scheme, shown in (5-10), was not tried since very low recall was expected. Other combinations of the initial and derived query were also not considered because doing so would greatly complicate the approach taken and lacked firm theoretical support.

5.3.3. Selection of Terms for Feedback Queries

5.3.3.1. Problem

As mentioned in Section 5.3.2, it seems wise to consider using terms in Q_i. Similarly, terms in R_i should be analyzed for inclusion in Q_{i+1}. Additionally, related terms from a maximum spanning tree [Van Rijsbergen 1979], lexically or semantically based thesaurus, or other source could be added. The simplest approach to selecting terms is to compute some weight w_j for each single term s_j and then utilize only terms with suitable w_j values. Suggested is computing

$$w_j^s = w_j^{sc} \cdot w_j^{sr}$$

where a single term's value is its relevance weight w_j^{sr} ("sr" for single, relevance) tempered by a measure of its value in the collection as a whole, w_j^{sc} ("sc" for single, collection). The actual weight selected could reflect idf values, any of various estimates of relevance weight [Robertson & Sparck Jones 1976], the EMIM weight suggested in [Harper & Van Rijsbergen 1978 page 198], or other computations.

5.3.3.2. Solution

A key distinction between the DNF method of Chapter 4 and the feedback method described here is the more general formula for term and clause weighting computations. That is to say, the automatic query formulation method can be viewed as a simplification of the feedback approach described below. Consequently, no changes in procedures are required if an initial query yields no relevant retrieved documents.

For a given single term s_i, the tempered relevance weight w_i is given by

$$w_i^s = w_i^{sc} \cdot w_i^{sr} \tag{5-12}$$

$$= \text{tempered relevance weight}$$

where a collection component, monotonic with idf,

$$w_i^{sc} = 1 - \frac{n_i}{N} \tag{5-13}$$

$$= \text{collection weight}$$

for

n_i = postings for s_i

N = collection size

is employed in identical form as that for DNF techniques and a term relevance form, derived by feedback data, is given by

$$w_i^{sr} = \frac{r_i}{R} - \frac{n_i}{N}$$

for

r_i = number of relevant retrieved docs. containing s_i

R = number of relevant retrieved documents.

Note that the important parameters can be easily understood by considering the following contingency table, giving numbers of documents in those categories of interest here.

Table 5.1: Feedback Contingency Chart

	Relevant Retrieved	Non-relevant Retrieved	(totals)
Including Term s_i	r_i	$n_i - r_i$	n_i
Lacking Term s_i	$R - r_i$	$N - R - n_i + r_i$	$N - n_i$
(totals)	R	$N - R$	N

It should be noted that when no feedback information is available, one might consider $r_i = R$, which lets w_i^{sr} reduce to w_i^{sc}. Thus, w_i^s becomes $(w_i^{sc})^2$, and, ranking by w_i^{sc}, as is done for DNF cases, is identical to using the above feedback formulation (5-12).

A few non-obvious but essential details relating to implementing the above scheme must still be explained. First, consider how to exclude very high frequency terms from the feedback query, where such terms are not present in many relevant retrieved documents. Essentially, a postings cutoff factor PC can be chosen so that terms with $r_i \leq 1$ and $\frac{n_i}{N} \geq PC$ are excluded. Typically, $PC = .10$ so terms appearing in no fewer than 10% of all documents will be omitted.

Second, there is the matter of defining what is meant by a retrieved or relevant retrieved document. If a query retrieves no documents, one might have $R = 0$, leading to undefined values in the above equations. Though various schemes, such as adding .5 to each value in the contingency table [Robertson & Sparck Jones 1976] have been used, they

can lead to problems [Porter 1982]. Furthermore, there should be some way to easily include query terms in the above considerations, and following the rules above, they would actually not be present unless also occurring in retrieved documents.

The "trick" that solves this awkward dilemma is to claim that the query itself is a document. Following the vector space approach, a query certainly can be manipulated like a document, so this assumption seems intuitively pleasing. [Croft 1980] mentions this idea, stating that "the query can be treated as a relevant document." Generalizing the idea further, one would like to allow that perhaps a query should be viewed as better than a document - perhaps it should count for two documents. This perspective is allowed by defining the "query count" parameter

$$qcount = \text{no. of relevant retrieved documents the query counts as} \qquad (5\text{-}14)$$

With this definition, one must be sure that all four essential values shown in Table 5.1 are suitably incremented by *qcount* for any term s_i included in the original query. Typically, $qcount = 1$, but one can emphasize the importance of query terms by setting $qcount = 2$, or by using even higher values.

5.3.4. Grouping of Terms into Elemental Clauses

5.3.4.1. Problem

While Dillon and Desper suggested using pairs or triples of terms in their feedback queries, they only determined weights on single terms. Using collection statistics one can estimate values w_j^{pc} for pairs and w_j^{tc} for triples. Feedback data readily provides relevance values w_j^{pr}, w_j^{tr}. Thus w_j^{p} and w_j^{t} can be computed. Elemental clauses of two or three terms ANDed together can act like single terms in the query construction process.

5.3.4.2. Solution

In all previous discussions, values for single term s_i have been specified. However, one can easily consider elemental clauses – e.g., a pair of terms ANDed together or a triple of terms linked by AND – in a similar fashion. By estimating the postings values

$$n_{ij} = \frac{n_i * n_j}{N}$$

$$n_{ijk} = \frac{n_i * n_j * n_k}{N*N}$$

as in Chapter 4, and by examining the query and retrieved documents for presence of such pairs and triples, pair and triple feedback weights w_i^p and w_i^t can be determined.

5.3.5. Construction of Derived Query Q_i'

5.3.5.1. Problem

Given suitable single terms and other elemental clauses such as term pairs or triples, it is necessary to have a reasonable methodology to construct the subsequently derived query Q_i' from Q_i.

5.3.5.2. Solution

Since reasonable behavior for the binary weight case is desired, only the DNF scheme described in Chapter 4 seems appropriate. The feedback query can be formed as a combination of suitable singles, pairs, and triples. The DNF algorithm was given in Chapter 4, and the extensions required to adapt it to feedback query construction have been discussed in the sections above, so further discussion is not warranted.

5.3.6. Evaluation of Results

5.3.6.1. Problem

Several key factors are worth considering in discussing evaluation methods for Boolean feedback results. First, the general framework must be described. Secondly, there are a number of possible ways to evaluate feedback. Finally, there are issues of what comparisons are meaningful and of interest.

5.3.6.2. Solution

Regarding the general evaluation framework, evaluation as described in Chapters 3 and 4 will be advocated. Specifically, the documents in a collection will be assumed ranked by a query, recall/precision values will be computed according to that ranking, recall level averages will be determined for a set of queries as a whole, and the average recall at three points (.25, .50, .75 levels of recall) will be reported.

Regarding the treatment of documents retrieved in previous iterations, the "partial rank freezing" approach [Chang, Cirillo & Razon 1971] is employed since: it allows easy comparison of results from one iteration to the next, it allows comparison between different approaches adopted for a particular iteration, and it reasonably mirrors the results seen by a system user.

Regarding comparisons desired, a number of specific issues should be discussed. Note that a key concern is choosing proper base cases for each comparison. While one might wish to contrast feedback starting with a manual query versus that commencing with an automatically formulated query, it is obvious that each such feedback run has a different baseline and so only relative improvements of the two approaches can be contrasted.

To make direct comparisons, two schemes must be based on the same previous search history. For consistency, each must employ uniform methods, that is, the same weighting scheme (e.g., strict Boolean query with binary weights), the same query formulation method (e.g., spt50), the same feedback technique (e.g., $qcount = 2$), and the same handling of previous queries (e.g., $Q_{i+1} = Q_i \ OR \ Q_i'$).

Furthermore, a straightforward base case is needed. Following intuition, that base is simply chosen as the previous iteration's query continued, using the same partial rank freezing approach as for feedback cases. For more detailed explanation see [Salton, Fox, Buckley & Voorhees 1983].

5.3.7. Basing Retrieved Set on User Wishes

5.3.7.1. Problem

In manual searching, it is commonplace to try to formulate either a narrow, medium, or broad search depending on the user's wishes and other constraints. It seems sensible to generalize Boolean feedback query methods to adapt to such requirements.

5.3.7.2. Solution

The parameter "desired number retrieved" which is used to control the construction of DNF queries can serve just this function.

Since various tests had been made (see discussions in Chapter 4) of the effects of varying the retrieval threshold value, that parameter has been fixed in cases considered here. A reasonable value seems to be 50, for a collection such as Medlars.

5.4. Experimental Results

Two sets of tests were made of the DNF feedback query construction process. The first set used the DNF "sort" technique while the second used DNF "narrow." Both methods were initially described in Section 4.4 but the feedback extensions are discussed in Section 5.3.3.

Promising preliminary results were obtained for the Medlars collection using the DNF "sort" approach. However, since performance was so dependent on minor variations of parameter settings it is not felt useful to present such data. Rather, it seems wisest for attention to be focused on the stabler "narrow" method.

Additional explanation, examples of the algorithm being applied to some of the Medlars queries and retrieved documents, and many tables of results for the DNF "narrow" method are given in [Salton, Fox, Buckley & Voorhees 1983]. Consequently only a very general summary is given below.

The feedback process works well for both types of initially provided queries – either manual or automatic – though relative improvements for the manual case are somewhat better. Feedback also works in both the strict Boolean environment and for relaxed p-norm cases, though dramatically higher improvements occur when document weights and p-values are allowed. Including the original query in an OR construction with the newly built expression is beneficial in most cases, especially when the original query was automatically constructed. When the old query is omitted, *qcount* values higher than one are better if a manually formed query is the starting point. Apparently, it is best to obtain in one form or another the proper mixture of information provided by a manual query and by automatic processing of feedback information.

The most interesting cases for initial feedback as well as later iterations are found in Table 5.2. Separate columns are shown for the two possible sources of the initial query. *Qcount* is fixed at 2 and queries are formed using the spt50 criteria.

The base case is that of an original Boolean search continued after an initial retrieval (i.e., one simply looks further down the list returned past the first group considered). Strict Boolean feedback yields 30% improvement for the automatic queries and 82% improvement for manual queries during the initial feedback iteration. Those improvements jump to 101% and 157%, respectively, when the extended Boolean system (i.e., p-values and weights) is employed. Further small improvements result from a second iteration but the third iteration seems of little value.

The final average precision values after two iterations indicate rather a high level of performance, better than that of cosine methods.

5.5. Summary and Conclusions

As promised in earlier chapters, an effective method of beginning with a good natural language query and employing p-norm initial query and feedback query construction methods has been described. Significant performance improvements result from each of two feedback iterations tried with the Medlars collection, resulting in a very high level of average precision, substantially better than that obtained by a simple vector search.

To employ such a method as a "back-end" to a conventional retrieval system, one might follow the procedure given in Figure 5.1.

Table 5.2: Summary of Improvements for Boolean Feedback Process
(Medlars 1033 Docs., 30 Queries)

Description of Case	Initially Available Manual Queries Aver. Prec.	% Improv	Automatic Boolean Queries, NL Terms Aver. Prec.	% Improv
Original Run cont'd once	.2372	-	.3552	-
First Iter. Feedback $Q = Q_{new}$ OR Q_{old}				
Strict Bool., bin. wts.	.4322	82	.4628	30
p=2, wt'd doc. terms	.6103	157	.7131	101
First Iter. Feedback Continued Once (p=2, wt'd doc. terms)	.6672	9	.8014	12
Second Iter. Feedback $Q = Q_{new}$ OR Q_{old} (p=2, wt'd doc. terms)	.7668	15	.8234	3
Second Iter. Feedback Continued Once (p=2, wt'd doc. terms)	.8081	5	.8701	6
Third Iter. Feedback $Q = Q_{new}$ OR Q_{old} (p=2, wt'd doc. terms)	.8058	0	.8734	0

Figure 5.1: Application of P-Norm Feedback

(1) Given a user supplied natural language query, perform a broad initial Boolean statement aimed at retrieving most potentially relevant documents. The initial Boolean query could be supplied by the user, obtained using methods of Chapter 4, or simply formulated as the disjunction of all medium frequency query terms.

(2) Retrieve documents satisfying the above query. The resulting subcollection is the target for subsequent searching and ranking.

(3) Construct, using given terms from the initial query, an automatic p-norm query using $p = 1$ or 2.

(4) Using the current query, rank documents in the subcollection retrieved in step 2 above, according to p-norm rules.

(5) Have the user judge relevance of the top-ranked retrieved documents, and construct a new query using supplied feedback data if requested by the user. Upon request, return to step 4 and iterate the feedback method.

The suggested feedback query construction method is not substantially different from the DNF procedure of Chapter 4. Rather, it is a straightforward generalization, where elementary clauses (i.e., singles or ANDed pairs or triples) have weights determined based on feedback data. The query used in a previous iteration is reflected in subsequent iterations directly since the best form is usually $Q_{i+1} = Q_i \ OR \ Q_i'$, and indirectly by counting the query as one or more (i.e., $qcount \geq 1$) relevant documents in the weight determination calculation.

Summary results of various feedback iterations illustrated the value of the proposed methodology. At long last, performance exceeding that of vector cosine methods was obtained – the absolute performance achieved is quite satisfactory and if reproducible on other collections, worthy of serious consideration as an automatic technique to enhance conventional Boolean system performance.

With this finale on exploration of p-norm query formulation methods, attention must shift to suggestions for improving retrieval system performance by employing better document representations.

CHAPTER 6

EXTENDED VECTORS: MODEL

As mentioned in Chapter 1, it is appropriate to consider extensions of the vector space model to allow handling of various types of concepts. Consequently, Sections 6.1 through 6.4 describe a new extended model, demonstrating in a step-by-step fashion how additional types of concepts can be added to the usual terms only vector. The last section of the chapter, 6.5, completes the discussion by elaborating on the overall rationale for the model and by explaining the applicability of the model to collections of interest.

6.1. Terms Only

Consider a collection, C, containing N documents, that is processed by automatic indexing routines which first eliminate stop words and reduce remaining words to their respective stems. Call T the set of all distinct "term" that result. Typically, a given collection uses only a small fraction of the full vocabulary of the natural language, so the size of set T, i.e., $M_{tm} = |\mathrm{T}|$, is often on the order of 10,000 to 50,000. A dictionary can therefore be stored as an array of character strings by consecutively assigning a unique integer designation to each term in T. The simple system adopted for the static test collections used with SMART programs is to alphabetically order elements of T and then to number them from 1 through M_{tm}. Each term thus has a unique integer assigned: its term concept number.

Any document can be automatically indexed so its original list of words is reduced to stems after removal of stop words. Each stem can have its number of occurrences in the

document counted for use in determining a suitable weight for that term. The i^{th} document $\vec{D}_i$ can then be represented as

$$\vec{D}_i = (tm_{i1}, tm_{i2}, ..., tm_{iM_{tm}}) \tag{6-1}$$

where

tm_{ik} is the possibly zero weight of term k in $\vec{D}_i$.

The similarity between two vectors can accordingly be defined using the cosine measure. Thus, between documents $\vec{D}_i$ and $\vec{D}_j$ the similarity is

$$\begin{aligned} SIM(\vec{D}_i, \vec{D}_j) &= SIM^{COS}_{TM}(\vec{D}_i, \vec{D}_j) \\ &= \frac{\sum tm_{ik} \cdot tm_{jk}}{\sum tm_{ik}^2 \cdot \sum tm_{jk}^2}. \end{aligned} \tag{6-2}$$

Here the notation $SIM^{\text{measure}}_{\text{concept-type}}$ is adopted to clarify that the designated measure (e.g., cosine) is applied to concepts of the given type (e.g., terms). While not necessary in this simple case, since the model presented so far leaves little variation possible, it is very convenient to use in the extended model described below.

6.2. Additional Information – Authors

Let M_{au} be the number of distinct authors for the N documents of collection C. A dictionary or list of authors can be created, assigning unique identifiers to each author, yielding a set A of M_{au} author concepts. The question now is how these concepts can be combined with terms. One might propose including A in T, and such an approach will be discussed below, but it does seem reasonable to at least consider treating author names differently than regular language terms.

The approach taken by Salton [1963] when adding various types of bibliographic data. was to simply extend the document vectors given by (6-1) above. Thus, a single extended dictionary E was formed, with entries 1 through M_{tm} being terms, and entries $M_{tm}+1$ through $M_{tm}+m_{au}$ being author names. Specifically, he defined an extended vector with terms and authors as

$$\vec{D'}_i = (tm_{i,1}, \ldots, tm_{i,M_{tm}}, tm_{i,M_{tm}+1}, \ldots, tm_{i,M_{tm}+M_{au}}) \tag{6-3}$$

term portion author portion.

However, it seems to be practically and conceptually better to more clearly separate the extended vector into two subvectors. Representing the term subvector for the i^{th} subvector as $\vec{tm}_i$ and the author subvector as $\vec{au}_i$, the i^{th} document is described as

$$\vec{D}_i' = (\vec{tm}_i, \vec{au}_i). \tag{6-4}$$

Expanded, the subvectors have the equivalent form

$$\vec{D}_i' = (tm_{i1}, \ldots, tm_{i,M_{tm}}, au_{i1}, \ldots, au_{iM_{au}}). \tag{6-5}$$

The collection as a whole, which was originally a single matrix of N (document) rows and M_{tm} (term) columns, e.g.,

$$\underline{D} = \begin{vmatrix} tm_{11} & \cdots & tm_{1M_{tm}} \\ \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot \\ tm_{N1} & \cdots & tm_{NM_{tm}} \end{vmatrix} \tag{6-6}$$

or, equivalently

$$\underline{D} = \begin{vmatrix} \vec{tm}_1 \\ . \\ . \\ . \\ \vec{tm}_N \end{vmatrix} = \underline{TM} \qquad (6\text{-}7)$$

can now be compactly referred to as

$$\underline{D} = \begin{vmatrix} \vec{tm}_1 & \vec{au}_1 \\ . & . \\ . & . \\ . & . \\ \vec{tm}_N & \vec{au}_N \end{vmatrix} = \left| \underline{TM} \;\; \underline{AU} \right| \qquad (6\text{-}8)$$

instead of using the full form

$$\underline{D} = \begin{vmatrix} tm_{11} & \cdots & tm_{1M_{tm}} & au_{11} & \cdots & au_{1M_{au}} \\ . & \cdots & . & . & \cdots & . \\ . & \cdots & . & . & \cdots & . \\ . & \cdots & . & . & \cdots & . \\ tm_{N1} & \cdots & tm_{NM_{tm}} & au_{N1} & \cdots & au_{NM_{au}} \end{vmatrix} \qquad (6\text{-}9)$$

The separation of attributes characteristic of database systems and found in Boolean systems that divide documents into fields (e.g., abstract, author, descriptors) is often useful and should be made possible by a generalized vector system. A typical p-norm query exploiting such a separation is given in Figure 6.1, where author concepts have a designatory prefix while terms, the default type, do not. Thus, "ALGOL" is shorthand for "term.ALGOL" in the example.

Figure 6.1: Boolean Query with Multiple Concept Types

$$\begin{bmatrix} \text{"author.Wirth\_N"} \\ \text{OR} \\ \text{"author.Hoare\_A"} \end{bmatrix} \; AND \; \begin{bmatrix} \text{"ALGOL"} \\ \text{OR} \\ \text{"PASCAL"} \end{bmatrix}$$

An additional capability made possible by the separation of terms and author subvectors is that a user can, either in a vector or p-norm query environment, specify relative importance of authors versus terms for general retrieval situations.

To illustrate this point, a certain amount of notation will be needed. Given two vectors, $\vec{D}_i$ and $\vec{D}_j$, the overall similarity between them will be written as

$$SIM(\vec{D}_i, \vec{D}_j). \tag{6-10}$$

Considering only a portion of each of the vectors, say the terms, the cosine similarity for that portion will be described as

$$SIM^{COS}_{TM}(\vec{D}_i, \vec{D}_j) = SIM^{COS}(\vec{tm}_i, \vec{tm}_j) \tag{6-11}$$

while the inner product similarity between authors would be

$$SIM^{IP}_{AU}(\vec{D}_i, \vec{D}_j) = SIM^{IP}(\vec{au}_i, \vec{au}_j). \tag{6-12}$$

When author and term vectors are both used for calculating a combined similarity value on the overall vectors, relative weights can be applied to the similarities for each concept type. These weights will be designated by

$$w_{au} \text{ and } w_{tm} \tag{6-13}$$

respectively. Though not necessary for vector methods, the rule

$$w_{au} + w_{tm} = 1.0 \qquad (6\text{-}14)$$

will be followed since comparisons between different pairs of weights is thereby facilitated.

In order to specify the situation for a case of combining similarities, it is convenient to use a simple table. Thus, to indicate that cosine similarity on terms should be used, and that a weight of 0.8 should be assigned to that component, the first row of the table of Figure 6.2 suffices. Similarly, weighting the author component .2 and using inner product similarity is shown in the second line of Table 6.1.

Table 6.1: Tabular Form of Combined Similarity Specification

Concept-Type	Similarity	Weight
tm (terms)	cosine	.8
au (authors)	inner-prod.	.2

For this example, the formula for combined similarity is easily constructed.

$$\begin{aligned} & SIM(\vec{D}_i, \vec{D}_j) \qquad (6\text{-}15) \\ &= w_{tm} \cdot SIM^{COS}_{TM}(\vec{D}_i, \vec{D}_j) + w_{au} \cdot SIM^{IP}_{AU}(\vec{D}_i, \vec{D}_j) \\ &= 0.8 \cdot SIM^{COS}(\vec{tm}_i, \vec{tm}_j) + .2 \cdot SIM^{IP}(\vec{au}_i, \vec{au}_j). \end{aligned}$$

Incidentally, the various subvectors could be constructed using different weighting schemes; an additional column in Table 6.1 could show that, for example, term weights were computed using the scheme tf*idf while author entries were given binary weights. Such flexibility might be of practical benefit, since author matches may be considered qualitatively different from term matches. One suggestion is to have binary matches for the former and real-valued weights for the latter.

6.3. Additional Information Based on References

In addition to terms and authors, other types of information are available in many collections. Dates and controlled vocabulary terms may be properly separated from regular terms. Of more concern here, bibliographic information such as direct references between documents and other derived measures such as those of bibliographic coupling and cocitation strength can be employed.

The first subsection below reviews some of the research work that has gone on in this area during the last twenty years. Various measures were defined, preliminary tests were made, and a number of theoretical and experimental studies performed. The bibliographic measures described have been useful in both retrieval and clustering applications. Since the next chapter deals with clustering of extended vectors, however, previous work relating to clustering will only be touched on lightly in this section.

After the relevant literature has been reviewed discussion of the extended vector model will continue. The particular measures of interest here will be defined and described, and then construction of vectors based on those measures will be explained.

6.3.1. Previous Work with Bibliographic Information

Kessler [1962] selected the proceedings of a 1958 conference on transistors, including tutorial, editorial, original research, and review articles – a total of 40 papers with 947 references to various journals and other media – and applied his newly defined measure of bibliographic coupling to see if the resulting similarities computed between documents would be usable to separate the documents into meaningful groups. The strength of bibliographic coupling was simply defined as the number of articles in common between the bibliographies of a pair of articles.

Kessler followed up that study with another involving 265 articles [1963a], a case study report of ten articles chosen from among 8186 [1963b], and a comparison between the classification results on 334 papers obtained by manual subject indexing versus that automatically produced using bibliographic coupling data [1965b]. Though none of the results very clearly demonstrated that the new measure would be of special value in retrieval, the success evinced in classification tests suggested the utility of including such information in retrieval systems, and so the TIP programs and files were suitably enhanced ([Kessler, Ivie & Mathews 1964], [Kessler 1965a]).

Small [1973] suggested, however, that instead of using bibliographic coupling – i.e., the written agreement between each one of a given pair of articles which thus relates the value of past publications – the perspective of later authors should be utilized. His co-citation measure enlists subsequent writers in related fields, people particularly qualified to pinpoint the intellectual connections between documents, as indexers, by way of referring to several publications in a bibliography. Pairs of such items that are cited in common are co-cited, and the strength of the co-citation measure is simply the number of articles in which later authors make such co-citations.

Though Small criticizes bibliographic coupling as being retrospective only, Marshakova, who independently defined the measure of co-citation during the same year as Small, realized that bibliographic coupling would also change over time and so was not as critical of the earlier measure [Weinberg 1974]. Many other authors have commented on the value, limitations, and theory of each of the above mentioned kinds of citation information. Cronin [1981] recently pointed out the need for a theory of citing to explain why authors refer to other works. Subsequently, Lawani has classified instances of the most

obvious source of error, author self-citation [Lawani 1982], and also has statistically demonstrated the validity of using citation counts to assess the influence of publications [Lawani & Bayer 1983]. Such counts, however, probably should be extrapolated to expected lifetime figures [Geller, Cani & Davies 1981] before being used.

Several authors have considered using citation information in practical applications unrelated to retrieval, such as for deciding when to retire documents from a collection, but care must be exercised in properly relating citation counts and document age [Sandison 1975, Cawkell 1976]. Brittain and Line [1973] touch on many such analytical uses for citation information and further provide handy outlines, including one listing the best sources to obtain the basic raw citation data from.

Small [1980] utilized co-citation information to create nets representing the interactions among authors, and other nets to show connections between ideas; examination of the contexts where co-citations occur allow one to attempt to represent the underlying conceptual structure of a discipline. Then, Small [1981] used co-citations to analyze information science and how it relates to various social sciences. His view is that citation of a document is an act of symbolic usage of the idea embodied by the document, and that citation contexts add in the perspective of later authors, which are often consistent with the views of others [Small 1978]. Small [1974] also explains how tri-citation information can be used to better portray the "scenery" of citation patterns; since tri-citations occur in smaller numbers than co-citations, however, it is likely that this measure of three way coupling will be used primarily in specialized clustering and modelling applications such as that considered by Small.

Following the lead of the initial exploratory work of Kessler, various studies have used clustering methods on citation data. Salton and Bergmark [1979, 1980] studied the computer science literature using citation data and showed how the field could be subdivided into smaller areas. Journals too can be studied by clustering; Small and Koenig used bibliographic coupling [1977] while Arms and Arms [1978] used citations as the raw data. Carpenter and Narin [1973] partitioned scientific journals based on citations since it was assumed that journals dealing with the same subject area: 1) will have similar journal referencing patterns, and 2) will refer to each other. In general, the results of all these clustering studies seem to match intuition and so seem valuable for research in bibliometrics or the sociology and history of science.

Various related approaches have been followed in trying to use citation information to aid the retrieval process. The usual argument is that citations are objective, easily quantified, language independent indexing units [Garfield 1979]. However, it is clear that authors have various motives when they cite other works, that the value of a citation relates to the total size of the bibliography, that citation habits are affected by an author's knowledge of foreign languages, and that the sought after "unit" is an elusive goal [Weinberg 1974]. Nevertheless, a wide range of instances of how citations can aid retrieval have been explored.

An initial problem to confront is how to incorporate citation data into document representations. Kwok [1975] simply added the words in titles of cited documents as extra keywords to better index the citing article. O'Connor reversed the process, by adding words from citing statements to the set of index terms of the cited article [1982]. Later, due to the encouraging success evinced by manual tests, O'Connor devised automatic pro-

cedures to identify the citing statements in a text [1980a]. A related study was that of Herlach [1978], where both the number of occurrences and the location in a paper (e.g., in Introduction, Methods, Results, or Discussion section) of references help determine the significance of links between cited and citing articles. Closest to the extended vector model proposed here, however, are several studies by Salton [1963, 1971b] and Michelson et al. [1971]. In Salton's earlier study, direct citations between documents were considered, and used instead of or along with terms in a vector formulation; a small scale preliminary test showed improvements over those expected from random behavior. The later study used 200 documents and 42 queries, and showed that improvements occur when vectors with both term and direct citation concepts are employed compared to when terms only are allowed. A special feature of the experiment was that queries were each based on a single source document, so either the query terms or the bibliography of the original document could be used interchangeably. Finally, the SMART team of Michelson et al. had success when extended vectors of the form shown in equation (6-3) were applied to feedback problems. However, they only experimented with the rather small ADI collection and ran into difficulties relating to their method of computing vector correlations.

Citation information in indexes can also be used in the retrieval process. Given a citation index, one can employ cycling, which can be mathematically described [Cummings & Fox 1973] but can easily be understood as the repetitive movement backward and forward in time: beginning with one or more articles present in a citation index go backward to the citing documents, go forward by considering their references, go back using the new larger set, etc. Furthermore, with an online author citation index and a pair of names of authors who have each worked in the field of interest, one can find articles that each cite at least one publication of each author [White 1981].

Though clustering will be emphasized more in the next chapter, several studies have used clustering based on citation information to aid retrieval. Schiminovich [1971] used direct links and a pattern analysis scheme to construct clusters that could later aid retrieval. Bichteler and Parsons [1974] modified the Schiminovich algorithm and reported that retrieval with a citation instead of a term classification gave higher precision at low recall levels.

Of all the studies using bibliographic coupling or co-citation information, the one by Bichteler and Eaton [1980] which combines both measures is the most relevant here. Using 1712 papers, they showed that a composite function giving the linkage similarity in terms of the normalized total of the bibliographic coupling and co-citation connections performed better than when only bibliographic coupling was measured. Building upon an earlier suggestion by Amsler, they tried one composite similarity function and evaluated the retrieval performance for a set of 10 queries. These tentative findings are supported by more comprehensive tests reported in Chapter 8 below.

With this historical background, it is appropriate now to focus on the precise use of bibliographic data within the extended vector model. It should be clear that the emphasis is on combining all of the easily identified useful types of bibliographic data in a simple model so that testing of clustering and composite retrieval functions will be facilitated.

6.3.2. Example and Definitions

For illustration purposes it is convenient to represent bibliographic connections between articles using network diagrams [Cummings & Fox 1973], with nodes designating documents and arcs indicating references between documents. By way of example, consider the network in Figure 6.2 and the tabular listing of its arcs given in Table 6.2.

Figure 6.2: Example of Citation Network

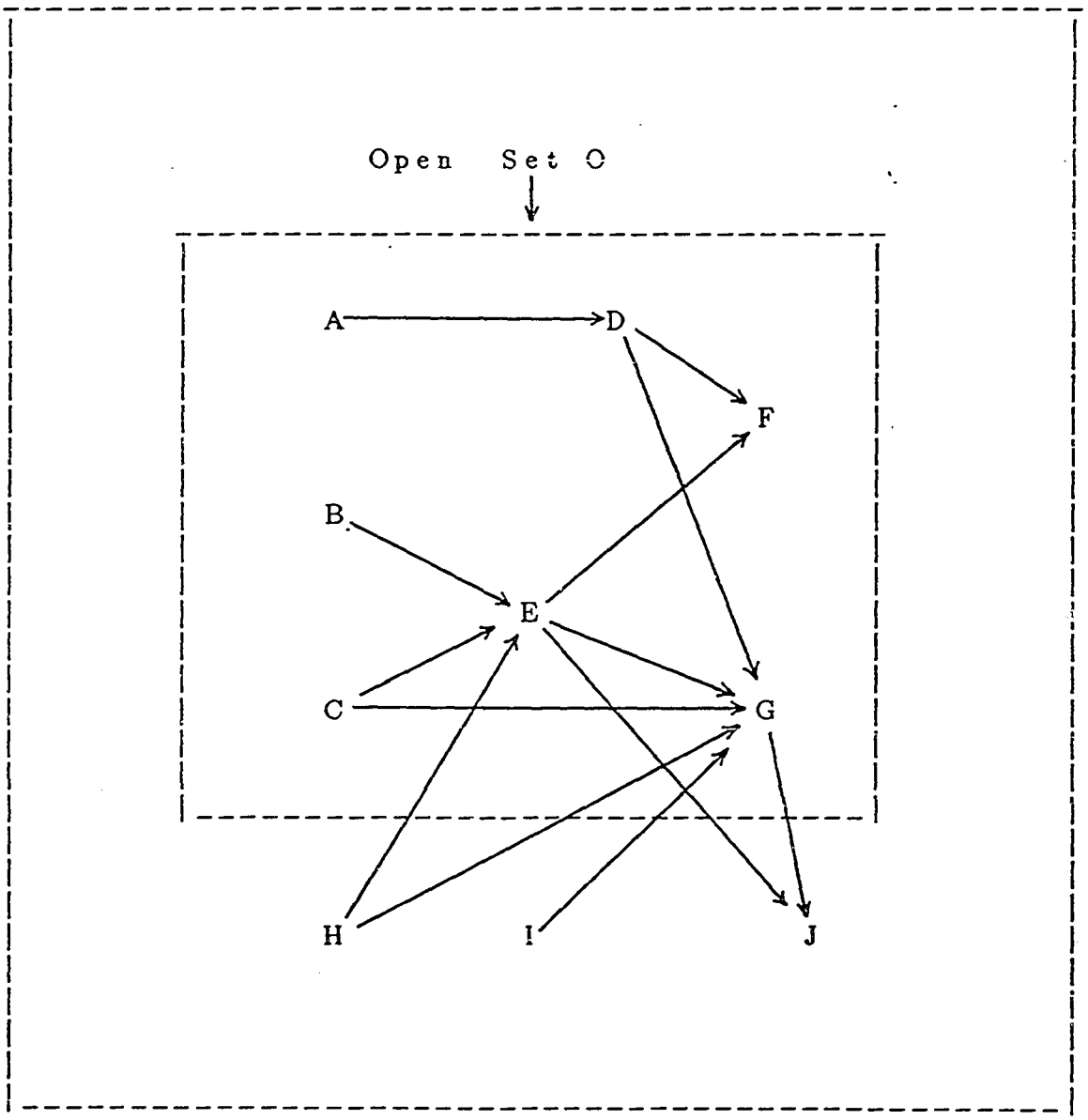


Table 6.2: Chart of Citation Arcs
(Primary Sort on Citing,
Secondary Sort on Cited Docs.)

Citing Doc.		Cited Doc.	Citing Doc.		Cited Doc.
A	→	D	E	→	G
B	→	E	E	→	J
C	→	E	G	→	J
C	→	G	H	→	E
D	→	F	H	→	G
D	→	G	I	→	G
E	→	F			

Note that it is conceptually clearer, when possible, to work with a closed set of documents. e.g., *C* in Figure 6.2, where all arcs are inside the set. However, limitations on data availability may require using an open set like *O*, where arcs may come in from or go out to (not present) external documents as well. Another comment to bear in mind is the unconventional use of terminology in Figure 6.4 and elsewhere in subsequent discussions. The common practice in bibliometric studies is to talk about "referring" instead of "citing" articles [Sandison 1975]. Yet, to highlight the symmetry of the situation, the term "citing" will be freely used here instead when the companion term "cited" is employed. Thus, if A refers to D so that D is cited by A, it can be stated that A is the citing document and D is the cited document.

Based on the reference pattern for a set of documents, one may define various derived measures of the interconnection between those documents. The relevant notation and definitions follow, using the data of Figure 6-3 to illustrate each point.

(6-16) $A \rightarrow D$ Direct Reference

when A refers to (cites) document D, so that D is referred to (cited by) A. By definition, $D \rightarrow D$ always holds.

(6-17) $A \rightarrow^{k} G$ Indirect Reference

when A indirectly refers to (cites) G (e.g., at distance $k = 2$), so that G is indirectly referred to (cited by) A.

An example is when there is some document such as D where $A \rightarrow D \rightarrow G$. Note that k designates the number of arcs that must be traversed to reach the cited document; when $k = 1$ it is not shown since the notation of (6-16) is used instead.

Now, citing directly as given in (6-16) or indirectly as in (6-17) are binary events – either they occur or not. On the other hand, the next two definitions can result in an assignment of weights that are based upon integer counts.

(6-18) B and C are bibliographically coupled [Kessler 1962]

if some document, say E, is referred to by both B and C.

Hence a computer can count how many articles provide a coupling connection in a similar fashion to E – in Figure 6.2 there are no more – and define the degree of bibliographic coupling. Thus, for arbitrary documents i and j,

$$bc_{ij} = |\underline{D}'|$$

where

$$D_k \in \underline{D}' \Leftrightarrow D_i \rightarrow D_k \text{ and } D_j \rightarrow D_k$$

and $\underline{D}'$ is restricted to the document set of definition, e.g., O.

In the example of Figure 6.3,

$$bc_{B,C} = 1 \quad \text{and} \quad bc_{D,E} = 2$$

since one document, E, is referred to by both B and C, while two documents, F and G, are each referred to by both D and E. Thus B→E, C→E, and D→F, E→F, D→G, E→G.

Note that bc_{ii} is simply the number of articles referred to by document i, i.e., the length of its bibliography. This is useful, since one would hope that the coupling of a document to itself would be a maximum that other couplings could be normalized by.

(6-19) F and G are co-cited [Small 1973]
if some document, say D, refers to both of them in its bibliography.

One can count the total number of articles that each refer to both F and G. For arbitrary documents i and j, the co-citation strength is then given by

$$cc_{ij} = |\mathcal{D}''|$$

where

$$\mathcal{D}'' \subseteq C,$$

the source set of documents considered, and

$$D_k \in \mathcal{D}'' \Leftrightarrow D_k \rightarrow D_i \text{ and } D_k \rightarrow D_j.$$

Note that cc_{ii} is simply the number of articles that cite document i, that is, its citation count. That value can be used for normalizing other cc values or to gauge the importance of the given article. In the example, then, one observes that

$$cc_{E,G} = 2, \quad cc_{F,G} = 2, \quad cc_{F,J} = 1.$$

The first observation is based on the fact that documents C and H each "co-cite" or refer to both E and G. The reader may wish to verify the validity and find the relevant data for

the other two co-citation counts given above.

(6-20) A and D are linked if either A → D or D → A [Salton 1963].

This definition allows the computer to symmetrically view citation connections between documents, regardless of the ordering of the articles based on time of publication. More formally,

$$ln_{i,j} = \begin{cases} 1 & \text{if } D_i \rightarrow D_j \\ 1 & \text{if } D_j \rightarrow D_i \\ 1 & \text{if } i = j, \text{ by definition} \\ 0 & \text{otherwise.} \end{cases}$$

In the example, there are ln_{ij} values of 1 for pairs such as A and D or C and G.

Now that the various measures have been defined and illustrated, it remains to be seen how they can be utilized to enhance document representations by incorporation into additional concept types.

6.4. Submatrices for Reference Information

6.4.1. Definitions

Document representations as given in the first two sections of this chapter follow the assumption that for each type of concept such as that of textual terms or author names a dictionary of all possible entries can be constructed. Each concept is thus assigned a unique concept number. Concept types "term" and "author" lead to separate subvectors for each document (as in equation 6-4) and separate submatrices for the collection as a whole (as in 6-8). The term submatrix $\underline{TM}$ has dimensions N x M_{tm} while the author submatrix has dimensions N x M_{au}.

For each document it is straightforward using the definitions of the last section to determine values of the linkage, bibliographic coupling, and co-citation measures between that document and any other document. Rather than using a dictionary to provide concept numbers, the document numbers themselves can be used so that

$$M_{bc} = M_{cc} = M_{ln} = N \tag{6-21}$$

and submatrices $\underline{BC}$, $\underline{CC}$, and $\underline{LN}$ will each be of size N x N. Note that according to the definitions of the various measures all diagonal entries will be non-zero but in general the submatrices will be sparsely populated.

To obtain some intuition as to the meaning of these submatrices, consider the subvectors $\vec{bc}_i$, $\vec{cc}_i$, and $\vec{ln}_i$ for the i^{th} document. Diagonal entries are

$$\begin{aligned} bc_{ii} &= \text{no. of references in bibliography of } i \\ cc_{ii} &= \text{no. of articles that refer to } i \\ ln_{ii} &= 1 \end{aligned} \tag{6-22}$$

where another way to understand cc_{ii} is to view it as the incoming citation count.

Off diagonal entries show how the i^{th} document relates to other documents. Thus, the j^{th} column of each submatrix shows how documents relate to the j^{th} document - one in effect treats a document as a "bibliographic concept". Off diagonal values have the following significance:

$$\begin{aligned} bc_{ij} &= \text{no. of articles referred to by both } i,j \\ cc_{ij} &= \text{no. of articles that each refer to both } i,j \\ ln_{ij} &= 1 \text{ if the } i^{th} \text{ doc. refers to the } j^{th}, \text{ or vice versa} \end{aligned} \tag{6-23}$$

One important point in these definitions is the fact that bibliographic connection is dependent on the overall set of documents being considered. This is most clearly seen in

the case of co-citation values. In the universe of all documents, there may be many articles that co-cite a given pair of documents $\vec{D}_i$, $\vec{D}_j$, leading to a high value of cc_{ij}. If the set of articles considered is severely restricted, say to only articles in a certain year or appearing in a certain journal, c_{ij} may be reduced because the co-citing articles are excluded.

6.4.2. Example

Based on these definitions, one can illustrate the citation submatrices $\underline{bc}$, $\underline{cc}$, and $\underline{ln}$ for the example set O given in Figure 6.2 and Table 6.2. The desired submatrices are shown in Figures 6.3 through 6.5.

The $\underline{cc}$ submatrix shown is sparser than would occur in practice, since the example is only based on a small fictitious collection, with only a few references per article. The $\underline{ln}$ submatrix, as expected, has many entries, but one might expect that since all of its entries are binary valued, then the much rarer $\underline{bc}$ and $\underline{cc}$ entries with values greater than one would be a more accurate indication of document similarity.

Figure 6.3: $\underline{bc}$ Submatrix

	A	B	C	D	E	F	G
A	1						
B		1	1				
C		1	2	1	1		
D			1	2	2		
E			1	2	3		
F						0	
G							1

Note: $bc_{E,G} \neq 1$ since J is not $\in O$.

Figure 6.4: $\underline{cc}$ Submatrix

	A	B	C	D	E	F	G
A	0						
B		0					
C			0				
D				1			
E					3		2
F						2	2
G					2	2	5

Note: this includes the fact that H cites E,G when $cc_{E,G}$ is computed. The reason is that H is in the source set C for co-citations.

Figure 6.5: $\underline{ln}$ Submatrix

	A	B	C	D	E	F	G
A	1			1			
B		1			1		
C			1		1		1
D	1			1		1	1
E		1	1		1	1	1
F				1	1	1	
G			1	1	1		1

The $\underline{cc}$ submatrix shown is sparser than would occur in practice, since the example is only based on a small fictitious collection, with only a few references per article. The $\underline{ln}$ submatrix, as expected, has many entries, but one might expect that since all of its entries are binary valued, then the much rarer $\underline{bc}$ and $\underline{cc}$ entries with values greater than one

would be a more accurate indication of document similarity.

One disadvantage of the *bc* and *cc* measures is that they depend upon the date of publication of documents since that strongly affects which other documents cite them or are referred to by them. Thus, document J in Figure 6.2 is unlikely to be bibliographically coupled with others since it was apparently one of the first ones published. Similarly, recent articles like A, B, and C lack co-citations because nothing refers to any of them. Most likely, that situation would be remedied in a few years time as new articles appear. By using both of the *bc* and *cc* measures in an additive fashion, however, it is hoped that this effect of age would be diminished. That is, in computing $SIM(\vec{D}_i, \vec{D}_j)$, one should always include $SIM^{BC}(\vec{D}_i, \vec{D}_j) + SIM^{CC}(\vec{D}_i, \vec{D}_j)$. The total called for would balance out handling of document pairs that are:

(1) both very old, probably giving low bc_{ij} value but high cc_{ij} value;

(2) both of medium age, having moderate values each of b_{ij} and cc_{ij}; or

(3) both recent, suggesting high bc_{ij} value but low cc_{ij}.

The symmetrical definition of ln_{ij} allows for a similar balanced handling of pairs of articles since linkage does not depend on which of the two articles appeared first. In a sense, linkage is already an additive or combined measure constructed as the logical OR of one matrix containing citing connections with the transpose matrix containing cited connections.

The use of *bc*, *cc*, and *ln* submatrices seems justified as an initial approach to better incorporating bibliographic data in the vector space model. Experiments in later chapters will contrast the utility of these measures and see how they can best be combined to aid retrieval system performance. The first requisite for such utilization, however, is an effective means to include the appropriate subvectors when computing similarities.

6.4.3. Composite Similarity

Thus, given these submatrices, the question is how one should best employ the data to compute the similarity between two documents. The usual or direct approach toward utilization of bibliographic coupling or co-citation measures is to define the document similarity as a normalized version of the proper single matrix element. Thus, using the cosine function, one would have

$$SIM_{BC}^{COS}(\vec{D}_i, \vec{D}_j) = \frac{bc_{ij}}{bc_{ii} \cdot bc_{jj}} \tag{6-24}$$

$$SIM_{CC}^{COS}(\vec{D}_i, \vec{D}_j) = \frac{cc_{ij}}{cc_{ii} \cdot cc_{jj}}. \tag{6-25}$$

In effect these give the similarity based only upon direct bibliographic coupling and direct co-citations, respectively. For example, in Figure 6.2:

$$SIM_{BC}^{COS}(D, E) = \frac{2}{6} = .33$$

and

$$SIM_{CC}^{COS}(F, G) = \frac{2}{10} = .20.$$

However, there are several difficulties relating to using these direct measures. First, in a typical collection, there are not many co-citations; usually also, only one document will co-cite a given pair. Even in groups of articles that deal with a common topic, some pairs in the group may never be co-cited. While the example in Figure 6.2 is too densely connected to show this, one can imagine having X and Y co-cited, as are Y and Z, but not have X and Z be co-cited. This situation causes no problems if what is involved is a single link clustering which automatically utilizes the indirect relationship. But in a feedback

environment, finding X through an initial iteration would not allow the location of document Z until later feedback iterations, causing a loss in recall at lower recall levels.

Second, consider documents D and E. Their bibliographic coupling is 2 which indicates a fairly strong relationship. Perhaps, though, there should be some further means of acknowledging the fact that D and E are related in other ways as well. In Table 6.2, note that document C is also bibliographically coupled with each of these through joint citation of G. Consider too the matrix of Figure 6.3, where both D and E appear in the column for document C. Further note that the two rows for D and E have three matching entries, instead of the normal two that arise from coupling. Utilizing indirect couplings like these should enhance recall – highlighting additional similarities that would be ignored otherwise.

Finally, consider the practical problems of implementation. Assuming the existence of bibliographic information based matrices as explained earlier, how can the similarity be computed between documents, or between queries and documents? Certainly, formulas (6-24) and (6-25) could be followed. But if one document was a cluster centroid instead (as discussed in the next chapter), or the query had feedback information from several retrieved relevant documents, it is unclear how to proceed. Thus, if Q_i is of form

$$Q_i = \alpha_0 \cdot Q_{i-1} + \alpha_1 \cdot D_{i1} + \alpha_2 \cdot D_{i2} + ... + \alpha_k \cdot D_{ik}$$

it is problematic to define

$$SIM_{BC}^{COS}(\vec{Q}_i, \vec{D}_j).$$

One could use

$$\sum_{l=1}^{k} \alpha_l \cdot SIM_{BC}^{COS}(\vec{D}_{il}, \vec{D}_j)$$

but there should probably be better normalization.

What is proposed, then, is simply using the cosine similarity between two subvectors as a measure of both direct and indirect connection. Defining

$$SIM_{BC}^{COS}(\vec{D}_i, \vec{D}_j) = SIM^{COS}(\vec{bc}_i, \vec{bc}_j) \tag{6-26}$$

results in a similarity between D and E (see rows D and E of Figure 6.3) of

$$\frac{1\cdot 1 + 2\cdot 2 + 2\cdot 3}{\sqrt{(1^2 + 2^2 + 2^2)\cdot(1^2 + 2^2 + 3^2)}} = 0.98.$$

As hoped for, a very high similarity for these two articles results.

Definition (6-26) then is utilized to compute similarity due to bibliographic coupling. The similarity for co-citations is likewise defined as

$$SIM_{CC}^{COS}(\vec{D}_i, \vec{D}_j) = SIM^{COS}(\vec{cc}_i, \vec{cc}_j) \tag{6-27}$$

and for links as

$$SIM_{LN}^{COS}(\vec{D}_i, \vec{D}_j) = SIM^{COS}(\vec{ln}_i, \vec{ln}_j). \tag{6-28}$$

Because of the definition $ln_{ii} = 1$, the same approach applies to computing similarities for the $\underline{ln}$ submatrix. As with $\vec{bc}$ and $\vec{cc}$ subvectors, the procedure includes both direct references between documents and indirect references in order to compute the similarity between $\vec{ln}$ subvectors. Thus, in the example of Figure 6.2, the similarity between E and G, based on their $\vec{ln}$ subvectors as given in Figure 6.5, is

$$\frac{1 + 1 + 1}{\sqrt{5\cdot 4}} = .67.$$

Two other documents, F and G, have $\vec{ln}$ similarity of

$$\frac{1+1}{\sqrt{3 \cdot 4}} = .58$$

even though there is no direct citation between them; in effect the method considers paths $F \rightarrow^2 G$ and $G \rightarrow^2 F$ to help determine the magnitude of the association between F and G.

6.5. Multiple Concept Types

So far in this chapter a number of extensions to the vector model have been proposed. This section aims at tying those notions together into the multiple concept type model of vector document representation in information retrieval systems. Specifically, the rationale for the model is briefly explored and its applicability to a few experimental test collections is demonstrated.

6.5.1. Rationale

The following subsections present some of the motivations and justifications of the proposed model.

6.5.1.1. Similar Method in Other Fields

Using additional factual information such as author name or perhaps publication date has precedent in the areas of database processing and in the operation of many common Boolean retrieval services. File management and database systems always relate a number of fields or attributes to a given real-world entity; an article must have a title, journal name, author name (or names), and date of publication.

Database and Boolean systems commonly allow explicit references to each subfield or attribute. Queries typically include expressions with syntax such as

AUTHOR = Smith or Article.author = Jones.

Range searches on publication date are often employed to restrict the size of retrieved sets, and are based on testing of the date field.

There is no straightforward way to accommodate these needs in the simple vector model – for example, one simply cannot distinguish a name appearing in the author field from the same name in the abstract, unless special indexing techniques are employed to simulate the separation of concepts into different types.

6.5.1.2. Searching Many Fields

Retrieval experiments testing searching on each of the possible textual fields of a document representation and on combinations of those fields indicate that allowing searching on several separate fields gives better performance. Thus, in [Kaizer et al. 1982], experiments in which several different representations were searched separately are described. Each representation included different fields or combinations of fields and so searches were restricted to the fields present in a certain document representation. Typically, two searches using different representations to express the same query would have only about 15% overlap in the resulting retrieved sets. Since none of the chosen representations located all relevant items, an appropriate strategy seems to be to make up several searches and merge the results. Equivalently, a single search using all of the available fields might give very good recall and possibly good precision.

6.5.1.3. Clarity

The subvector and submatrix notation introduced in Section 6.2 is conceptually clean. Each type of information is named differently, can be treated separately, and yet all data is present in vector form. Since it is often useful to have independence among vector ele-

ments, that provision can be approximated among entries of each subvector and optionally be ignored across subvector boundaries. One can compute similarities as an appropriate linear combination of the subvector similarities, as discussed in subsequent chapters.

6.5.1.4. Updating

The various items of information for different subvectors have different sources and different requirements for updating. Thus, term information is available initially and unchanging over the life of a document (unless, for example, spelling errors are corrected). On the other hand, co-citation information continuously changes, since each newly added article may cite numerous other articles previously stored in the collection. Hence, it would be helpful to only update submatrix *cc* and to protect *tm* from accidental revision.

6.5.1.5. Indexing

The indexing process is often different for data that will go into each subvector. Thus, terms need to be stemmed while authors should be normalized into, say, "last-name_first-initial" form. Operationally, it is useful to separate such data into groups that can each be manipulated in a uniform fashion. This situation is even more obvious in the case of handling bibliographic references, where multi-step processing is required to obtain such submatrices as those for strength of co-citation (*cc*) or bibliographic coupling (*bc*).

6.5.1.6. Weighting

Weighting methods may vary for different subvectors. Dates should undoubtedly receive binary weights, whereas terms benefit from applying an inverse document frequency (idf) factor. Bibliographic submatrices should also use some type of weighting.

6.5.1.7. Solution to Vector Problems

Finally, separation into multiple subvectors may facilitate handling of problems relating to long or missing subvectors. The general issue is whether it is indeed proper to truly separate different subvectors, or to treat the entire vector as a whole – ex., should one compute the similarity of two vectors as the cosine between them or compute the similarity of each subvector and then combine the similarities, possibly as an appropriate linear combination. The normalization in cosine similarity computations will yield different results.

Of course, dividing into subvectors allows one to later compute similarity in either of the two proposed ways, since it is easy enough to merge subvectors into a composite vector. Going the other way is much harder.

When a subvector is empty or has many entries, the separate subvector approach restricts the effect of that fact specifically to the similarity value associated with that subvector. On the other hand, if only a single composite similarity is computed, the effect of very long or very short (i.e., those with many or few non-zero or non-null) subvectors may, due to normalization by overall vector length, unduly influence other subvectors' contributions to the final similarity.

Michelson et al. [1971], working with feedback of vectors which include term and bibliographic concepts, point out that query-document similarities should only consider types of data actually present in a query, after the query is enhanced by feedback terms. An illustration of their point can be seen in the context of retrieving articles present in a collection from the *Communications of the ACM* (CACM). In that collection, computer articles are classified into one or more of about 200 categories used in the sister publication, *Computing Reviews*. A subvector for *Computing Reviews* (cr) categories therefore seems

appropriate. Many CACM articles have no $\underline{cr}$ categories since they were written before the category system was inaugurated. The usual cosine computations, treating the document vector as a whole, would normalize by the entire vector length, regardless of whether a $\vec{cr}$ subvector was present.

6.5.2. Model Applied to ISI and CACM Collections

The collection of articles identified using data supplied by the Institute for Scientific Information© (ISI©) is of information science documents receiving many citations. Authors in standard form, e.g., Salton_G, and textual terms, i.e. stems remaining after a stop word list has been applied to the title and abstract, were indexed using normal SMART methods. Some 90,000 source-cited document number pairs were processed to yield co-citation values for all pairs of cited documents. The final concept type scheme was therefore obtained as shown in Table 6.3.

Table 6.3: ISI Concept Types

Number	Designation	Description
0	tm	terms
1	au	authors
2	cc	co-citations

Since the ISI collection lacked other types of bibliographically based information, and since a number of concept types were already available for the CACM collection (see Table 6.4), a more extensive test of the multiple concept type approach was planned.

Table 6.4: CACM Concept Types

Number	Designation	Description
0	tm	terms
1	au	authors
2	bi	bibliographic information
3	cr	*Computing Review* categories
4	cc	co-citations
5	bc	bibliographic coupling
6	ln	links

References between all CACM articles were considered, so the bibliographic based subvectors were fairly short. However, the available data did allow easy construction of $\vec{bc}$, $\vec{cc}$, and $\vec{ln}$ subvectors. Similar to the case for ISI, $\vec{au}$ and $\vec{tm}$ subvectors were also obtained. Each article appeared during a given month and year, so that information was recorded in the $\vec{bi}$ subvector. Finally, as mentioned earlier, many articles were assigned one or more controlled vocabulary-type categories, taken from the *Computing Reviews* system, so $\vec{cr}$ subvectors were also present. Thus, the CACM collection used seven different concept types, including ones based on textual terms (tm), ones of factual information (au,bi), ones derived from bibliographic references (bc, cc, and ln), and one based on indexer interpretation (cr).

Now that the multiple concept type model has been explained, it is appropriate to discuss its usefulness for clustering and cluster search (Chapter 7) and for enhanced feedback (Chapter 8).

CHAPTER 7

CLUSTERING OF EXTENDED VECTORS

A new model using extended vectors for the representation of document content was described in the last chapter. A key question to consider when evaluating the value of any such proposal is how it can be used to enhance retrieval performance. One such scheme, namely clustering and clustered searching, is described in this chapter. A second, relevance feedback, is elaborated upon in the next chapter.

Given a user query, typically just containing terms from the natural language text of that query, and documents that are extended to contain other information besides terms, the question is how performance of an initial search can benefit from the extended document vectors. If one accepts the cluster hypothesis [Van Rijsbergen 1974], namely that closely associated documents tend to be relevant to the same requests then an obvious answer is to use not only terms but also the latent information in other components of extended document vectors to effect a superior clustering of all collection documents. Typically, a query just containing terms will retrieve clusters containing documents whose terms match its terms. The query will probably also retrieve from those chosen clusters documents which have little in common with the query terms but are highly correlated through other components of the extended vectors with already retrieved items in the clusters. Hopefully, these will be mostly relevant documents, leading to improvements in both recall and precision.

Not only is clustering of interest in retrieval, but it also has many applications for classification. When a new scheme is available for computing document-document

similarity or for clustering a collection of documents using those values, it is interesting to examine the resulting clusters, see why they were formed, judge if the constituent members fit well together, and decide if the clustering outcome seems reasonable. In the current case, where various combinations of the components of the extended vectors are possible, some such analysis of the results of various clustering trials seems particularly appropriate.

Now that the main concepts to be explored in this chapter have been mentioned it is appropriate to outline the method of presentation chosen. The first section focuses on previous work: clustering in general, clustering based on the terms of documents, and information retrieval clustering incorporating bibliographic data. Next, to illustrate the clustering results, two subcollections of CACM documents were selected and various clustering runs made and reported upon. Finally, clustering runs on the entire CACM collection are described, illustrating the effects on retrieval of using extended document vectors in various ways.

7.1. Previous Work

7.1.1. General Clustering

Numerous books have been written about taxonomic classification and clustering theory, methods, algorithms, and practice (e.g., [Sneath & Sokal 1973], [Hartigan 1975]). The reader is encouraged to consult them or relevant reviews and summaries (e.g., [Jardine & Van Rijsbergen 1971]) for further background information.

7.1.2. Clustering for Information Retrieval

7.1.2.1. Number of Attributes

According to the vector space model a document is characterized by a large number of attributes. If there are T word stems there will often be at least T attributes. In a collection such as that of the 3204 CACM articles, $T = 10{,}446$. When extended vectors are used then the co-citation subvector will have N attributes, where N is the number of articles in the collection. Thus, for a homogeneous collection of one million articles a term based clustering might require perhaps $T = 100{,}000$ attributes while an extended vector clusterings might employ several million.

It is this problem of having a large number of attributes that suggests careful study of how various algorithms could be applied to extended vectors. Keyword classifications have similar problems with the number of attributes.

7.1.2.2. Keyword Classifications

In order to help construct thesauri and synonym dictionaries, the terms themselves can be clustered. The underlying assumption is that similar terms, as far as searching needs are concerned, tend to appear in the same documents. Sparck Jones [1971] did an early empirical study of keyword classifications and tested various hypotheses.

Since there were so many techniques and parameters to consider, she focused on methods aimed as producing stable, well defined classifications. That is, minor additions or changes in the data should give minor changes in the final results and regardless of the implementation or processing a given set of inputs should yield a unique result. Often, such approaches are called graph theoretic since one considers each term as a point and decides

on the requirements and organization of links between those points.

Typically, one computes the pairwise similarity between each of the T term vectors and arrives at a similarity matrix with values for the $T \cdot (T-1)/2$ different pairs. Unfortunately, this step requires $O(T^2)$ space and time (i.e., as T increases, the time to compute the matrix and the space required to store it go up with the square of T). Thus even before one tries to classify the terms a significant amount of space and time is required.

Given the similarity matrix among terms one can apply a threshold value and decide which term pairs are sufficiently similar to be connected in the graph representation. Thesaurus categories can then be identified by looking for appropriate groupings: cliques, clumps, stars, chains, etc. Sparck Jones found that a number of different methods gave similar results when appropriate parameter settings were chosen for each.

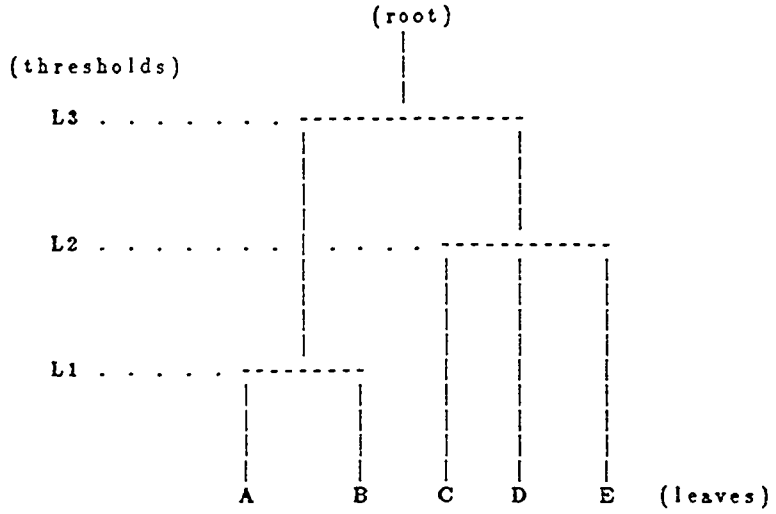
7.1.2.3. Single Link Document Classifications

Jardine and Van Rijsbergen [1971] review some of the early clustering work for retrieval purposes and describe some initial work in hierarchical clustering using the theoretically pleasing single link method. After a similarity matrix has been produced a unique hierarchical clustering results which can be displayed as a dendogram such as that shown in Figure 7.1. With threshold value L1, only A and B are linked, while with the lower threshold value L2, C,D, and E are also connected – i.e., pairwise similarities with value greater than L2 exist among at least two of the three possible pairs: C-D, C-E, and D-E.

The single link method has the following properties:

(1) Results are independent of the order of processing.

Figure 7.1: Dendogram from Hierarchical Clustering



(2) The rank ordering of the similarity values is all that is needed.

(3) All and only those documents with sufficient pairwise similarity are linked at a given level.

(4) The method is stable in terms of the effect of small errors in similarity and in terms of minimizing alterations as documents are added.

Unfortunately, the number of levels, tree size and shape, and branching ratio between nodes at various levels, are all uncontrolled. Thus, the number and average size of bottom level clusters are not known in advance and tight bounds on the cost of a depth first search of the resulting cluster tree are not obtainable.

Using the Cranfield database, Jardine and Van Rijsbergen performed experiments in clustering and searching. They had trouble with "aberrant individuals", documents not closely associated with any others – these entered the tree towards the top with low similarity levels and might best be assigned to the nearest cluster or separated into a separate "garbage" cluster. Nevertheless, the search results were promising especially when the evaluation method emphasized precision instead of recall. A simple downward search, descending the tree from the root to the leaves, selecting the most promising branch at each point, did very well.

Van Rijsbergen [1974] performed further tests of the single link clustering technique using three different collections, and found cluster based strategies to be "quite competitive". Searching was done either according to a narrow scheme like that of a depth first search or a newly proposed broad scheme: to retrieve more documents leading to possibly higher recall the search procedure back tracks to find alternate paths after pursuing the most promising ones initially. Broad searching is often more effective than narrow searching especially when recall is emphasized.

Van Rijsbergen and Croft [1975] repeated and extended previous tests using a larger collection of 1400 Cranfield documents. Another alternative to the narrow and broad search procedures was proposed, namely "bottom up", where an initial source document, for example, identifies a low level centroid. Searching would consider that centroid first and then proceed by going up the tree and then descending downward again to other hopefully useful adjoining clusters.

To improve upon the efficiency of single link clustering, Croft [1977] utilized the fact that prior to clustering both a document and an inverted file are typically present. In a

sense, the inverted file "is a one-level overlapping clustering of the documents". By using it, by ignoring occurrences of high frequency terms, and by only computing similarity values for document pairs as needed, one can substantially reduce the storage and space requirements for single link clustering. Though an upper bound on complexity is still $O(N^2)$ for N documents, the fact that few document pairs have non-trivial similarity implies that one need only compute similarities between documents that occur in the same inverted list entry. If, for example, one restricts attention from T down to T' lists containing no more than p document postings each, then even if there is complete overlap among the the T' lists the computation required is $O(T' \cdot p \cdot (p-1)/2)$. Croft found that only 5.9% of the full similarity matrix actually had to be computed in his experiment.

In a later study aimed at reducing storage requirements and at improving search efficiency Croft [1978] proposed a specialized file organization where cluster representatives can be computed dynamically, thereby saving substantially on storage overhead. By restricting attention to cluster schemes utilizing bottom up searching, Croft only required an inverted file to provide access points to the low level clusters and a slightly augmented document file.

More recently, Croft [1980] focused on probabilistic methods for computing similarity between queries and documents and was further able to improve upon the effectiveness of a bottom up search. All in all, then, specialized single-link clustering, bottom up searching, compact representations, and probabilistic similarity computation techniques can greatly aid the efficiency and effectiveness of single link methods as applied to information retrieval.

7.1.2.4. Other Clustering Approaches

Clustering has been proposed for use in retrieval by many people for various purposes. Thus, Preece [1974] proposed a negative cluster hypothesis, namely that "false drops are likely to be closer to each other than to the relevant documents". He suggested that after retrieval one has eliminated many non-relevant documents and can cluster the selected documents, possibly just by considering terms in the query, in a very efficient fashion.

Williamson [1974] developed an effective and highly efficient clustering and search system employing heuristic methods to construct a cluster tree in time $O(n \cdot \log n)$. Though lacking the theoretically pleasing qualities of the single link method his algorithm seemed to be stable and yield good results as attested by a number of experimental trials. The resulting tree is easily controlled by parameter settings to obtain a desired degree of overlap, a limitation of cluster size at each level of the tree, and a specific constraint on the branching ratios. Since the tree is built by adding documents one by one, collection growth is easily accommodated. Finally, when addition to a given node forces that node to exceed the allowable size then it is split apart only after computing the full similarity matrix for all of its children – a technique that enables reliable control and yields a good overall end-product. Williamson's algorithm served as the basis for the method employed to cluster extended vectors as described in Section 7.2.

Using a modified version of Williamson's algorithm, Salton and Wu [1981] compared the efficiency and effectiveness of various clustering and search procedures. Specifically, they contrasted inverted file searching, top down clustered file searching, and bottom up cluster searching based on an inversion of data in the low level cluster centroids. Inversion on document vectors gave the best results when short Boolean type queries were processed

while a top down tree search seemed best for long vector type queries. The low level centroid inversion, however, required the smallest storage overhead of any of the approaches.

Since a realistic technique yielding reasonably good results and having $O(n \cdot \log n)$ efficiency was required, the modified Williamson algorithm was selected for experimental use. Given that operational use would call for long extended vectors, vector type queries, and feedback queries of even greater length, the approach selected seemed particularly appropriate. With the aid of a Cornell graduate student, Robert Harper, the clustering program was implemented under UNIX and clustered search procedures were integrated into the rest of the SMART system.

7.1.3. Clustering with Bibliographic Data

In Chapter 6, the extended vector model was introduced where various types of bibliographic data were utilized to supplement the standard term vectors. Now attention will be focused on clustering studies considering bibliographic data instead of or in addition to terms.

Clustering can be performed with various purposes in mind. For preliminary investigations of new clustering methods, the aim may be simply to show that a new technique is viable. Ultimately, however, the goal is often to effect a classification of articles ([Kessler 1962, 1963a, 1963b, 1965b], [Schiminovich 1971], [Small 1973, 1974, 1978, 1980, 1981], [Kwok 1975], [Salton & Bergmark 1977, 1979]) or journals ([Carpenter & Narin 1973], [Small & Koenig 1977], [Arms & Arms 1978]). In recent years there has been increased interest in clustering to help chart the interrelationships and developments in specific areas of the sciences and social sciences ([Small 1973, 1974, 1978, 1980, 1981], [Salton & Bergmark 1977, 1979]).

Clustering can also result in a document organization that aids retrieval. Having previously attempted clustering with bibliographic coupling data, Schiminovich [1971] developed what was termed a "pattern discovery algorithm" to directly utilize links between documents. Afterwards, Bichteler and Parsons [1974] modified that method for document retrieval and found that it produced results comparable with those of standard subject indexing.

Salton did several preliminary studies of the usefulness of bibliographic information in retrieval. In [Salton 1963] he clustered documents using several schemes and found that when terms and references were employed the results were better than if only terms were considered. Later, Bichteler and Eaton [1980] demonstrated that for retrieval purposes using a similarity formula combining bibliographic coupling and co-citations was better than if bibliographic coupling alone was included. And, though on a small scale, they did do a certain amount of grouping of documents based on the resulting combined similarity values.

In summary, bibliographic data does seem valuable in a number of applications. Apparently, better clustering can be expected when that data is considered. Hence it seems appropriate to consider algorithms for clustering that allow inclusion of the various components of an extended vector representation.

7.2. Algorithms

Algorithms used are based in general on the work of Williamson as described in his Ph.D. thesis [1974]. The procedures were later adapted for experiments requiring control over the degree of overlap between low level clusters and for comparisons with other

storage and retrieval schemes [Salton & Wu 1981]. For the retrieval runs described later further modifications were made to handle the requirements of clustering extended vectors.

7.2.1. Clustering

The algorithm produces a hierarchical clustering where all N documents in a collection end up as leaves of a multilevel tree. Interior nodes are associated with cluster centroids which represent all the documents in the subtree below them. Viewed another way, a given centroid summarizes all the information contained in the children immediately below regardless of whether those are documents or other centroids.

Clustering proceeds by adding documents one by one starting with an initially empty tree. The addition process involves a search for the proper place to insert the new document and a subsequent adjustment of the tree to first include the new entry and secondly conform to the various constraints enforced during the build operation. In particular, adding a document may require a low level node or "twig" to be split and splitting may be recursively repeated all the way up to the root. This way the tree stays relatively balanced and all documents are the same distance from the root.

Table 7.1 gives specific parameters required to handle clustering of extended vectors. The first three values indicate choices specifying how the overall similarity between documents can be determined based on available subvectors – relative weighting method, similarity function used, and whether real valued weights are allowed. The last two parameters relate to special processing when a centroid subvector gets too long and must be shortened to fit into available space.

Table 7.1: Combined Retrieval Parameters for Each Concept Type

similarity coefficient ≡ coefficient used for a given concept type before adding it to arrive at overall similarity, based on formula:

$$\text{combined similarity} = \sum_{\text{all types } t} coeff_t \cdot sim_t$$

similarity computation method ≡ specification of function to compute similarity: cos correlation, inner product, normalized inner product (i.e., divided by sum of vector values)

weighting method ≡ use binary or real values

maximum subvector length ≡ length of this subvector that must not be exceeded; if it is, then low frequency values in the subvector are deleted to shorten it to within bounds

subvector deletion frequency: initial value and increment ≡ when subvector must be shortened, all entries below the initial value are deleted, and for subsequent deletions the increment is added to the cutoff previously used

7.2.2. Searching

The cluster search method utilized was a specially modified version of that proposed by Williamson. Its uniqueness relates to the handling of extended vectors and the method of implementing a broad top down search. Extended vectors are handled using the same similarity computation scheme as for clustering (see top three entries of Table 7.1). Conducting a broad search was controlled by special processing of similarities as additions are made to a heap of centroids and documents that are, respectively, yet to be expanded or yet to be presented to a user.

7.3. Small Collection Clustering Examples

To illustrate the behavior of the clustering algorithm described in Section 7.2.1 two different small subcollections from the CACM collection were required. Selection procedures and clustering results are described below.

7.3.1. Clustering the Last 55 Documents

The CACM collection has 3204 articles. For some initial testing of the clustering algorithm articles 3150-3204 were employed. Articles 3150-3183 were the most recent ones in the collection – all appeared in issues during 1979. Articles 3184-3204 came from earlier years and so their inclusion here gives greater diversity than would be expected if only a thin time slice was considered.

7.3.1.1. Input Data

The raw data for clustering is the complete extended vector for each of the 55 articles. Since the vectors themselves are not especially enlightening, it seems more sensible to provide more useful background information.

Table 7.2 gives the article number (did), year of publication, volume, number, title, and authors of each document being considered. For indexing purposes the titles were supplemented by abstracts and author assigned keywords but the information shown here should be adequate description for a reader familiar with the computer science literature.

Table 7.3 gives the occurrence information of *Computing Review* category concept numbers for articles which have $\vec{cr}$ subvectors. They are short and have some overlap so they provide a good example of what is involved in one component of the extended representations.

Table 7.2: Doc. Number, Year, Vol., No., Title, Author for Last 55 Articles

Did	Yr	Vo	No	Title (first part)	Author (first)
3150	79	07	01	Beyond Programming Languages	Winograd, T.
3151	79	07	02	An Optimal Real-Time Algorithm for Plana	Preparata, F.P.
3152	79	07	03	Storage Reorganization Techniques for Ma	Fischer, P.C.
3153	79	07	04	The Control of Response Times in Multi-C	Hine, J.H.
3154	79	07	05	Algorithm = Logic + Control	Kowalski, R.
3155	79	08	01	The Paradigms of Programming	Floyd, R.W.
3156	79	08	02	Computing Connected Components on Parall	Hirschberg, D.S.
3157	79	08	03	Proving Termination with Multiset Orderi	Dershowitz, N.
3158	79	08	04	Secure Personal Computing in an Insecure	Denning, D.E.
3159	79	08	05	Further Remark on Stably Updating Mean a	Nelson, L.S.
3160	79	09	01	Rejuvenating Experimental Computer Scien	Feldman, J.A.
3161	79	09	02	An ACM Executive Committee Position on t	McCracken, D.D.
3162	79	09	03	On Improving the Worst Case Running Time	Galil, Z.
3163	79	09	04	An Optimal Insertion Algorithm for One-S	Raiha,K.J.
3164	79	09	05	Progressive Acyclic Digraphs-A Tool for	Hansen, W.J.
3165	79	09	06	Approximation of Polygonal Maps by Cellu	Nagy, G.
3166	79	09	07	Computing Standard Deviations- Accuracy	Chan, T.F.
3167	79	09	08	Updating Mean and Variance Estimates- An	West, D.H.D.
3168	79	10	01	Comment on "An Optimal Evaluation of Boo	Laird, P.D.
3169	79	10	02	Note on "An Optimal Evaluation of Boolea	Gudes, E.
3170	79	10	03	On the Proof of Correctness of a Calenda	Lamport, L.
3171	79	10	04	Line Numbers Made Cheap	Klint, P.
3172	79	10	05	An Algorithm for Planning Collision-Free	Lozano-Perez, T.
3173	79	11	01	A Psychology of Learning BASIC	Mayer, R.E.
3174	79	11	02	Password Security- A Case History	Morris, R.
3175	79	11	03	Breaking Substitution Ciphers Using a Re	Peleg, S.
3176	79	11	04	Storing a Sparse Table	Tarjan, R.E.
3177	79	11	05	How to Share a Secret	Shamir, A.
3178	79	12	01	Introduction to the EFT Symposium	Kling, R.
3179	79	12	02	Overview of the EFT Symposium	Kraemer, K.L.
3180	79	12	03	Costs of the Current U.S. Payments Syste	Lipis, A.H.
3181	79	12	04	Public Protection and Education with EFT	Long, R.H.
3182	79	12	05	Vulnerabilities of EFTs to Intentionally	Parker, D.B.
3183	79	12	06	Policy, Values, and EFT Research- Anatom	Kraemer, K.L.

(Continued on Next Page)

Table 7.2 cont'd: Doc. No., Year, Vol., No., Title, Author for Last 55 Articles

Did	Yr	Vo	No	Title (first part)	Author (first)
3184	63	01	17	Revised Report on the Algorithmic Langua	Naur, P.
3185	72	10	10	The Humble Programmer	Dijkstra, E. W.
3186	68	03	03	GO TO Statement Considered Harmful	Dijkstra, E. W.
3187	66	05	16	Certification of Algorithm 271 (QUICKERS	Blair, C. R.
3188	66	03	19	Semiotics and Programming Languages	Zemanek, H.
3189	62	11	24	An Algebraic Compiler for the FORTRAN As	Stiegler, A. D.
3190	67	02	14	Correction to Economies of Scale and the	Solomon, M.B.
3191	68	06	17	Generating Permutations by Nested Cyclin	Langdon, Glen G.
3192	58	07	02	The Lincoln Keyboard - a Typewriter Keyb	Vanderburgh, A.
3193	58	07	03	MANIAC II	
3194	59	01	02	A Non-heuristic Program for Proving Elem	Dunham, B.
3195	62	11	23	Reiteration of ACM Policy Toward Standar	Gorn, S.
3196	63	01	18	The Reactive Typewriter Program	Mooers, C. N.
3197	63	06	26	Structures of Standards-Processing Organ	Gorn, S.
3198	66	03	18	Microprogramming, Emulators and Programm	Greem, J.
3199	66	08	13	ALGEM - An Algebraic Manipulator	Gotlieb, C. C.
3200	66	08	14	A FORMAC Program for the Solution of Lin	Cuthill, E.
3201	66	08	15	Symbolic Manipulation of Poisson Series	Danby, J.
3202	66	08	16	MANIP- A Computer System for Algebra and	Bender, B.
3203	66	08	17	GRAD Assistant - A Program for Symbolic	Fletcher, J. G.
3204	66	08	18	An On-Line Program for Non-Numerical Alg	Korsvold, K.

Table 7.3: $\vec{CR}$ Subvector Information for Last 55 Articles

Doc. Id.	List of $\vec{cr}$ Category Concept Numbers
3150	105 113 115 127
3151	132 157 164
3152	123 145 157
3153	121 196
3154	85 113 119 153 156
3156	157 164 181
3157	156 174
3158	18 179
3159	150 171
3162	94 127 157
3163	93 94 123 157 163
3164	122 123 164
3165	38 123 198
3166	142 150 171
3167	150 171
3168	74 93 94
3169	70 90 94
3170	156
3171	109 110 113 129
3172	39 85 87 196
3173	7 59 115
3174	24 124
3175	65 84
3176	94 109 123 157
3177	165 173
3179	17 72 73 98
3180	72
3181	18
3182	17 21 72 98
3183	17 21 72 73 98
3186	115 155 156
3191	165

7.3.1.2. Parameters

Regarding combined similarity calculations and limitations on the various subvectors, the values given in Table 7.4 are those selected. Note that very preliminary notions about proper similarity coefficients and subvector lengths provided the basis for each setting. Thus, the similarity coefficient reflected the assumed value of the subvector, and the maximum on number of concepts related to the assumed length of the vector component for a given concept type.

Table 7.4: Subvector Specific Parameter Values

concept type abbrev.	similarity coefficient value	term freq. cutoff for deletions	max. no. of concepts in subvector
au	1.00	1	100
bi	0.15	10	20
cr	0.30	2	50
tm	0.30	1	800
bc	1.00	1	200
ln	0.80	1	100
cc	1.00	1	200

7.3.1.3. Clustering Process

Initially, documents are added to the single root cluster until the splitting limit of 20 is reached. The articles 3150-3169 are split to give the tree shown in Figure 7.2. It should be noted that node 4 has 11 children – it is the "garbage" or "orphan" cluster containing all documents that do not easily group together.

More documents are added to the tree shown in Figure 7.2 until splitting of node 0 is called for. Figure 7.3 shows the five clusters that replace node 0. Finally, the remaining documents are added to the existing tree. Table 7.5 shows the final tree formed.

Figure 7.2: Tree After First Split

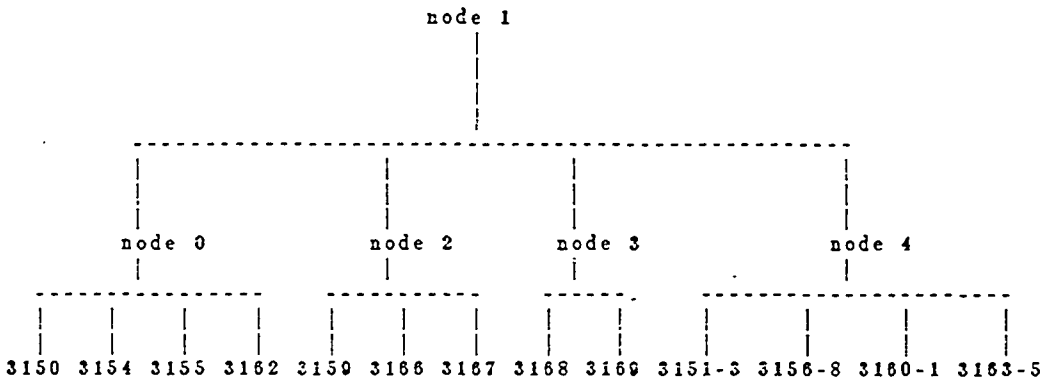


Figure 7.3: Result of Splitting Node 0

node 1 is root and parent for nodes below:

node 0	node 5	node 6	node 7	node 8
3150,3154-5, 3162,3170-2	3185 3186	3192,3193, 3196,3199	3189,3195, 3197	3184,3187, 3188,3198

Table 7.5: Final Tree for Documents 3150-3204

node 1 is root and parent for nodes below:

node	list of documents below
0	3150,3154,3155,3162,3170,3171,3172
2	3159,3166,3167,3175,3178,3179,3180,3181, 3182,3183
3	3168,3169
4	3151,3152,3153,3156,3157,3158,3160,3161, 3163,3164,3165,3173,3174,3176,3177,3190, 3191,3194
5	3185,3186
6	3192,3193,3196,3199,3200,3201,3203,3204
7	3189,3195,3197
8	3184,3187,3188,3198

7.3.1.4. Labelling of Clusters

Since classification is often aimed at grouping similar items and then attaching meaningful labels to each such group, that policy is adopted for the example of this section. Accordingly Table 7.6 is a labelled version of Table 7.5.

7.3.1.5. Discussion of Results

It can be seen from Tables 7.5 and 7.6 that node 4 is overly large, containing a number of relatively unrelating groupings. As the "orphan" or "garbage" cluster, it would be better off if split further or if entries were moved to other more closely related clusters.

In general it seems clear that the cluster size is somewhat large, allowing several groups of documents to co-exist inside certain of the clusters. Instead of splitting at size 20, perhaps a limit of 10 would be more appropriate. Nevertheless, the topical separation

of clusters shown in Table 7.6 does seem to be fairly reasonable. Given this background, a more detailed study of clustering of a small CACM subcollection is in order.

Table 7.6: Labelling of Clusters for Documents 3150-3204

node 1 is root and parent for nodes below:

node	list of topics for documents below
0	Programming,Algorithms
2	Probability,Statistics,Electronic Funds Transfer
3	Boolean Expressions and Queries
4	Complexity,Systems,Experiments
5	Dijkstra's articles
6	Programs/Systems
7	Compilers,Standards
8	Programming Languages & Algorithms

7.3.2. Clustering of Highly Cited Articles

There tends to be a good correlation between important articles as judged by peers and those receiving a relatively large number of citations [Lawani & Bayer 1983]. Hence it seems reasonable to further illustrate the behavior of the extended vector clustering algorithm using articles with many citations.

7.3.2.1. Choosing Test Set

Since a detailed examination of the clusters resulting from 3204 articles is a very time consuming task it was decided to use a much smaller number of highly cited articles. The assumption made was that highly cited articles would tend to have non-null $\vec{bc}$, $\vec{cc}$, and $\vec{ln}$ subvectors due to their many bibliographic connections.

In addition to being cited, however, it was also desirable for chosen articles to be categorized according to the *Computing Reviews* scheme since contrast could then be made between category based clustering and attribute based clustering.

Accordingly, Table 7.7 gives the size for various sets with non-null $\vec{cr}$ subvectors and with varying numbers of citations. The "-" in various places represents a "don't care", i.e., that any value is acceptable. For example, there are 1424 with $\vec{cr}$ subvector, 1161 with at least one citation, and 254 with at least two citations and with $\vec{cr}$ subvectors.

Table 7.7: Set Sizes with Citations and $\vec{cr}$ Subvector

No. of Citations	$\vec{cr}$ Subvector?	Set Size
-	Y	1424
≥ 1	-	1161
≥ 1	Y	549
≥ 2	-	560
≥ 2	Y	254
≥ 5	-	117
≥ 5	Y	52

It can be seen that if one considers only those CACM articles which have a non-null $\vec{cr}$ subvector and which have received at least five citations then one arrives at a set H of 52 highly cited articles. The set H is what is used in subsequent clustering tests. For ease of reference, the articles in set H are listed in Table 7.8, ordered by ascending document identifier number.

Table 7.8: Document Number and Title for 52 Selected in *H*

1684 Ambiguity in Limited Entry Decision Tables
1728 Further Experimental Data on the Behavior of Programs in a Paging ...
1741 BRAD: The Brookhaven Raster Display
1746 Protection in an Information Processing Utility
1749 The Structure of the "THE"-Multiprogramming System
1751 The Working Set Model for Program Behavior
1754 Dynamic Storage Allocation Systems
1771 CURRICULUM 68 -- Recommendations for Academic Programs in Computer ...
1781 Translator Writing systems
1785 Scatter Storage Techniques
1786 An Improved Hash Code for Scatter Storage
1826 A LISP Garbage-Collector for Virtual-Memory Computer Systems
1834 An Axiomatic Basis for Computer Programming
1877 Prevention of System Deadlocks
1879 A Note on Storage Fragmentation and Program Segmentation
1901 Dynamic Space-Sharing in Computer Systems
1936 Variable Length Tree Structures Having Minimum Average Search Time
1947 Object code Optimization
1972 A Nonrecursive List Compacting Algorithm
1973 The Linear Quotient Hash Code
1976 Multi-attribute Retrieval with Combined Indexes
1989 Transition Network Grammars for Natural Language Analysis
2046 A Relational Model of Data for Large Shared Data Banks
2053 On the Conversion of Decision Tables to Computer Programs
2060 GEDANKEN-A Simple Typeless Language Based on the Principle of ...
2080 The Nucleus of a Multiprogramming System
2107 The Quadratic Quotient Method: A Hash Code Eliminating Secondary ...
2109 The Use of Quadratic Residue Research
2110 An Efficient Context-free Parsing Algorithm
2111 Spelling Correction in Systems Programs
2138 BLISS: A Language for Systems Programming
2150 Concurrent Control with "Readers" and "Writers"
2203 Key-to-Address Transform Techniques: A Fundamental Performance ...
2204 Program Development by Stepwise Refinement
2220 Conversion of Limited-Entry Decision Tables to Computer Programs ...
2247 On the Criteria To Be Used in Decomposing Systems into Modules
2345 Curriculum Recommendations for Graduate Professional Programs in ...
2356 A Technique for Software Module Specification with Examples
2373 Properties of the Working-Set Model
2435 A Class of Dynamic Memory Allocation Algorithms
2438 A Model and Stack Implementation of Multiple Environments
2569 Computer Generation of Gamma Random Variates with Non-integral ...
2597 Monitors: An Operating System Structuring Concept
(Continued on Next Page)

Table 7.8 cont'd: Document Number and Title for 52 Selected in H

2629 The UNIX Time-Sharing System
2632 HYDRA: The Kernel of a Multiprocessor Operating System
2723 Multiprocessing Compactifying Garbage Collection
2732 Guarded Commands, Nondeterminacy and Formal Derivation of Programs
2751 Illumination for Computer Generated Pictures
2767 A Comparison of Simulation Event List Algorithms
2839 An Insertion Technique for One-Sided Height-Balanced Trees
3076 Value Conflicts and Social Choice in Electronic Funds Transfer ...
3186 GO TO Statement Considered Harmful

7.3.2.2. Clustering using CR Categories

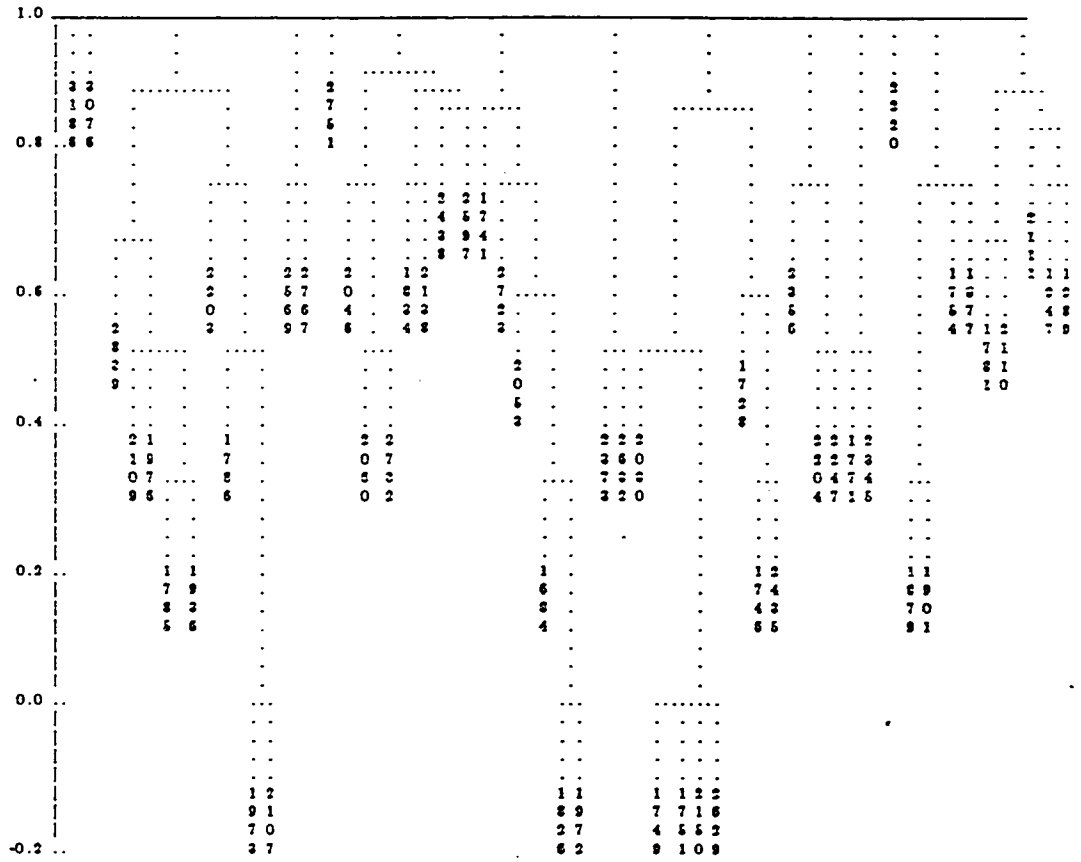
Since the $\vec{cr}$ subvectors had maximum length around 200, it is possible to use packaged routines for performing single link clustering and multidimensional scaling of the 52 chosen vectors. In addition, the fast heuristic clustering method could be utilized.

First, consider single link clustering where the input data is a binary vector of $\vec{cr}$ entries for each of the 52 articles. Based on computed pairwise distances between articles, Figure 7.4 gives the dendogram produced using the "S" statistical analysis package [Becker & Chambers 1981]. It is laid out like the example shown earlier in Figure 7.1.

The four digit document numbers are read from top to bottom, and similar documents are found near to each other. Horizontal lines show at what level the documents or clusters are connected. The vertical axis indicates the Euclidean distance between linked items, based on comparisons of the binary $\vec{cr}$ subvectors.

Articles linked together towards the bottom of the diagram are highly similar and so such clusters are well chosen. When the distance is more than about 0.5, however, the groupings seem rather arbitrary. Perhaps this explains why bottom up or broad searching

Figure 7.4: Single Link Clustering Based on $\vec{cr}$ Subvectors



of single link clustering output were found to be more effective by Van Rijsbergen and Croft [1975] than narrow top down methods.

As a second approach to clustering, consider multidimensional scaling. Using the appropriate "S" routine, the $\vec{cr}$ subvectors, with some 200 possible binary valued entries can be utilized to yield a two dimensional alternate representation that preserves many of the relative pairwise distances between documents. Figure 7.5 shows how the 52 documents of interest can be represented in a two dimensional map. Many of the points representing documents are labelled nearby with the corresponding document identifier numbers and topical areas are indicated in a number of places. The dense central region focuses on programming languages which is apparently a crucial component of CACM literature. As one moves up and to the right the emphasis changes to run time environments, multiprogramming, and finally operating systems. Moving instead down and to the right, subjects include database methods, simulation, hashing for scatter storage, tree structures, and other storage methods. This two dimensional portrayal makes the variation of subjects fairly easy to grasp.

To obtain a one dimensional listing that best captures the $\vec{cr}$ clustering based on multidimensional scaling one reduces the dimensionality to one instead of two. Variations and distances between documents are compressed to a single real value and one can then order the documents based on that value. Table 7.9 gives the resulting listing of documents. The first column gives the scaling value, while the second indicates the cluster number (C) derived from that scaling. The grouping seems to correspond to the two dimensional result in that the given ordering would result if one traversed Figure 7.5 from bottom right to left center and then back up to top right. In other words, the documents seen along that path

Figure 7.5: Two Dimensional Scaling Result

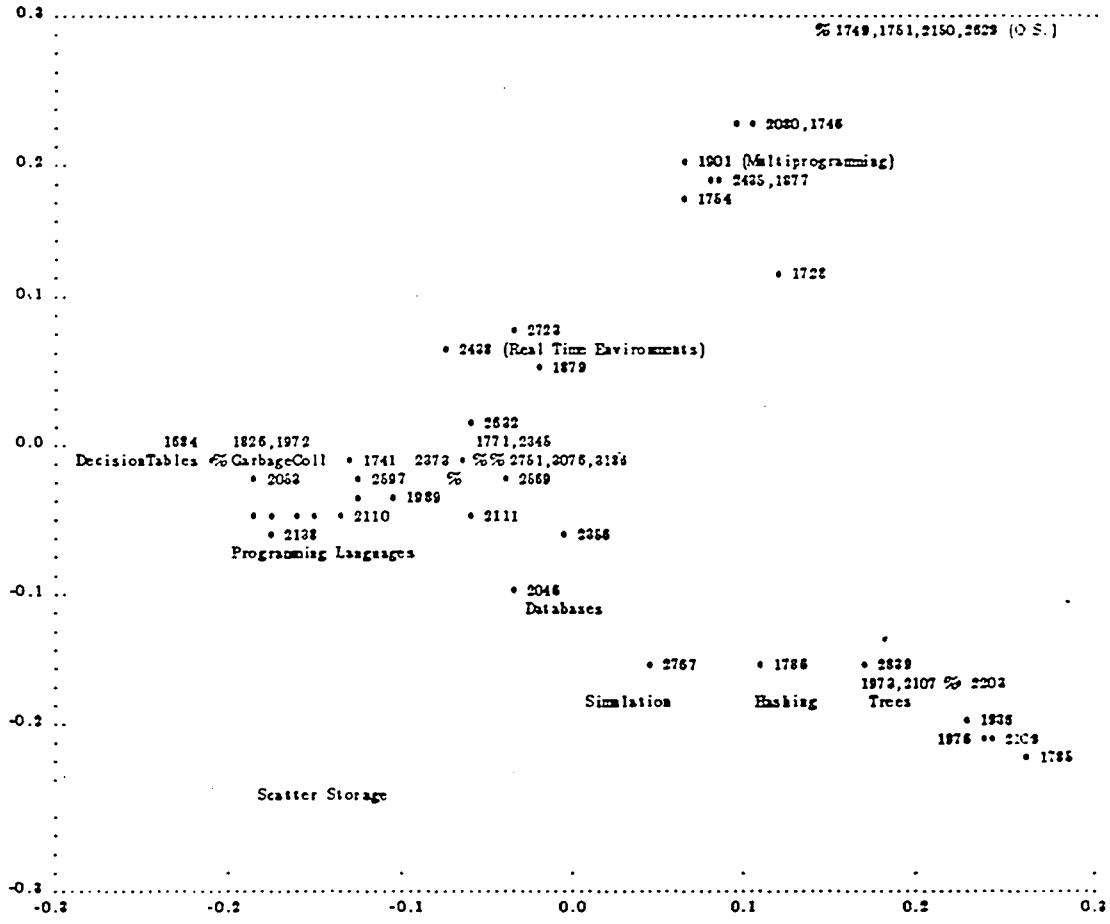


Table 7.9: One Dimensional Scaling Result

ScaleVal. C Did Title

-0.220835 1 1785 Scatter Storage Techniques
-0.207128 1 1976 Multi-attribute Retrieval with Combined Indexes
-0.204346 1 2109 The Use of Quadratic Residue Research
-0.200938 1 1936 Variable Length Tree Structures Having Minimum Averag...
-0.175750 1 2107 The Quadratic Quotient Method: A Hash Code Eliminatin...
-0.175750 1 1973 The Linear Quotient Hash Code
-0.175224 1 2203 Key-to-Address Transform Techniques: A Fundamental Pe...
-0.158407 1 2839 An Insertion Technique for One-Sided Height-Balanced ...
-0.155781 1 1786 An Improved Hash Code for Scatter Storage
-0.150897 1 2767 A Comparison of Simulation Event List Algorithms
-0.098880 2 2046 A Relational Model of Data for Large Shared Data Banks
-0.066542 2 2138 BLISS: A Language for Systems Programming
-0.059449 2 2356 A Technique for Software Module Specification with Ex...
-0.057820 2 1781 Translator Writing systems
-0.054351 2 2111 Spelling Correction in Systems Programs
-0.054196 2 2732 Guarded Commands, Nondeterminacy and Formal Derivatio...
-0.053493 2 2060 GEDANKEN-A Simple Typeless Language Based on the Prin...
-0.047077 2 2110 An Efficient Context-free Parsing Algorithm
-0.045092 2 1834 An Axiomatic Basis for Computer Programming
-0.043457 2 1989 Transition Network Grammars for Natural Language Anal...
-0.041092 2 1947 Object code Optimization
-0.031364 3 2597 Monitors: An Operating System Structuring Concept
-0.025255 3 2569 Computer Generation of Gamma Random Variates with Non...
-0.024553 3 2247 On the Criteria To Be Used in Decomposing Systems int...
-0.024159 3 2220 Conversion of Limited-Entry Decision Tables to Comput...
-0.023143 3 2204 Program Development by Stepwise Refinement
-0.019345 3 2053 On the Conversion of Decision Tables to Computer Prog...
-0.014721 4 1771 CURRICULUM 68 -- Recommendations for Academic Program...
-0.014721 4 2345 Curriculum Recommendations for Graduate Professional ...
-0.013667 4 1684 Ambiguity in Limited Entry Decision Tables
-0.012606 4 3186 GO TO Statement Considered Harmful
-0.012605 4 2751 Illumination for Computer Generated Pictures
-0.012605 4 3076 Value Conflicts and Social Choice in Electronic Funds...
-0.012234 4 1741 BRAD: The Brookhaven Raster Display
-0.010543 4 1972 A Nonrecursive List Compacting Algorithm
-0.010542 4 1826 A LISP Garbage-Collector for Virtual-Memory Computer ...
-0.006661 4 2373 Properties of the Working-Set Model
 0.012503 4 2632 HYDRA: The Kernel of a Multiprocessor Operating System

(Continued on Next Page)

Table 7.9 continued: One Dimensional Scaling Result

ScaleVal. C Did Title

+ 0.055680 5 1879 A Note on Storage Fragmentation and Program Segmentation
+ 0.070841 5 2438 A Model and Stack Implementation of Multiple Environments
+ 0.079979 5 2723 Multiprocessing Compactifying Garbage Collection
+ 0.125338 6 1728 Further Experimental Data on the Behavior of Programs ...
+ 0.167427 6 1754 Dynamic Storage Allocation Systems
+ 0.185266 6 1877 Prevention of System Deadlocks
+ 0.186621 6 2435 A Class of Dynamic Memory Allocation Algorithms
+ 0.197273 6 1901 Dynamic Space-Sharing in Computer Systems
+ 0.219584 6 1746 Protection in an Information Processing Utility
+ 0.228533 6 2080 The Nucleus of a Multiprogramming System
+ 0.296548 7 2629 The UNIX Time-Sharing system
+ 0.296548 7 1751 The Working Set Model for Program Behavior
+ 0.296549 7 2150 Concurrent Control with "Readers" and "Writers"
+ 0.296549 7 1749 The Structure of the "THE"-Multiprogramming System

---

would be roughly the same as those seen when sequencing through the entries of Table 7.9 from top to bottom.

Finally, one gets a third perspective on $\vec{cr}$ based clustering through use of the fast heuristic method described in Section 7.2.1. Table 7.10 shows the resulting tree with additional summary information. In particular, for each cluster there is a listing of the most commonly occurring *CR* categories found in documents attached to that node. Thus, one can characterize each cluster for the sake of analyzing the results by considering the categories listed. Unfortunately, no regard is made of the fact that the category system is a hierarchical one, and that categories near each other in the classification system imply that the subjects are fairly similar. For a clearer understanding of the situation the reader might wish to examine the report describing the latest revision to the *Computing Reviews* category list [Denning et al. 1981].

Table 7.10: Cluster Results Using $\vec{cr}$ Subvector

Node 1 is root of tree and parent of other nodes

Node No.	Category List	Document List
0	4.3,4.32,4.39	1728,1746,2373,2435,2438,2569,2751 2767,3076
2	4.30,4.32	1749,1751,1754,2080,2150,2597,2629
3	4.9,3.74	1728,1786,1973,2107,2356
4	4.32,6.2,6.20	1754,1877,1879,1901,2632
5	3.73,3.74,3.79	1785,1936,1976,2046,2109,2111,2203, 2732,2839
6	3.50,4.19,4.49	1684,1741,1826,1972,2053,2220,2723
7	4.0,4.12,4.22-4,5.23	1771,1781,1834,1997,1989,2060,2110, 2138,2204,2247,2375,3186

Based on the $\vec{cr}$ clustering, one can list the documents that correspond to the linearization of the cluster tree. Table 7.11 therefore gives a summary of the $\vec{cr}$ cluster results. Each document is labelled by the node number of its cluster parent (C), its document number (Did), and the title. Cluster 7 seems the most heterogeneous and cluster 0 seems next worst. Since the emphasis of the clustering is to form clusters based on high pairwise similarity values with documents added to the cluster that has most categories in common. it appears that the first documents added to each cluster largely determine its subsequent composition (at least until splitting). The initial cluster, node 0, tends to have diverse elements since it is added to at the beginning by all documents that are entered before the first splitting. The last cluster, node 7, on the other hand, probably contains all the documents that do not fit neatly into any other cluster. It should be noted that when splitting

Table 7.11: Linearized Tree from $\overline{cr}$ Clustering

C Did Title

0 1728 Further Experimental Data on the Behavior of Programs in a Paging ...
0 1746 Protection in an Information Processing Utility
0 2373 Properties of the Working-Set Model
0 2435 A Class of Dynamic Memory Allocation Algorithms
0 2438 A Model and Stack Implementation of Multiple Environments
0 2569 Computer Generation of Gamma Random Variates with Non-integral Sha...
0 2751 Illumination for Computer Generated Pictures
0 2767 A Comparison of Simulation Event List Algorithms
0 3076 Value Conflicts and Social Choice in Electronic Funds Transfer Sys...
2 1749 The Structure of the "THE"-Multiprogramming System
2 1751 The Working Set Model for Program Behavior
2 1754 Dynamic Storage Allocation Systems
2 2080 The Nucleus of a Multiprogramming System
2 2150 Concurrent Control with "Readers" and "Writers"
2 2597 Monitors: An Operating System Structuring Concept
2 2629 The UNIX Time-Sharing system
3 1728 Further Experimental Data on the Behavior of Programs in a Paging ...
3 1786 An Improved Hash Code for Scatter Storage
3 1973 The Linear Quotient Hash Code
3 2107 The Quadratic Quotient Method: A Hash Code Eliminating Secondary C...
3 2356 A Technique for Software Module Specification with Examples
4 1754 Dynamic Storage Allocation Systems
4 1877 Prevention of System Deadlocks
4 1879 A Note on Storage Fragmentation and Program Segmentation
4 1901 Dynamic Space-Sharing in Computer Systems
4 2632 HYDRA: The Kernel of a Multiprocessor Operating System
5 1785 Scatter Storage Techniques
5 1936 Variable Length Tree Structures Having Minimum Average Search Time
5 1976 Multi-attribute Retrieval with Combined Indexes
5 2046 A Relational Model of Data for Large Shared Data Banks
5 2109 The Use of Quadratic Residue Research
5 2111 Spelling Correction in Systems Programs
5 2203 Key-to-Address Transform Techniques: A Fundamental Performance Stu...
5 2732 Guarded Commands, Nondeterminacy and Formal Derivation of Programs
5 2839 An Insertion Technique for One-Sided Height-Balanced Trees
6 1684 Ambiguity in Limited Entry Decision Tables
6 1741 BRAD: The Brookhaven Raster Display
6 1826 A LISP Garbage-Collector for Virtual-Memory Computer Systems
6 1972 A Nonrecursive List Compacting Algorithm
6 2053 On the Conversion of Decision Tables to Computer Programs
6 2220 Conversion of Limited-Entry Decision Tables to Computer Programs ...
6 2723 Multiprocessing Compactifying Garbage Collection

(Continued on Next Page)

Table 7.11 continued: Linearized Tree from $\vec{cr}$ Clustering

C Did Title

7 1771 CURRICULUM 68 -- Recommendations for Academic Programs in Computer...
7 1781 Translator Writing systems
7 1834 An Axiomatic Basis for Computer Programming
7 1947 Object code Optimization
7 1989 Transition Network Grammars for Natural Language Analysis
7 2060 GEDANKEN-A Simple Typeless Language Based on the Principle of Comp...
7 2110 An Efficient Context-free Parsing Algorithm
7 2138 BLISS: A Language for Systems Programming
7 2204 Program Development by Stepwise Refinement
7 2247 On the Criteria To Be Used in Decomposing Systems into Modules
7 2345 Curriculum Recommendations for Graduate Professional Programs in I...
7 3186 GO TO Statement Considered Harmful

---

took place the average correlation was 0.057 and that after that event 4 articles were added to the cluster tree with 0.0 similarity to existing centroids (i.e., no matching terms between the new documents and previous ones).

All in all, the clustering seems reasonable for search purposes, though the classification is not as neat and orderly as that produced by multidimensional scaling. Hence it seems sensible to examine the corresponding results when other subvectors or combinations of subvectors are utilized.

7.3.2.3. Clustering Using Various Subvectors

Having seen the results of several types of clustering methods applied to the $\vec{cr}$ subvectors, it is natural to consider the results of clustering with other subvectors. In particular, the $\vec{bc}$, $\vec{cc}$, $\vec{ln}$, and $\vec{tm}$ subvectors should be used as the basis for clustering. Further, appropriate combinations such as $\vec{bc}$-$\vec{cc}$ for $\vec{bc}$ and $\vec{cc}$ together or $\vec{mx}$ for a mix of all of the types allow one to see if several different subvectors complement each other.

Though it would be interesting to use single link or multidimensional scaling for these tests, the large number of possible attributes for each subvector exceed the limits of packaged programs available for these purposes. While 200 different $\vec{cr}$ categories is manageable, some 3200 documents or 10,000 terms are far too many to allow standard matrix manipulations to take place. Hence the Williamson-like-fast heuristic clustering algorithm described in Section 7.2.1 was employed for all clusterings reported in the remainder of this subsection.

Table 7.12 summarizes the results of clustering the chosen documents using each of seven similarity formulas. Entries are for single subvector clustering using $\vec{bc}$, $\vec{cc}$, $\vec{ln}$, and $\vec{tm}$ subvectors and then of two composite schemes. Those last two are for equally weighted $\vec{bc} - \vec{cc}$ clustering and for clustering based on a mix of all components. Specifically, the mix has weighting values: authors .1, dates .05, and all the other subvectors ($\vec{bc}$, $\vec{cc}$, $\vec{cr}$, $\vec{ln}$, $\vec{tm}$) .17.

By way of analysis of results, first consider the number and size of clusters for each case. Apparently, all the single subvector cases have many documents in the last cluster, the result of having to create an orphan cluster. That is, insufficient linkage information was available to give a clean split into homogeneous clusters and so relatively unconnected documents were set aside into the last large cluster. When combined similarity computations were performed, however, there was little need for an orphan cluster. Thus, using both $\vec{bc}$ and $\vec{cc}$ subvectors covered documents coupled with others and those co-cited by others; in the chosen group of highly cited articles it seems sensible that all documents would be connected by at least one of these two types of relationships. Given that occurrence, the combined similarity that uses a mix of all concept types certainly had ade-

Table 7.12: Summary of Sizes for Clustering Schemes

Subvectors Used to Cluster	No. of Clusters	Sizes of Clusters
cr	7	9,7,5,5,9,7,12
bc	4	18,9,8,17
cc	5	12,9,7,6,18
ln	5	11,11,5,8,17
tm	5	17,3,11,11,11
bc-cc	8	9,9,10,8,2,3,3,8
mx	6	12,18,10,3,7,5

quate information to avoid having to form an orphan cluster. Perhaps almost too much data is available. The second cluster is rather large and contains documents from a number of different subject areas, possibly because of many spurious interconnections.

Secondly, a number of observations are in order based on the actual clusters formed for each case. There are only four large heterogeneous clusters based on $\vec{bc}$ subvectors. Referring back to a trace of the clustering one finds that the only split that took place used a matrix of child-child similarities with the rather low average correlation of 0.012. After that split, 12 documents were added to the tree, rather randomly, since they had 0.0 similarity to each centroid present. Apparently, then, the $\vec{bc}$ subvectors do not contain enough information on their own to connect even highly cited articles together and so the resulting clustering is not very good.

The $\vec{cc}$ clustering results seems slightly better, though there is still a rather large "orphan" cluster. Two nodes seem fairly cleanly defined but two others could be profitably

split up. Turning to the cluster trace one notes that splitting was based on an average correlation of 0.039 and that thereafter only one document was added to the tree with 0.0 similarity to existing clusters. This performance is accordingly much better than that for the $\vec{bc}$ subvector clustering.

The $\vec{ln}$ results are similar to those for $\vec{cc}$ clustering. From the clustering procedure trace the average correlation is found to be only 0.02, a rather low value, but only one document wound up being added to the tree with 0.0 similarity to clusters. Apparently, the documents are fairly well linked together but the links are not very strong. This situation adds weight to the argument that single citation connections are not always representative of significant subject matter overlap.

Clustering with $\vec{tm}$ was not particularly good. Aside from one small well defined cluster, all of the others were rather diffuse. Splitting took place with average correlation 0.034 the first time and 0.042 the second time. All document additions were based on positive similarity with at least one cluster. Note, however, that term connections are often especially based on high frequency stems and so would probably be less significant than matches of other bibliographic data elements. Furthermore, 17 documents were added to the cluster tree based on a document-centroid similarity of less than 0.1. Thus, it is not surprising that the end-product classification did not seem to have particularly high quality.

Turning now to schemes for composite similarity based clustering one observes that when $\vec{bc}$ and $\vec{cc}$ subvectors are used then a larger number of smaller clusters are formed. Each cluster seems relatively homogeneous, though several have one or two articles that do not seem to be particularly connected to the rest of the cluster. The last cluster is more heterogeneous than the others and probably contains some of the orphans that were left

over. From the cluster trace one notes that the average similarity for splitting was 0.026 the first time, 0.03 the second, and 0.029 the third. Only a single document was added to the tree with 0.0 correlation to all centroids. One might wonder why the clustering appears so good when average correlations for splitting are so low. A possible explanation is that the combined similarity computation averages the effects of both $\vec{bc}$ and $\vec{cc}$, and since it is not particularly likely that two articles will be connected by both types of relationships at the same time, the result is that combined similarities are about half of the underlying values. Perhaps if the total or the maximum functions were used instead of the average then correlations would be higher. In any case the clustering results seem promising.

The last case of the series is based on a mix of all of the subvectors. The groupings seem rather heterogeneous. Average correlation for splits are 0.048 and 0.04. No articles had to be added with 0.0 similarity to centroids. If the clustering result does not seem as good as that of other methods then a likely explanation is that improper coefficients were chosen and used in computing the combined similarity value.

Finally, now that the results of the various subvector clustering schemes have been explained a summary seems in order. Each separate subvector and two combinations of several subvectors were used for clustering and the classifications that emerged appeared to vary greatly in quality. $\vec{bc}$ gave rather poor results but the $\vec{bc} - \vec{cc}$ combination seemed rather good. $\vec{tm}$ did not do especially well, perhaps because many of the term connections are due to high frequency term matches, which convey little information. $\vec{cc}$ and $\vec{ln}$ subvectors seemed to yield comparable results, each somewhat better than $\vec{bc}$. A mixture of all types ($\vec{mx}$) was not as productive of good clusters as was the simpler $\vec{bc} - \vec{cc}$ combination; this suggests that a better composite function needs to be utilized when many concept

types are considered. All in all, these preliminary clustering tests gave a number of interesting perspectives on the overall process.

7.4. CACM Full Collection Clustering

Based on the previous preliminary tests of clustering by the modified Williamson algorithm it seemed reasonable to proceed with clustering and searching of the full collection. Since for retrieval purposes the results of a clustered search are what one uses for evaluation, attention will be given first to the search process.

7.4.1. Search Efficiency

A clustered search can be analyzed in terms of its efficiency and effectiveness: how many centroids and documents must be examined and what is the recall-precision behavior of the retrieved set. If one can ascertain parameter settings for the search process that give good performance in general then with that component of the retrieval system fixed one can try different clustering methods and determine how clustering affects retrieval.

In this subsection, then, emphasis is on search efficiency. The algorithm is that mentioned in Section 7.2.2 and yields a relatively broad top down search. By varying a few parameters one can control two characteristics of the search, namely:

(1) whether when a low level centroid is retrieved the user sees part, most, or all of its contents - since a ranking can be done and only the promising ones presented while the rest are saved for later.

(2) whether expansion of low level centroids versus other centroids in the tree are encouraged.

After a fair amount of experimentation the parameters involved were fixed. The real significance of this is that once reasonable efficiency is achieved then one can simply vary clustering schemes and get effectiveness results that enable comparisons to be made between those cluster methods.

Tables 7.13 and 7.14 summarize the efficiency statistics for the search procedure utilized in all subsequently reported experiments. Thirty two different queries were used, and for each query the system had to retrieve 30 documents. Table 7.13 serves as a key to the interpretation of data found in Table 7.14. Efficiency is measured in terms of the amount of work required to find 30 documents, i.e., by the number of low level centroids, the total number of centroids, and the number of documents examined. Since absolute numbers are often not very useful in gauging performance, percentage values are also supplied.

From Table 7.14, one can make a number of observations:

(1) In all cases, less than 5% of the documents are considered. Usually only 2% are visited. Note that exactly 30 documents are shown to the user, but actually more documents are retrieved in the sense that the clusters containing them are considered. Many of these documents, however, have lower query-document similarity than centroids and so are not presented to the user but rather saved in the search heap.

(2) There is a wide spread in terms of the percentage of centroids that are examined – anywhere from about 5% to 21%.

(3) The percentage of low level centroids considered vary from 3% to almost 18%.

(4) In the worst case observed slightly over 200 centroids and documents are examined. This apparent bound on computational cost does not seem unreasonable.

Table 7.13: Definitions Pertaining to Efficiency Statistics Given in Table 7.14 for CACM Clustering

Explanation of Columns:

Column Label	Explanation of Column
query id. no.	identifying number for query selected
no. of cents.	the actual number of centroids examined during the course of the search
% of all cents.	percentage of the total number of centroids that were examined during search
no. of l.l. cents.	number of low level centroids examined during course of the search
% of l.l. cents.	percent of total number for above
no. of all docs.	number of docs. in all low level centroids (only ones with high sim. are in 30 chosen)
% of all docs.	percent of total number for above

Table 7.14: Efficiency Statistics for CACM Search Returning 30 Documents

query id. no.	no. of cents.	% of all cents.	no. of l.l. cents.	% of l.l. cents	no. of of docs.	% of all docs.
1	52	10.970	39	9.220	65	2.029
2	72	15.190	55	13.002	144	4.494
3	47	9.916	30	7.092	68	2.122
4	36	7.595	23	5.437	65	2.029
5	50	10.549	37	8.747	71	2.216
6	41	8.650	23	5.437	100	3.121
7	29	6.118	16	3.783	64	1.998
8	52	10.970	39	9.220	87	2.715
9	42	8.861	29	6.856	93	2.903
10	47	9.916	30	7.092	108	3.371
11	47	9.916	30	7.092	83	2.591
12	56	11.814	43	10.165	95	2.965
13	46	9.705	37	8.747	68	2.122
14	35	7.384	18	4.255	37	1.155
15	65	13.713	43	10.165	76	2.372
16	37	7.806	24	5.674	89	2.778
17	48	10.127	31	7.329	77	2.403
18	57	12.025	40	9.456	66	2.060
21	59	12.447	37	8.747	66	2.060
22	25	5.274	16	3.783	61	1.904
23	100	21.097	74	17.494	128	3.995
24	30	6.329	17	4.019	74	2.310
25	53	11.181	40	9.456	54	1.685
26	37	7.806	24	5.674	69	2.154
27	44	9.283	31	7.329	72	2.247
28	77	16.245	59	13.948	74	2.310
29	46	9.705	37	8.747	81	2.528
30	59	12.447	46	10.875	154	4.806
31	73	15.401	42	9.929	70	2.185
32	25	5.274	16	3.783	61	1.904
33	28	5.907	19	4.492	57	1.779
36	28	5.907	19	4.492	53	1.654

All in all, then, the proposed cluster search method seems to be reasonably efficient. By varying the amount of work the algorithm attempts to automatically perform a narrow or somewhat broader search according to the relative degrees of similarity observed between

documents examined and centroids stored on the to-be-expanded heap. Even the broadest search, however, does not appear to be overly expensive.

7.4.2. Clustering Experiments

Finally, now that the methodology and preliminary tests have been explained, it is timely to consider experiments conducted to provide an initial test of the extended vector model. Though when various clusterings of 55 documents were discussed in Section 7.3 the one resulting from using a combination of the $\vec{bc}$ and $\vec{cc}$ subvectors seemed to give better results than when the $\vec{tm}$ subvector alone was used, the evidence was far from conclusive. To be sure that extended vectors yield better clusters than when only terms are used, one should focus on the effectiveness of large retrieval runs.

In particular, then, the following experiment was planned:

(1) Fix parameters for the cluster search program.

(2) Have a large document collection – 3204 CACM articles with 7 subvectors each.

(3) Have a reasonable number of queries – 52 were employed.

(4) Have relevance judgments for each query – an elaborate scheme involving repeated searches and various feedback techniques was employed to approximate getting full relevance information; see details in [Fox 1983b].

(5) Fix general document clustering parameters – e.g., maximum node size before a cluster should be forced to split.

(6) Vary cluster parameters related to the combined retrieval process such as the coefficients for each concept type (used to compute combined similarities).

7.4.2.1. Fixed Parameters

Since clustering routines are affected by a number of parameters, some of the actual settings made are described below.

(1) divide = 20

Clusters were split when the size reached 20. For more effective results perhaps this should have been reduced but at least when large clusters were split there were usually enough similar documents to give reasonable selection.

(2) overlap of roughly 10% allowed.

A moderate amount of overlap among clusters is reasonable. Thus about one in ten articles could be similar enough to the centroids of at least two different clusters to be included in each.

(3) loose-child clusters form if $\geq$ 3 unassigned vectors.

Thus, a few forced assignments are not considered bad but otherwise constructing an "orphans" cluster seems warranted.

(4) concentration and definition tests.

These tests attempt to ensure that clusters will have some meaningful basis for formation. The cluster concentration test requires 2 vectors with correlation of at least .9 standard deviations from the mean and 30% of the children of the proposed cluster not assigned to other centroids. The cluster definition test requires 5 vectors with correlation at least .1 standard deviation from the mean. Given that splitting occurs when there are 20 children, even if the distribution of correlations is not nearly normal, it still should allow at least 2 or 3 clusters to be formed.

(5) subvector coefficients.

For extended vectors with various components, the coefficients on the subvector similarities are .1 for authors, .05 for dates, and .17 for all other components. The rationale is that dates convey very little information to aid clustering and that author matches are relatively rare while other data should be treated equally since no definitive knowledge about relative importance is available.

(6) subvector sizes.

Limits on sizes of the subvectors are set based on the number of possible values for each. For dates the ceiling is 50, for *Computing Reviews* categories 100, while for authors, bibliographic coupling, links, and co-citations the upper bound is 300. Since the overall cluster size limit is 2048 concepts, the number of terms must be less than that, and if all other categories are also present, a limit of 1200 is used for constraining the terms.

7.4.2.2. Variables

The crucial variable to obtain experimental information about is that of whether extended vectors are better than vectors with terms only. Unfortunately, there are other related variables that are confounded with this most important one so they must be defined and interactions considered before any general conclusions can be reached. The variables of interest are:

(1) Do terms alone or a mixture of terms and other vector components perform more effectively?

(2) What is the effect of vector length on results?

(3) What weighting of centroid concepts is best?

(4) What order of addition should be followed in building the cluster tree?

To be more specific one should consider the actual variable settings arrived at. First, there is the make up of centroid vectors. One can have a limit of 2048 terms, a limit of 2048 concepts with no more than 1200 terms and a mix of other concept types, or simply have a short vector with no more than 1200 terms. Second, the concepts can have tf (term frequency) or tf*idf (term frequency times inverse document frequency) weights. Finally, documents can be added in various ways. Random addition is possible but it seems likely that depending on the make up of vectors other orderings might be better. If extended vectors are used it seems likely that adding longer vectors first, e.g., those receiving many citations and hence having longer $\vec{bc}$, $\vec{cc}$, and $\vec{ln}$ subvectors would be wise. Vector length should also depend on the age of the documents. Thus, the following addition schemes were tested:

(1) random.

(2) document identifier (did) ascending – roughly oldest first.

(3) did descending – roughly newest first.

(4) number of citations ascending, and if there are ties, then order by did ascending – roughly shortest first.

(5) number of citations descending, then did ascending – roughly longest first, but of those without citations oldest first.

(6) number of citations descending, then did descending – roughly longest first and then newest first.

7.4.2.3. Contrasts

In order to obtain some insight regarding appropriate combinations for later testing, 11 cases were identified which partially cover the range of reasonable settings of 3 key variables. Table 7.15 identifies the values of each variable for all 11 tests.

Many contrasts are possible using these cases. To compare use of a mix versus having terms only, where the total vector length was 2048, one can consider 3 versus 4 or 6 versus 7. By comparing 6, 7, and 8 one can see the effects of having a mix of 2048 concepts (with up to 1200 terms) versus 2048 terms versus 1200 terms. To contrast weighting schemes, one can compare 1 versus 2 or 7 versus 11. To contrast addition orders, one can compare 1 versus 3 versus 5 versus 6, 2 versus 9 versus 10, or 4 versus 7.

Thus, the contrasts listed above should give a number of insights as to the best clustering method to employ. Since each cluster run requires about a day of computer and

Table 7.15: Variable Settings for 11 Test Cases

Test Case No.	Terms of Mix	Max. Vector Length	Concept Weighting Scheme	Order of Addition
1	mix	2048	tf	did ascending
2	mix	2048	tf*idf	did ascending
3	mix	2048	tf	no. citing desc., did desc.
4	terms	2048	tf	no. citing desc., did desc.
5	mix	2048	tf	did desc.
6	mix	2048	tf	random
7	terms	2048	tf	random
8	terms	1200	tf	random
9	mix	2048	tf*idf	no. citing desc., did asc.
10	mix	2048	tf*idf	no. citing asc., did asc.
11	terms	2048	tf*idf	random

special manual processing it is valuable to be able to obtain as much guidance as possible from this experiment. In the future more comprehensive tests can be made to better focus in on the most appropriate variable settings.

7.4.2.4. Test Results

Various measures have been proposed for evaluating clustered searches. For the cases of interest, a search for 10 documents and another alternate search for 30 documents were conducted. Precision, recall, and E value (as in [Jardine & Van Rijsbergen 1971]) for various β values are reported. In addition, since the 30 document search is fairly broad, the average precision measure value as given for earlier experiments is also given.

Table 7.16 shows the results for retrieving 10 (no_retrieved=10). Note that for those familiar with seeing higher values in connection with better performance, the reported statistic is 1-E instead of E. Further, the three settings of β listed should cover typical user interests. Note that β indicates how important recall is relative to precision. Table 7.17 is similar to 7.16, but for no_retrieved=30.

First, consider the best cases. For no_retrieved=10, a high precision search, case 2 is best, corresponding to having a mix of concept types, tf*idf weights on document and centroid concepts, and adding oldest documents first. On the other hand, for the broader search with no_retrieved=30, the best case, 11, was to add documents in random order, use tf*idf weights, and only include the term subvector (expanded to be up to size 2048). Since all possible cases were not tested, however, these results are not terribly conclusive.

Table 7.16: Clustered Search Results for 10 Retrieved

Test Case	Precision Value	Recall Value	1 - E $\beta = .5$	1 - E $\beta = 1$	1 - E $\beta = 2$
1	0.269	0.233	0.225	0.199	0.199
2	0.281	0.252	0.234	0.208	0.211
3	0.237	0.168	0.195	0.168	0.160
4	0.262	0.225	0.214	0.187	0.188
5	0.215	0.171	0.180	0.157	0.153
6	0.204	0.163	0.168	0.147	0.147
7	0.190	0.157	0.156	0.137	0.137
8	0.181	0.155	0.150	0.132	0.133
9	0.250	0.220	0.204	0.179	0.182
10	0.267	0.232	0.222	0.197	0.198
11	0.262	0.226	0.215	0.189	0.189

Table 7.17: Clustered Search Results for 30 Retrieved

Test Case	Precision Value	Recall Value	1 - E $\beta = .5$	1 - E $\beta = 1$	1 - E $\beta = 2$	Average Precision
1	0.135	0.298	0.141	0.157	0.194	0.1473
2	0.143	0.314	0.149	0.165	0.203	0.1649
3	0.121	0.232	0.127	0.141	0.171	0.1330
4	0.134	0.290	0.138	0.151	0.185	0.1624
5	0.105	0.222	0.110	0.123	0.151	0.1181
6	0.106	0.239	0.110	0.123	0.153	0.1104
7	0.094	0.200	0.097	0.107	0.132	0.1041
8	0.081	0.188	0.085	0.096	0.121	0.0887
9	0.146	0.302	0.150	0.164	0.198	0.1704
10	0.141	0.299	0.147	0.162	0.199	0.1571
11	0.149	0.336	0.156	0.175	0.217	0.1723

Second, consider the contrasts regarding weighting methods. Regardless of the number retrieved and the β values, tf*idf weighting is superior to tf weighting.

Third, consider the matter of order of addition of documents. Here it should be pointed out that Williamson claimed his algorithm was relatively immune to the effects of order of addition. The current findings do not support his observation, possibly because addition order is confounded with the multiple concept type issue. Considering extended vector cases 1,3,5 and 6 one observes that their performance follows the ordering from worst to best of 6, 5, 3, 1 (in all cases except for a few where 3 and 1 are swapped). This suggests that random is worst, followed by did ascending, no. citing descending and did descending, and ending with did ascending as best. Thus, for a mix of concept types one should add the oldest (hence shortest) documents first. Comparing cases 4 and 7, one again sees that random addition is unwise. When cases 2, 9, and 10 are compared, one has ordering 2, 10, 9 for no_retrieved=10 and 10, 2, 9 for most cases of no_retrieved=30. It appears that the no. citing descending and did ascending scheme is best but conclusions are not very firm. Apparently, by itself or in combination with no. citing descending, it seems wise to have did ascending. In summary, then, it is relatively clear that random addition gives poor results while adding documents is ascending sequence number (which roughly corresponds to date of publication) aids performance.

Finally, consider the matter of whether to have terms only or extended vectors. When a total vector length of 2048 is enforced, and one can either have a term vector of 2048 entries or an extended vector of the same length (but with no more than 1200 terms included), the results are found to depend on order of addition. When documents are randomly added, the mix is better than if there are only terms and having 2048 terms is better

than having 1200. However, possibly because the addition order is particularly bad for extended vectors (see discussion in the preceding paragraph), when documents are added following no. citing descending and did descending, it is best to use only terms in the vectors. One can only tentatively conclude, then, that if documents are wisely added to the cluster tree, it is better to use extended vectors.

7.4.2.5. Cluster Run Conclusions

Based on the 11 test runs made some tentative conclusions can be reached, subject, of course, to further experimental verification. Such runs would include variations in the basic clustering parameters (e.g., having smaller centroids) and possibly even working with other collections or clustering algorithms.

Nevertheless, it is valuable to restate the conclusions of the several contrasts made. First, it seems best to employ tf*idf weights; that is reasonable since clustering should be better when the importance of concepts is properly considered. Second, it seems best to add documents in a particular ordering. The easiest to implement, and one of the best, is to add documents oldest first – in the natural order they arise. Finally, it appears that if document addition is not done in an unwise fashion that extended vectors may be better than vectors with terms only.

It might be of interest to propose an explanation for some of these findings. When the earliest documents are added first, they help construct the initial skeleton of the cluster tree. Since they probably only have entries in the term subvector, the initial tree is constructed using terms which provide all the then available information. Later as vectors are added with other subvector elements they are placed in the proper place based on their term components. Even if some earlier documents have entries in the $\vec{cc}$ subvector it is

unlikely that they will be linked to many other early additions. However, as more and more documents are added the tree picks up entries with extended vector portions and those are included in all affected centroids. Thus, gradually, additions and splitting become influenced by the extended vector elements. Documents lacking these extra parts are added early in the process and so will not be put in the wrong place by a split operation due to their not having a large value of the combined similarity measure; they were located in the correct node when all combined similarity values were relatively low (since averaging a number of zero components in with the term based similarity gives a low overall result). Conceivably, this hypothetical explanation may explain how extended vector clustering should best be carried out.

Since there are so many parameters involved it is unclear at this time if extensive further clustering tests are worthwhile to test the hypothesis of whether extended vectors are better than simple term vectors. Hence that hypothesis will instead be further checked in the next chapter using feedback techniques generalized to apply to extended vectors.

CHAPTER 8

FEEDBACK WITH EXTENDED VECTORS

In Chapter 6, the extended vector model of document representation was proposed and explained. In Chapter 7, the model was tested using cluster formation and search techniques. However, due to the subjectivity of comparing classifications produced by clustering and the complexity of interactions among parameters involved in obtaining cluster search results, it is not particularly clear how well the extended vector model has been validated through those clustering runs.

To give a more convincing test of the extended vector model further experiments are described in this chapter using relevance feedback. The underlying perspective is that by feeding back user judgments one can extend a short, terms-only user supplied query with information from one or more relevant retrieved documents and thereby obtain a query with extended vector elements as well. If that extended vector performs more effectively than the term portion of it, the usefulness of extended vectors is demonstrated.

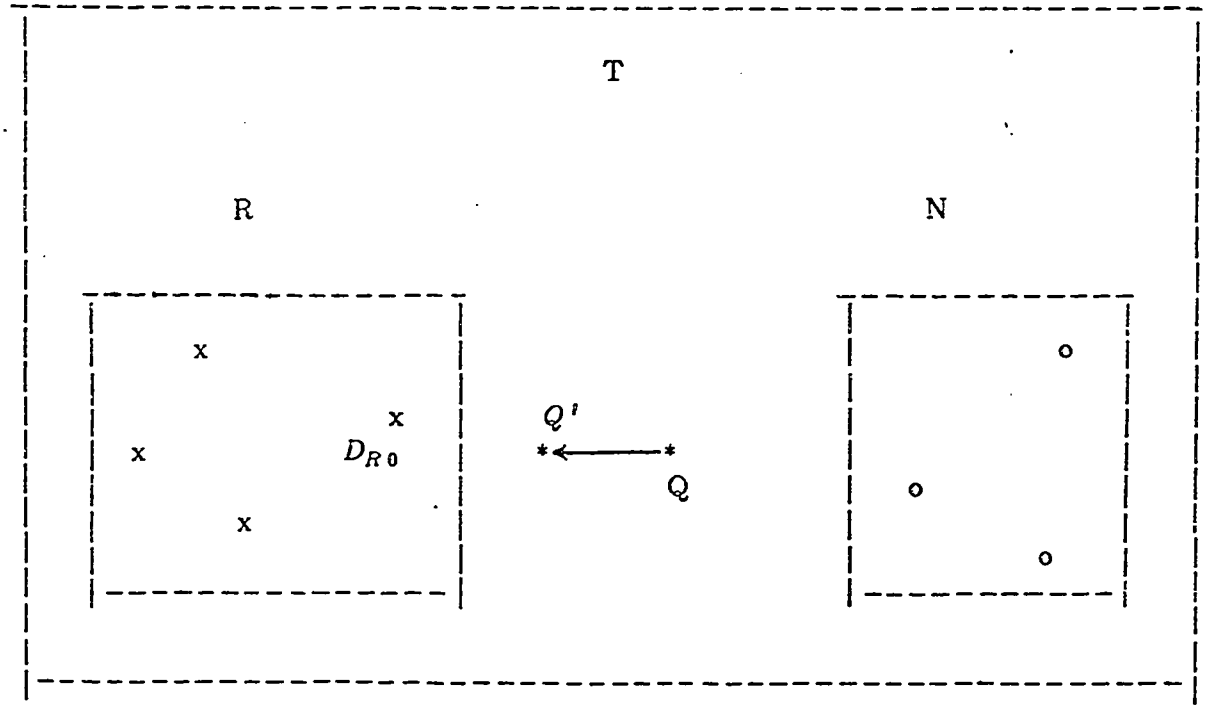
Before going into the details of extended vector feedback some background in the theory of vector feedback is required. The discussion in Section 8.1 should complement that at the beginning of Chapter 5, where feedback concepts were originally introduced. Now, however, the underlying query model is that of vectors instead of Boolean or p-norm forms.

8.1. Previous Work

8.1.1. Basic Feedback Model

Figure 8.1 provides intuition regarding the key aspects of feedback processing. Query

Figure 8.1: Feedback Diagram in Vector Space



Key:

* designates query, where Q is original and Q' is new

R is the set of relevant documents, each shown as "x"

N is the set of non-relevant documents, each shown as "o"

T is the total set of documents in the top ranked list

Q is submitted to the retrieval system which retrieves a set T of top ranked documents through some means such as by a limited cluster search. Of these $|T|$ documents, the $|R|$ in set R are judged relevant and the $|N|$ in set N are considered not relevant. The object is to improve Q with this feedback information. The technique suggested in [Rocchio 1971] and [Ide 1971] is to move toward the relevant retrieved documents and move away from the non-relevant retrieved set. Thus, Q' should be better than Q. Rocchio's formula for this is essentially:

$$Q' = \alpha Q + \beta\left(\frac{1}{|R|}\sum_{D_i \in R} D_i\right) - \gamma\left(\frac{1}{|N|}\sum_{D_i \in N} D_i\right) \tag{8-1}$$

where α, β, and γ are constants (e.g., 1, .5, .25) specifying the relative importance of the original query, the relevant documents retrieved, and the non-relevant documents retrieved.

Figure 8.1 is an idealized version of what occurs in practice, since the relevant and non-relevant documents need not be so well separated. It is possible, too, that either set may be empty; when R is empty the only real recourse is to continue the search with the original query or perhaps a slightly modified form of it.

8.1.2. Feedback Evaluation

After an initial search has been conducted and a new query Q' is formed, the key question to consider regarding evaluating the new results for a search with Q' is what should be done with the documents seen by the user. Several schemes for dealing with this are proposed in [Ide 1971] and [Chang, Cirillo & Razon 1971].

In accord with the decision made earlier for evaluating Boolean feedback (see Section 5.3.6), the partial rank freezing technique is the one selected. In reality, the choice of evaluation method is probably not very important, as long as it is consistently made.

8.1.3. Term Relevance Feedback

An important contribution to the theory of retrieval made during the last decade was the idea of using decision theoretic or probabilistic methods to arrive at near optimal feedback performance. Yu and Salton [1976] and a number of British researchers (in [Robertson & Sparck Jones 1976], [Van Rijsbergen 1977], [Harper & Van Rijsbergen 1978]) suggested a technique that will be referred to here as term relevance weighting. Though various formulations and related estimations methods have been proposed, only one form of the general scheme will be described.

Consider the documents in a collection, and whether they are indexed by the i^{th} term of query Q and whether they appear in the relevant or non-relevant sets. The numbers in each category are shown in the contingency chart of Table 8.1 below (which is like Table 5.1 of Chapter 5).

Robertson and Sparck Jones [1976] derived a formula for term weighting which gives optimal weights under the assumption that terms are independent. If complete relevance information is available, the weight computed is

$$\log \left[\frac{r}{R-r} \Big/ \frac{n-r}{N-n-R+r} \right]. \qquad (8\text{-}2)$$

Various strategies have been proposed to estimate the parameters based on feedback data or to adjust the formula for trivial cases (e.g., $r = R$ or $r = n$) but the basic method should be clear enough from equation (8-2).

Table 8.1: Feedback Contingency Chart

Indexing	Relevance: Relevant	Relevance: Non-relevant	(totals)
Term i Present	r	$n-r$	n
Term i Absent	$R-r$	$N-R-n+r$	$N-n$
(totals)	R	$N-R$	N

where

N = number of documents in collection

R = number of relevant documents

r = number of relevant documents with term

n = number of documents with term

8.1.4. Feedback of Extended Vectors

In 1963 Salton reported on some preliminary tests where vectors included elements of bibliographic references [Salton 1963]. In 1971, he described further work using such information for feedback [Salton 1971b]. A related work [Michelson 1971] gave further evidence of the utility of feedback methods when references were available.

None of these tests, however, used sophisticated feedback weighting techniques such as those of the last section. Furthermore, the idea of having separate subvectors was not proposed so different weighting coefficients were not considered either. Finally, tests were done with small sets of data – several large collections with a number of queries were not then available and only bibliographic references (akin to the $\vec{In}$ subvector) were added to terms.

Nevertheless, the early tests with feedback of extended vectors were sufficiently promising to encourage further investigation. In the next section, preliminary tests of extended vector feedback on the ISI and CACM collections are described.

8.2. Single Document Feedback

To quickly test the utility of extended vector feedback, a very simple scheme was chosen. Referring to Figure 8.1, one notes that there must be some relevant document, say D_{R0}, which is the closest relevant document to query Q. In other words, if a user scans the ranked list returned in response to the query, eventually the top ranked relevant document will be identified.

The feedback query Q' can then simply be the document vector D_{R0}. Viewed in terms of equation (8-1), one has $\alpha = 0$, $\beta = 1$, $|R| = 1$, $r = D_{R0}$, and $\gamma = 0$. Since only a single relevant document is considered, this technique will be called single document feedback.

Aside from its simplicity, this technique has a number of advantages. First, since a document is selected and that document is the closest relevant to the query it is likely that useful extended vector elements will be present in Q'. Second, forming Q' requires no real programming effort and so tests are easy to perform. Third, there are no problems with

parameter settings or estimation. Finally, since the only requirement is that the top ranked relevant document be chosen one is assured that a relevant document will be identified even if it is not in the top 10 positions for example.

On the other hand, the single document scheme does have several disadvantages. First, the original query (along with its set of terms) is completely ignored; it is assumed that the top ranked relevant document captures all important information from it which is certainly not the case. Second, it is unclear how further iterations could be carried out and doubtful if they would improve over results of the first iteration. Third, the implicit assumption that the documents relevant to Q are exactly the same as documents relevant to D_{R0} is clearly not true. Finally, by only considering a single highly ranked relevant document one severely limits the overall length of the new vector Q' and reduces the accuracy of weights on its concepts. If instead a group of relevant documents had their vectors averaged, then unimportant high frequency terms might be down weighted and useful lower frequency terms would receive higher weights.

Now that the important pros and cons of the single document method have been aired and the basic notion has been explained it is appropriate to proceed with discussion of the experiments carried out.

8.2.1. ISI Experiments

8.2.1.1. Subvectors Used

The ISI collection has three subvectors in each document representation, $\vec{tm}$ of terms, $\vec{au}$ of authors, and $\vec{cc}$ of co-citations. For a query Q, some document, say $\vec{D}_i$, is retrieved as the top ranked relevant document and so the feedback query Q' is made up of $\vec{tm}_i$, $\vec{au}_i$,

and $\vec{cc}_i$. Here $\vec{tm}_i$ is the list of terms in $\vec{D}_i$, $\vec{au}_i$ represents the authors, and all entries cc_{ij} indicate that article numbers i and j are co-cited.

According to the definition of the $\vec{cc}$ subvector one notes that cc_{ii} will be greater than zero. That is because the corresponding term frequency value is the number of articles citing the i^{th} document and the document itself is included. Now consider another document vector D_j where the j^{th} document is co-cited with the i^{th} document. Certainly, cc_{ij} will be greater than zero. If one defines a vector $\vec{cd}_i$ like $\vec{cc}_i$ but where all entries are zero except for a one in the i^{th} column then the inner product of $\vec{cd}_i$ and $\vec{cc}_j$ is simply the value $cc_{ji} = cc_{ij}$, namely, the weight for the co-citations between documents $\vec{D}_i$ and $\vec{D}_j$. Thus, $\vec{cd}_i$ selects out "direct co-citations" between the i^{th} document and others.

On the other hand, if one computes the cross product between $\vec{cc}_i$ and $\vec{cc}_j$ there may be several values included. Certainly, one has positive values in pairs cc_{ii} and cc_{ji}, cc_{ij} and cc_{jj}. But there may be "indirect" or transitive connections as well between the two documents. Take, for example, the case where $\vec{D}_i$ and $\vec{D}_k$ are co-cited as are $\vec{D}_j$ and $\vec{D}_k$. Then the pair cc_{ik} and cc_{jk} will also be included in the cross product of $\vec{cc}_i$ and $\vec{cc}_j$. Hence, using $\vec{cc}_i$ selects out both direct and indirect co-citations. See Figure 8.2 for an example illustrating the differences.

In the following discussion, then, the subvectors considered will be $\vec{tm}$, $\vec{au}$, $\vec{cc}$, and $\vec{cd}$ where the last two refer to considering direct and indirect co-citations, or just direct co-citations.

Figure 8.2: Direct and Indirect Co-Citations
(Example co-citation submatrix with tf weights; x=don't care)

vector	document concept numbers 1	i	j	k	N
$\vec{cc}_1$	x	x	x	x	x
$\vec{cc}_i$	0	4	3	1	0
$\vec{cc}_j$	0	3	5	2	0
$\vec{cc}_k$	x	x	x	x	x
$\vec{cd}_i$	0	1	0	0	0

Sample computations of inner products:

(1) Direct + indirect co-citations.

$$\vec{cc}_i \cdot \vec{cc}_j = 4 \cdot 3 + 3 \cdot 5 + 1 \cdot 2 = 29$$

(2) Direct co-citations only.

$$\vec{cd}_i \cdot \vec{cc}_j = 1 \cdot 3 = 3$$

8.2.1.2. Retrieval Results

In Section 8.2.1.1, the four possible subvectors to test were discussed. Equation (8-3) gives a linear model for combined similarity computation using all of the subvectors and allows a uniform labelling of the retrieval results that follow.

$$SIM(\vec{Q}', \vec{D}_i) \tag{8-3}$$
$$= w_{tm} \cdot SIM(tm_{Q'}, tm_i)$$
$$+ w_{au} \cdot SIM(au_{Q'}, au_i)$$
$$+ w_{cc} \cdot SIM(cc_{Q'}, cc_i)$$
$$+ w_{cd} \cdot SIM(cd_{Q'}, cd_i)$$

According to equation (8-3), one can compute the similarity between the feedback query $\vec{Q}'$ and a document as a linear combination of the similarities between respective subvectors of that query and document. If only one coefficient is non-zero then the effect of using a single subvector can be seen. With two non-zero coefficients one can observe the effects of pairs of concept types while with more positive coefficients one can explore the effects of other combinations. In all cases of Section 8.2 cosine correlation is used for the similarity function SIM of (8-3).

The simplest case is that of single subvectors. Table 8.2 gives retrieval results for the ISI collection when only one concept type is considered. The base case is having standard

Table 8.2: Results of ISI Single Document Feedback
Using One Subvector

subvector used	average precision	% change from terms only
tm	.2465	
au	.0759	-69%
cc	.1559	-37%
cd	.1129	-54%

terms only vectors without any extensions. Author subvectors by themselves are not very useful for retrieval. Co-citations do much better though still not as well as terms. When only direct co-citations are considered they are better than authors but still not as good as the combination of direct and indirect co-citations. A possible explanation is that the author submatrix is very sparse, yielding few matches, and that direct co-citations give more matches. The average length of $\vec{cc}$ subvectors is about the same as that of tm subvectors so the probable cause of $\vec{cc}$ doing less well than $\vec{tm}$ is that relevant documents almost always have common terms but need not be co-cited with each other.

To validate the extended vector model, however, the key notion to test is whether having more than one subvector included is worthwhile. Accordingly, Table 8.3 shows results for various combinations of subvectors. From equation (8-2) it should be clear that

Table 8.3: Results of ISI Single Document Feedback Using Several Subvectors

Combination Cases				Average	% Change vs.
w_{tm}	w_{au}	w_{cc}	w_{cd}	Precision	Terms Only
1.0				.246	
.5	.5			.229	-7.2
.5			.5	.251	+1.8
.5		.5		.231	-6.2
.88		.12		.261	+6.1
.34	.33		.33	.234	-4.9
.34	.33	.33		.227	-7.8
.85	.05		.10	.252	+2.2
.85	.05	.10		.261	+5.8
.70	.05	.25		.257	+4.5

weights for each subvector must be specified. The obvious choice given no prior knowledge is to use equal weights. In addition, however, guesses based on the performance of each single subvector were also tried. The results of Table 8.3 suggest a number of things about the use of combined similarity:

(1) Terms are very useful and if the term subvector is not given sufficient relative weight then performance with a composite vector will be worse than having terms only.

(2) Author subvectors are not very useful and if highly weighted will cause severe performance losses.

(3) When highly (in this case that means equally with other components) weighted, direct co-citations are better to use than direct plus indirect co-citations. A possible explanation relates to the fact that there are fewer matches to $\vec{cd}$ than $\vec{cc}$. Thus, for $\vec{cd}$, using equal weights will only affect similarities in the rare cases when a direct co-citation exists and there the likelihood of relevance is fairly high. The $\vec{cc}$ subvectors are longer so using equal weights with them might lead to great precision loss.

(4) Improvements over the use of terms only occur when terms are weighted fairly highly, and other subvectors are also included. It is best to omit $\vec{au}$ altogether. The best results of cases tried occur when more than 80% of the weighting is attached to $\vec{tm}$.

From Table 8.3, one should then conclude that extended vectors do seem useful, but only when proper coefficients are used on the subvectors. The question then remaining is how these coefficients can be determined. One possible solution is proposed in the next section.

8.2.1.3. Regression Based Coefficients

As has been seen in Chapter 3, regression is a powerful technique, useful to better understand the behavior of parameters involved in retrieval processing. One suggestion, then, is to use regression methods to obtain values for the coefficients in the combined similarity formula of equation (8-3).

Equation (8-3) specifies how combined similarity depends on each subvector similarity and the corresponding coefficients. To a user, however, the binary relevance value is what is important, and similarity should ideally try to match that. Hence, an appropriate regression model is that of equation (8-4).

$$\begin{aligned} \text{Relevance}_i \quad & (8\text{-}4) \\ &= w_{tm} \cdot SIM(tm_{Q'}, tm_i) \\ &+ w_{au} \cdot SIM(au_{Q'}, au_i) \\ &+ w_{cc} \cdot SIM(cc_{Q'}, cc_i) \\ &+ w_{cd} \cdot SIM(cd_{Q'}, cd_i) \end{aligned}$$

That is, for pairs of documents and queries one tries to decide whether the documents are in reality relevant by using a linear formula in terms of the query-document subvector similarities.

Given the model, the next step is try to obtain actual data for arriving at a reasonable fit. An initial attempt was made using the feedback values only, but because of the small amount of data, no good fit was detected. Consequently, the data chosen was the feedback values plus values for other relevant documents. In other words, for each query, similarity and relevance values were identified for all relevant documents and for all non-relevant documents retrieved by an initial search of 20 documents.

Objection might be raised regarding the use of all relevance values. However, there was no other real alternative. Further, the coefficients arrived at are not far from those guessed at (see Table 8.3) and are probably document collection but not query collection dependent. They were obtained as average values for the given set of queries and so lack the direct connection to a single query that probabilistic retrospective weights owe to their rather different origin. Hence they should be useful for most any similar query collection. Though it would be better experimentally to use the arrived at coefficients on another query collection, the approach taken seems fairly reasonable.

Table 8.4 shows the regression results for the model of equation (8-4). As was mentioned earlier, $\vec{au}$ is not very useful, and so should be dropped from the model. The fit is not very good, but nevertheless it is hoped that the coefficients will be usable. The t-values indicate that $\vec{tm}$ and $\vec{cc}$ subvectors are useful, and that $\vec{cd}$ is probably also. Accordingly, Table 8.5 gives regression results for $\vec{tm}$ and $\vec{cc}$, while Table 8.6 gives results for the $\vec{tm}$ and $\vec{cd}$ combination.

Table 8.4: Regression of Coefficients for Subvectors on ISI Relevance Data

Variable	Coefficient	Std. Error	t-Value
tm	1.71	.05	33.6
au	-0.07	.05	-1.4
cc	0.24	.05	5.2
cd	0.48	.15	3.2

Residual Std. Error 0.6 Multiple R-square 0.18

Table 8.5: Regression of Coefficients for $\vec{tm}$, $\vec{cc}$ on ISI Relevance Data

Variable	Coefficient	Std. Error	t-Value
tm	1.70	.05	32.6
cc	0.34	.03	9.7

Residual Std. Error 0.6 Multiple R-square 0.18

Table 8.6: Regression of Coefficients for $\vec{tm}$, $\vec{cd}$ on ISI Relevance Data

Variable	Coefficient	Std. Error	t-Value
tm	1.74	.05	32.8
cd	0.99	.11	8.7

Residual Std. Error 0.6 Multiple R-square 0.17

Even though the regressions do not exhibit very good fit to the data it is worthwhile to see how well retrieval does when the suggested coefficients are utilized. Table 8.7 summarizes the performance, using the same scheme as in Table 8.3. For contrast, the same base case of terms alone is included as well as the best combination guessed at earlier. The regression coefficients have been scaled so for each case they add to a total of 1.0.

Apparently, the best combination is to use $\vec{tm}$ and $\vec{cc}$. Regression methods lead to a combined similarity computation which is a 5% improvement over terms alone. A slightly better set of coefficients were guessed at, leading to a 6% improvement.

Table 8.7: Results of ISI Single Document Feedback
Using Regression Determined Coefficients

Combination Cases Name	w_{tm}	w_{cc}	w_{cd}	Average Precision	% Change vs. Terms Only
base	1.0			.246	
guess	.88	.12		.261	+6.1
regression1	.83	.17		.259	+5.2
regression2	.64		.36	.251	+1.7

8.2.1.4. Conclusions for ISI Document Feedback

Using the single document feedback technique, comparisons have been made between the use of various single subvectors and combinations of two or more subvectors. For the combination cases, equal weights, guessed at weights, and weights determined by regression techniques have been utilized.

Of all the subvectors, terms are best, though co-citations are not much worse. Author subvectors are not worthwhile, alone or in combination. Using regression or guessed at coefficients, the $\vec{tm}$ and $\vec{cc}$ combination yields a 5-6% improvement over the performance when terms alone are used.

Though the performance increment is not especially dramatic, it does seem to indicate that combined vectors are worth using for feedback. Consequently, further testing was done using the CACM collection.

8.2.2. CACM Experiments

Given the background of Section 8.2.1, which describes ISI experiments in single document feedback, corresponding CACM experiments should be easily understood. The only complexity for the CACM collection is that instead of three basic concept types there are six (if one omits dates which are unlikely to be very useful).

8.2.2.1. Single Subvectors

In addition to terms, authors, and co-citations, there are three extra subvectors: $\vec{cr}$ for *Computing Review* categories, $\vec{bc}$ for bibliographic coupling, and $\vec{ln}$ for links. Since for ISI, direct co-citations were examined also, some tests have also been made with: d-$\vec{bc}$, direct bibliographic coupling; d-$\vec{cc}$, direct co-citations; and d-$\vec{ln}$, direct links.

Table 8.8 shows the results for each of these single subvector cases. As expected, terms are still the best single concept type. Next best is $\vec{ln}$, possibly because there are probably more entries in those subvectors than others. Then comes $\vec{cc}$ followed by $\vec{cr}$ and then $\vec{bc}$.

Since the CACM collection has bibliographic data based only on internal references, that is, between pairs of articles in the same journal, it is not surprising that the $\vec{bc}$ subvector seems rather sparse and not very useful. As hoped, co-citations are reasonably useful and are more definitive than the rather loosely defined *Computing Reviews* categories.

Regarding the "direct" subvectors, these are uniformly worse than the normal subvectors they are based on. Since bibliographic data in CACM is somewhat sparsely present, it is best to use the longer full vectors and benefit from both direct and indirect connections.

Table 8.8: Results of CACM Single Document Feedback
Using One Subvector

subvector used	average precision	% change from terms only
tm	.3153	
au	.1135	-64
cr	.1376	-56
bc	.1189	-62
ln	.2106	-33
cc	.1544	-51
d-bc	.1127	-64
d-ln	.1458	-54
d-cc	.1244	-61

8.2.2.2. Regression Tests

Since determining the behavior of extended vectors is what is of real interest, regression methods were once again applied. With so many possible combinations to consider, however, the technique of "leaps and bounds" regression [Becker & Chambers 1981] was initially employed. That approach starts with the best single entry, considers the best pairs, then tries triples, etc. – it focuses on what are likely to be good groupings.

For each such combination of subvectors, the C_p statistic is computed. And for a combination of n cases, having a value of C_p close to n is desirable. Table 8.9 gives the C_p values for those combinations considered which seem relatively good.

As was shown in Table 8.8, among the possible single subvectors, $\vec{tm}$ and then $\vec{ln}$ are best. Consequently, $\vec{tm}$, $\vec{ln}$ is the best pair. It is followed by $\vec{tm}$, $\vec{cr}$ since terms and *Com-*

Table 8.9: C_p Statistic Values for Subvector Combinations

Desired Value	C_p Computed	Combination
1	577.2	tm
1	949.8	ln
2	177.4	tm,ln
2	345.6	tm,cr
3	23.2	tm,cr,ln
3	164.2	tm,au,ln
4	12.9	tm,au,cr,ln
4	15.3	tm,cr,bc,ln
5	5.2	tm,au,cr,bc,ln
5	14.8	tm,au,cc,cr,ln
5	17.0	tm,cc,cr,bc,ln
6	7.0	tm,au,cc,cr,bc,ln

puting Reviews categories give different kinds of information and so should be more useful when combined. As expected, the three types best for pairs make up the best triple: $\vec{tm}$, $\vec{cr}$, $\vec{ln}$ which is substantially better, according to the C_p statistic, than the next best triple: $\vec{tm}$, $\vec{au}$, $\vec{ln}$.

Once 4 or more subvectors can be included, there are a number of combinations that give relatively good performance. The best 4 are $\vec{tm}$, $\vec{au}$, $\vec{cr}$, $\vec{ln}$ as one might expect from considering the two best triples. Adding in $\vec{bc}$ gives the best combination of 5 subvectors.

In addition to the "leaps and bounds" test, the actual regression values for various combinations were determined. The best fit is with all 6 subvectors included, though the

multiple R-square value is not especially good even there. The results are shown in Table 8.10, with an added column giving coefficients scaled down so the values add to 1.0.

In Table 8.10, one can see that the least reliable coefficients are those for $\vec{au}$ and $\vec{bc}$ subvectors. The most important ones as judged by highest coefficient and t-values are $\vec{tm}$, $\vec{ln}$, and $\vec{cc}$. Though the coefficient value is low for $\vec{cr}$ the t-value is still reasonably high so it is fairly sure that using it with a very low weight is the appropriate choice.

8.2.2.3. Feedback Results for Combinations

Given the results of "leaps and bounds" regressions, the regression presented in Table 8.10, and similar runs for other combinations of interest, a number of feedback searches were planned and conducted. For comparison purposes, runs were also made with equal weights on coefficients. The results are shown in Table 8.11.

The first half of the table is for cases where all coefficients involved receive equal weights, and the second half is where weights are based on a regression run aimed at fitting

Table 8.10: Regression of 6 Subvectors on CACM Relevance Data

Sub-Vector	Scaled Coeff.	Actual Coeff.	Std. Error	t-Value
au	.04	.078	.049	1.60
cr	.04	.084	.022	3.87
tm	.62	1.19	.050	24.03
bc	.03	.057	.033	1.74
ln	.185	.357	.074	4.83
cc	.085	.164	.040	4.15

Table 8.11: Results of CACM Single Document Feedback
Using Subvector Combinations
With Equal (E) or Regression (R) Based Weights

Subvectors Used						Weight	Aver.	% Change vs.
tm	au	cr	bc	ln	cc	Scheme	Prec.	Terms Only
x							.3153	
x		x				E	.2698	-14.4
x			x			E	.2953	-6.3
x				x		E	.3431	+8.8
x					x	E	.3107	-1.4
x		x		x		E	.2942	-6.7
x			x	x		E	.3187	+1.1
x			x		x	E	.3063	-2.9
x				x	x	E	.3272	+3.8
x			x	x	x	E	.3198	+1.4
x	x	x	x	x	x	E	.2962	-6.1
x		x				R	.3268	+3.6
x			x			R	.3212	+1.9
x				x		R	.3470	+10.0
x					x	R	.3261	+3.4
x		x		x		R	.3507	+11.2
x			x	x		R	.3469	+10.0
x			x		x	R	.3328	+5.6
x				x	x	R	.3463	+9.8
x			x	x	x	R	.3437	+9.0
x	x	x	x	x	x	R	.3535	+12.1

the coefficients and similarities to relevance values. It can be clearly seen that comparing precisions pairwise, the regression scheme gives better results in all cases than the equal weight scheme. Indeed, in the bottom half of Table 8.11 all the combined vector cases show a net improvement over when the term subvector alone is used.

Examining the top half of Table 8.11, one notes that in some cases there are improvements over using terms alone. All cases of improvement have both the $\vec{tm}$ and $\vec{ln}$ subvectors, which makes sense since from Table 8.10 it can be seen that these two have the highest valued coefficients: .62 and .185, respectively. The worst degradations of performance arise when $\vec{au}$, $\vec{cr}$, and $\vec{bc}$ subvectors are included, since these have lowest coefficients and so an equal weighting scheme results in an exceptionally poor performance.

8.2.2.4. Conclusions for CACM Document Feedback

Apparently, using regression based weights is a viable scheme for obtaining good improvements in performance. When all subvectors are utilized with reasonable weights, a net increase of 12% results. The best three types, $\vec{tm}$, $\vec{cr}$, and $\vec{ln}$, allow an 11% improvement while the best two, $\vec{tm}$ and $\vec{ln}$, give 10% positive change. In the CACM collection, then, bibliographic connections and *Computing Reviews* categories provide separate supplemental information that is useful to aid retrieval behavior which would normally be based solely on term matches.

The recipe proposed is to at least employ terms ($\vec{tm}$), some manually assigned categorization scheme ($\vec{cr}$), and direct links between documents ($\vec{ln}$). When bibliographic information is only available among articles in a collection the simplest form of that information, references ($\vec{ln}$), seems to be most reliable and most useful of all the types considered ($\vec{bc}$, $\vec{ln}$, $\vec{cc}$). The $\vec{ln}$ subvectors are typically longer than the other two and are easier to obtain so use of them is encouraged by practicality considerations as well as effectiveness tests.

All in all, after using the simple single document feedback scheme and determining coefficients via regression techniques, the value of extended vectors for retrieval of CACM

documents seems clear. Hence, it is worthwhile to consider more sophisticated feedback schemes than just using a single document. With more data available about each concept of each subvector, it seems likely that the value of extended vectors will increase.

8.3. Term Relevance Feedback

Section 8.2 demonstrated the value of extended vectors using a rather crude feedback scheme that employed only the single top ranked relevant document. In this section the term relevance feedback technique is adopted instead since under certain assumptions it is known to give optimal results. As in Section 8.2 the basic method will be described and test runs using equal coefficient values and regression based coefficients will be described.

8.3.1. Term Weight Computation

For the single document feedback scheme described earlier, the new query is determined according to equation (8-1) with appropriate parameter settings – actually a trivial form – where Q' is the top ranked relevant document. For term relevance weighting, however, each concept is weighted roughly according to equation (8-2).

The actual term relevance computation utilized was suggested and programmed by Chris Buckley using a slightly modified approach. First, following the example of [Wu & Salton 1981], the feedback query is a linear combination of the old query and the new feedback portion as in equation (8-4).

$$Q_{i+1} = \alpha \cdot Q_i + (1-\alpha) \cdot Q'_i \qquad (8\text{-}4)$$

where $\alpha = 0.5$.

Second, each term of Q'_i is computed using the formula (8-5).

$$\log \left[\frac{r}{R-r} \Big/ \frac{n-r}{N-n} \right]. \tag{8-5}$$

This is the same as equation (8-2), except that the final term $N-n-R+r$ in equation (8-2) is replaced by the easier to compute approximation $N-n$.

Third, in order to avoid dividing by zero and to avoid other problems [Van Rijsbergen, Robinson & Porter 1980] that arise when the Jeffrey's prior version of equation (8-2) is used, the following special adjustments were implemented

Let $R = 15$, the average assumed no. of relevant documents per query. Then (8-6)

if $r = 0$ set weight $= 0$

if $r > R$ set $R-r$ to 0.5

if $r = n$ set denominator of (8-5) to 0.5.

Finally, to compute similarity between the feedback query and a document when several subvectors are involved the combined similarity formula (8-3) is utilized. Note, however, that since term relevance weights are computed, the inner product similarity function is used instead of cosine correlation.

8.3.2. CACM Single Subvector Experiments

To test the effects of extended vectors with relevance feedback, the CACM collection was selected, since it has so many different concept types. First the performance from using each subvector by itself was gauged. Table 8.12 gives the results, much like those shown in Table 8.8, except that "direct" bibliographic connections were ruled out as being inappropriate (since they performed poorly and do not fit the term relevance scheme well).

There are a number of interesting points here. First, compared to the single document feedback runs of Table 8.8, there are significant improvements. This is not surprising, since

Table 8.12: Results of CACM Term Relevance Feedback
Using One Subvector

subvector used	average precision	% change from terms only
tm	.3839	
au	.2181	-43.1
cr	.2876	-25.1
bc	.2820	-26.5
ln	.4032	+ 5.0
cc	.2727	-29.0

term relevance is much better than only feeding back a single document. Thus, for term subvectors, the change is from average precision of 0.3153 to 0.3839, an increase of 21.8%. Hence it seems appropriate to study the utility of extended vectors with this scheme that provides better feedback queries for each of the term vectors.

Second, note that the $\vec{ln}$ subvector actually gives better feedback performance than the base case $\vec{tm}$ subvector. When enough concepts are available, from having feedback of a number of documents instead of just one, the references between articles actually seem to convey more useful retrieval information than terms.

Third, observe that in both Tables 8.8 and 8.12, the author information seems least useful. Possibly there are few authors fed back, and authors tend to write articles about many subjects, so author matches are not especially helpful, even with good weighting. In terms of absolute performance, using term relevance is still better than using a single document but for both methods $\vec{au}$ is the worst subvector.

Finally, note that the other subvectors give roughly the same relative performance, both in Tables 8.8 and 8.12. They are not as good as terms, though the loss of 25-30% is not as bad for term relevance as the 50-60% losses due to the single document approach.

8.3.3. CACM Feedback Results for Combinations

Following the example given in Table 8.11, term relevance results for the CACM collection are shown in Table 8.13 for combined retrieval cases. Various pairs of concept types are shown using equal weights. With inner product similarity, where there is no normalization by vector length, using equal weights can be especially bad. Nevertheless, only when terms and co-citations are combined in such a manner is there an actual drop in performance. When terms and links are combined equally, there is a 13% improvement, indicat-

Table 8.13: Results of CACM Term Relevance Feedback
Using Subvector Combinations
With Equal (E) or Regression (R) Based Weights

Scaled Subvector Coefficients						Weight	Aver.	% Change vs.
tm	au	cr	bc	ln	cc	Scheme	Prec.	Terms Only
1.0							.3839	
.5			.5			E	.3866	+0.7
.5				.5		E	.4352	+13.3
.5					.5	E	.3696	-3.7
.33			.33		.33	E	.3851	+0.3
.17	.17	.17	.17	.17	.17	E	.3845	+0.1
.235				.765		R	.4876	+27.0
.099	.201	.372	.009	.318	0	R	.4979	+29.7

ing that the combination is a good one, and that using identical weights is not a terribly bad assignment of coefficients.

As was shown in Table 8.11, regression based weights perform much better than if the same coefficients are used on every subvector. The two most important subvectors, terms and links, with proper weighting to combine their respective virtues lead to a jump of 27% in average precision. When regression weights are attached to all of the subvectors, the performance increase is almost 30%.

Examining the scaled coefficients used to combine all subvectors does not convey a great deal of information. The coefficients not only reflect the relative performance of subvectors for retrieval but also serve to normalize the inner product computations. It can be inferred, however, that with the other subvectors present $\vec{cc}$ is not really needed and $\vec{bc}$ is probably not either.

All in all, the term relevance behavior for CACM is quite encouraging. The extended vector model seems to be of considerable value in improving retrieval performance.

8.4. Conclusions

In Section 8.1, background work and the basic notions of vector based relevance feedback were reviewed. Equations (8-2) and (8-3) gave two different formula for determining feedback queries, one adding back retrieved documents and the other using available data to arrive at probabilistic weights.

In Section 8.2, the first feedback strategy was employed in a simplified form. Single document feedback of extended vectors was demonstrated both for ISI and CACM collections. The idea of using regression techniques to arrive at coefficients for combined similar-

ity was described. Mild improvements for ISI suggested further testing and so the CACM collection was considered. Regression by leaps and bounds enabled identification of good subvector combinations and the results were a bit more promising than for ISI.

Based on the initial success with such a crude feedback scheme the term relevance scheme was used for a more definitive test. In Section 8.3, with better term weighting values, the CACM collection was utilized to again test the idea of combined similarity. With regression based weights the final result was that almost a 30% improvement over the terms only case was demonstrated.

Apparently, then, these preliminary tests with two collections indicate that extended vectors may significantly aid retrieval performance in a feedback environment. Thus, the extended vector model proposed in Chapter 6, illustrated and partially tested through clustering as described in Chapter 7, now appears to be worthwhile for feedback purposes.

CHAPTER 9

SUMMARY AND CONCLUSIONS

This chapter serves to bring together the findings of the last eight chapters, and to chart the way for future work. It presumes familiarity with the subject matter and discussions that have preceded it, but aims at providing a higher level perspective than was given before. Logically, it complements the introduction of Chapter 1, which should at the very least have been skimmed before this chapter can be appreciated.

Information retrieval as discussed in this thesis fits somewhere between the two fields which consider database management and question answering. Its distinguishing characteristic is that it focuses on problems relating to large numbers of documents. Modern day textual information retrieval systems have made significant contributions in modernizing and speeding up the services provided by traditional libraries; they have also been of value for legal searches, message processing, and other related applications.

Database management systems have become much more powerful in recent years, able to store and interrelate in flexible ways many types of structured data. Advances in query languages enable even casual users to easily express their interests and to retrieve or manipulate entries of interest.

Question answering systems have also developed, along with related areas of artificial intelligence. More detailed knowledge representation schemes have been devised, aimed at thoroughly and accurately organizing the available information.

In a similar vein, this thesis is aimed at encouraging the use of better query and document representation schemes for information retrieval. As distinct from conventional retrieval systems, however, the proposed query and document schemes are totally automatic in nature. In contrast to database systems, the measure of success is how effectively the average user's interests are satisfied, rather than how efficiently results are obtained. Compared to question answering systems where the methodology and accurate processing of a few questions and a few documents is of interest, the emphasis is on handling large numbers of queries and documents. Finally, a user oriented perspective is carried through the entire study in that each theory, model, or method proposed is tested through specially planned experiments.

Previous work in information retrieval has usually adopted one of two main models, which will be termed the Boolean and vector approaches. The Boolean model emphasizes the use of Boolean logic expressions to represent queries. The vector model, on the other hand, focuses on using the terms of documents to construct document representations.

As has been discovered in many other fields of science, two seemingly conflicting theories often both have applicability, and a deeper understanding leads to an approach that combines their virtues. In the realm of query handling, the p-norm theory initially proposed by Wu is a generalization of Boolean and vector techniques. For the problems of document representation, the extended vector technique discussed in this thesis also serve to generalize Boolean and vector approaches, and to include elements relating to the relational database model.

The integrated SMART system developed in connection with these studies thus provides fully automatic processing of queries and documents. Some procedures developed,

however, can also be applied to conventional Boolean systems, to effect improvements when only evolutionary change is permitted.

Along with the SMART programs, four main test collections have been developed and utilized to validate all essential ideas. Evaluation techniques have been adopted to make analysis of large numbers of results simpler. Regression techniques have been applied to a number of tasks where determination of parameters or validation of proposed models were called for.

Now that the orientation of the thesis is understood, attention will be given to the specific problems and solutions devised. First, the query concerns will be discussed, an area where both Boolean and p-norm techniques can be applied.

9.1. Boolean and P-Norm Methods

In conventional retrieval systems, users or their helpers (often called search intermediaries) use the logical connectives AND, OR, and NOT, to combine keywords or other concept identifiers into a Boolean expression. That formula is then used to access an inverted file, and after suitable unions, intersections, and complements of document number lists, a set of documents which logically satisfy the Boolean query is obtained.

To implement this scheme, it is only necessary to know whether each document is or is not indexed by each concept. Further, the retrieval set is sharply defined. Various proposals have suggested that a less strict interpretation would be better.

The p-norm model proposed by Wu allows queries of the form

$$A_a \ AND^p \ B_b$$

to be submitted, where the small letters a and b allow relative query weights to be

specified. Thus, in the above example, the importance of having terms A and B is according to the proportion $a :: b$. The other parameter, p, indicates how strictly the operator is to be viewed. On the one extreme, for $p = 1$, one simply adds together the effects of A and B, as is done for vector handling, and the operator chosen is essentially unimportant. On the other hand, for $p = \infty$, behavior is like that of Boolean logic, i.e., one has the normal understanding of say "A AND B". Thus, the p-norm scheme has the Boolean and vector query methods as special cases.

9.1.1. Analytical and Graphical Insights

Chapter 2 extends Wu's early development with some graphical and analytical studies. Appendix A gives examples of graphs constructed and Appendix B has details of proofs. Both through graphical and analytical insights it can be seen that query weights can lead to "space distortions" for high p-values. Hence, as later reaffirmed through experiments described in Chapter 3, it is inadvisable to use extreme combinations of differing query weights when medium to high p-values are employed.

9.1.2. P-Norm Validation

Chapter 3 is an experimental validation of the p-norm approach, carried out using four main document collections. First it is shown that standard Boolean queries with binary weights on document and query terms can be improved by using lower p-values than infinity. Second, an easy to compute document term weighting formula, using term frequency and inverse document frequency components, is described and shown to yield good improvements over binary document weighting. Third, query weights based on normalized inverse document frequency values are proposed.

Regression analysis shows how various query and document weighting combinations affect performance as the p-value parameter is varied. Lower p-values are generally best. There is a rapid decrease in retrieval effectiveness when query terms are weighted and p-values are increased, but cases with binary query weights are relatively immune to the effect of such increases. For all collections considered, using low p-values, binary query weights, and document weights as described above leads to significant improvements over other schemes for interpreting Boolean queries.

A fifth point about the p-norm validation experiments concerns automatically expanding queries by using OR^p clauses to add in lexically related terms as down weighted partial synonyms for low or medium frequency query elements. Based on the linguistic theory of Mel'chuk et al., preliminary tests with the ISI collection showed that relatively mild improvements might result. Further work is needed to develop a coherent scheme for using linguistic and other information to build thesaurus categories "on the fly" into queries.

The sixth and final point made in Chapter 3 is that by using document weights, low p-values, and good Boolean queries one can expect performance significantly better than that due to conventional Boolean retrieval, and slightly better than that of the vector approach using frequency based weights and cosine correlation.

9.1.3. Automatic Boolean and P-Norm Query Construction

In keeping with the SMART philosophy of automatic processing, the question arises as to whether Boolean or p-norm queries can be mechanically formed. Building Boolean queries by machine would be of value in conventional retrieval environments since naive users could simply submit a paragraph or keyword list and the computer could do the rest; search intermediaries need not be involved. Automatic p-norm queries would be handy

since p-norm queries can do better than either Boolean or vector schemes and since it is difficult to manually form such queries with all the possible weight and p-value options.

The first automatic query construction method derives its inspiration from previous studies of the distribution of term discrimination values as a function of collection frequency. Since low frequency terms need to be shifted toward higher frequencies, as is done in building thesaurus categories, the OR^p operator is used to connect them. High frequency terms should be moved toward lower values, as is done in phrase formation; AND^p connection can implement that transformation. In experiments with two collections, trials with between three and eight different groupings were made. Though using AND^1 throughout was tried in a number of cases, other tests used clausal connectors ranging from OR with high p-value down to OR^1 and then back up to AND with high p-value.

The so called frequency range method did fairly well – better than standard Boolean but not quite as well as vector methods. Note that low p-values were needed to make the scheme operate effectively and so the queries formed would not be applicable to a conventional Boolean environment. Furthermore, deciding on the number of frequency ranges and setting each of the p-values presented a number of parameter estimation problems that are not easily resolved.

An alternate approach, constructing either Boolean or p-norm queries in disjunctive normal form, has wider applicability and is more straightforward to implement. A user need only provide a list of words and an estimate of the number to retrieve. The resultant query is the disjunction of suitable single terms, pairs of terms, or triples. The p-norm version of this technique simply uses a low p-value and weights on document terms to effect the expected improvement over the conventional Boolean form.

In tests with the Medlars collection, the automatic Boolean queries were better than manually formed Boolean queries. With p-values and weights, the automatic queries did about as well as the vector approach. On the other hand, in the much larger INSPEC collection, with very long lists of low quality terms, only with p-values and weights was improvement over the manually formed Boolean queries shown, and that performance was still below that of vector techniques. For more details and discussion of results from using a slightly revised version of the original algorithm, the reader is referred to [Salton, Buckley & Fox 1983].

The automatic Boolean query approach does seem usable for situations where relatively short lists of good index terms are made available or where a naive user is unfamiliar with Boolean methods. When p-norm methods are allowed, however, better results follow, though they are about the same or slightly worse than would be expected from vector methods. Consequently, it seems valuable to enhance a conventional retrieval system with the automatic query construction techniques, but may not be especially useful if one is designing a new system.

9.1.4. Boolean and P-Norm Feedback

Since Boolean and p-norm initial queries can be automatically constructed, the obvious follow up is to carry out Boolean and p-norm feedback. Though several ad hoc approaches to Boolean feedback have been proposed and subjected to small scale tests, no really straightforward technique had been implemented and thoroughly validated. In addition, the useful idea of controlling the form of the query with a user supplied estimate of the desired number retrieved had never been proposed for Boolean feedback.

The disjunctive normal form approach was extended to utilize any available feedback information. For the Medlars collection, feedback processing of Boolean queries led to substantial performance improvements that could be further supplemented by using p-norm interpretation and document term weighting. These results continued for a second feedback iteration, and worked when the initial query was manually or automatically formed.

A major contribution of this thesis is the proposal, implementation, and validation of a fully automatic system for Boolean feedback – something that can be adapted to conventional retrieval systems without a great deal of trouble. That method can be supplemented by allowing the initial query, prior to feedback, to also be automatically constructed. Alternatively, the Boolean feedback approach could be implemented with p-norm interpretation and document term weighting, leading to performance far exceeding that of typical vector searches.

9.2. Extended Vectors

Having demonstrated the value of the p-norm query model, and having applied it to automatic query and feedback construction tasks, the question remaining was whether a similar generalization could be made for documents.

In Boolean systems, users typically can search various fields, asking for a certain author or for matches on a number of keywords. In vector systems, documents are simple vectors, with no indication of the type of each vector concept. The extended vector system, therefore, allows vectors to be split up into a number of subvectors, each to be viewed separately. For retrieval purposes, the similarity of a document to a query is then a linear combination of the similarities of the various subvectors.

9.2.1. Extended Vector Model

Chapter 6 explains various studies relating to enhancing or providing alternatives to term information. The bibliographic coupling measure advanced by Kessler, and the use of co-citation values as developed by Small, are two examples of bibliographic based sources of information. Consequently, one can have subvectors $\vec{tm}$ for terms, $\vec{bc}$ for bibliographic coupling, and $\vec{cc}$ for co-citations. In addition, direct references between documents can be recorded in another type, $\vec{ln}$ for links. Further, database types such as authors can be included in other subvectors – i.e., $\vec{au}$.

To compute the similarity between a query and a document, one can use a linear combination of the similarities for each subvector. This model allows considerable flexibility in that each component of the extended vector can be handled in a different fashion. Given such a scheme the question is whether its adoption will lead to effectiveness improvements in addition to a more elegant description of the retrieval model.

9.2.2. Extended Vector Clustering

One way to make use of the extra information in extended vectors is to cluster those vectors using a combined similarity formula. If the clustering seems better than if only terms are used then use of extended vectors is recommended.

The first issue is how to do such a clustering. The various techniques proposed and used for retrieval are discussed in Chapter 7, including the single link method, which has many nice properties. However, since single link requires $O(n^2)$ operations for n documents, the faster heuristic method developed by Williamson, which only requires $O(n \log n)$ steps, was selected. After suitable algorithm modification and generalization, fast clustering of extended vectors was possible.

To determine whether extended vectors are better than when terms only are used, two types of clustering tests can be made. One is to subjectively compare the resulting classifications, and the other is to objectively compare the effectiveness of carrying out searches over various cluster representatives.

In order to judge the quality of various cluster classifications, a few small subcollections were selected from the full CACM collection. The first test was with the last 55 documents in that collection. Using all subvectors, a fairly good classification emerged.

A more definitive and thorough test was conducted using the 52 highly cited articles that also had at least one entry in the $\vec{cr}$ subvector (which lists *Computing Reviews* categories assigned). The classifications produced by various tests should be fairly easy to interpret to one familiar with important papers published in CACM.

As a background, the articles of interest were clustered via $\vec{cr}$ entries using the single link method, and also using two and one dimensional versions of the multidimensional scaling technique. The single link method did well on grouping articles with high similarity, but not so well with articles not well correlated to others. Multidimensional scaling gave very reasonable classification results.

When applied to the chosen subcollections, the fast clustering procedure provided a different classification for each single subvector and for two cases where mixes of subvectors were utilized. Of the various single subvectors, *Computing Reviews* categories ($\vec{cr}$) gave the best results. Overall, the best behavior seemed to come when $\vec{bc}$ and $\vec{cc}$ were combined with equal weighting.

Thus, use of extended vectors seemed worthwhile, at least to the extent of warranting more objective testing. Hence, the entire collection of 3204 CACM articles was clustered

using various strategies, and a clustered search was run to check the retrieval performance of each case.

Numerous parameters were involved, so the conclusions were not as definitive as hoped for. Nevertheless, some valuable insights were gained, concerning in what order documents should be added to the cluster tree, and what type of weighting should be used for centroid concepts. Of most importance, however, was the tentative conclusion which showed that when a bad scheme for adding documents was not employed, extended vectors could yield slightly better performance than simple term vectors of the same length.

Since clustering techniques have so many confounding factors that can affect results, it was decided to forego further clustering runs and use feedback methods to give further validation of the extended vector model of retrieval.

9.2.3. Extended Vector Feedback

Chapter 8 focuses on feedback methods applied to extended vectors, as distinct from the discussion in Chapter 5 of Boolean and p-norm query feedback. After reviewing various approaches to feedback, two different methods were selected to test the notion of using extended vectors. The idea in each case was to begin with a query containing terms only and to use feedback to construct an extended query vector; the ultimate test was whether the extended query would do better than a feedback query which only included terms.

The first approach was to feed back a single document, the top ranked relevant one, and use it instead of the original query. Single document tests were made with both the ISI and CACM collections. In all cases, the $\vec{tm}$ subvector was the best single one to use. For ISI, $\vec{cc}$ was next best while for CACM $\vec{ln}$ was second.

To really test the value of extended vectors, one must use several subvectors together. However, it is essential that proper coefficients be determined so that a good composite similarity could be computed as a linear combination of the similarities of each subvector. Therefore, a regression was performed against the observed values of relevance, to identify proper coefficients.

For the ISI collection, using regression based coefficients, a net 5% improvement in single document feedback resulted from using terms and co-citations instead of just terms. For the CACM collection, regression coefficients on various groupings of subvectors yielded 10-12% gain.

A second, more accurate feedback approach was used in addition. Term relevance weights were computed, and in all cases were much more effective than those from single document feedback. The $\vec{ln}$ subvector was actually even better than the $\vec{tm}$ subvector in CACM single subvector trials. When combined, $\vec{tm}$ and $\vec{ln}$ subvectors with regression based coefficients led to a 27% improvement over terms alone. Finally, using all CACM subvectors with regression based weights led to an improvement of almost 30% over that due to term relevance feedback of the term subvector alone.

Thus, the extended vector model seems clearly justified, in terms of expected benefits in feedback performance. It is felt that regression methods can be applied to arrive at good coefficients to use in a combined similarity formulation, and that supplementing terms with other types of information will be very useful. Thus, for ISI, terms and co-citations seemed most valuable, while for CACM, terms, links and categories were best.

9.3. Generalizations and Applications

Both the p-norm approach and the extended vector models can be applied to a number of related problems, and generalized into broader methods. P-norm queries can be used in database management systems, as can extended vectors. Inexact matching is very often what is desired when a search is conducted, and the p-norm form is superb for such requests. Perhaps, just as fuzzy set theory was studied and applications proposed in many disciplines, the p-norm interpretation of logical expressions, which seems more robust and general, could be appropriately adapted.

Extended vectors can be used for both textual and database applications. Since constructing a dictionary, for example, of possible values of company earnings is clearly inappropriate, the extended vector model, which allows each domain to be manipulated separately, seems especially useful. The idea implemented in SMART of representing extended vectors as

<concept-type, concept-identifier, weight>

tuples might be combined with the earlier idea [Fox 1981] of having an abstract data type for each attribute of a relation. Thus, a document or other entity type which is to be stored and later retrieved would be described in terms of a number of different concept types. Each could have its own definition and attributes. In some, the concept identifier would point to an entry in a dictionary, while in others the concept identifier would be an actual value in some scalar or more complex domain. Similarity schemes would be available to gauge the degree of match between any two items of each type, and a composite similarity function would be defined to appropriately combine the various subvector results.

Even without such generalizations, many applications of the proposed methods are in order. Most obvious are the use of automatic query construction and feedback methods first for Boolean queries and later for p-norm queries. Initially, some type of front end system could be devised to demonstrate the value of these methods.

The SMART system could certainly be reorganized and applied to many real retrieval tasks. With clustering and cluster search, extended document representations, and p-norm or vector query processing, typical retrieval operations are immediately able to be carried out. P-norm queries might then be applied to complex objects such as reports where textual, numeric, and graphical elements are integrated. It would be interesting to develop feedback for p-norm queries handling complex extended vectors made up of abstract data types, each with its own complex internal structure.

9.4. Conclusion

This thesis has focused on a number of interrelated areas of information retrieval, and has tried to connect them through the common medium of the SMART system, along with appropriate test collections, and various theoretical or abstract models.

It is hoped that some small contribution has been made toward developing a coherent theory of information retrieval, that a workable and useful experimental system will aid others to develop and test their ideas, that newly constructed test collections will find other applications, that proposed practical methods will find their way into commonly available retrieval systems, and that further studies will continue to focus on the many open problems still awaiting investigation in the fertile field of information retrieval.

APPENDIX A

P-NORM CONTOURS

A.1. Introduction to Figures

In order to more clearly understand the behavior of p-norm queries it is useful to examine the figures presented in this appendix. For background information refer to Chapter 2 which deals with p-norm queries and to Section 2.2 which discusses the meaning of these contours through examples and explanations.

Each figure shows a two dimensional cross section of the n-dimensional vector space introduced in Chapter 1. That is, for a Boolean-type query with terms A and B the indexing characteristics of all documents in a collection can be shown with respect to just those two terms. Points on the two dimensional graphs represent documents which each have, using fuzzy set theory terminology, membership function values giving the appropriateness of indexing by A and B.

The figures were chosen to illustrate in general how the various parts of p-norm queries interact and in particular how relative weights on query terms affect the resulting similarity value as p varies in the range $[1, \infty]$. The queries are of form:

$$<A,a> \; OP^p \; <B,b> \qquad \text{(A-1)}$$

where

a = relative weight or importance of term A in the query
OP = operator or connective - either OR or AND
p = p-norm value attached to operator OP
b = relative weight or importance of term B in the query

and, for the sake of simplicity, $b = 1$ (since only the relative weighting between a and b is significant).

Contours shown were prepared using routines "graph" and "plot" available as part of the UNIX[1] operating system. For ease of presentation it was decided that on each graph there would be a number of curves shown such that each point (document) on a given curve yields the same similarity value with respect to the query of interest. Thus, separate curves are given for similarity values 0.0, 0.1, ..., 1.0.[2] What should be carefully considered is the shape of each curve and how that shape is influenced by the various parameters.

The next section lists which figures are shown in the final section of this appendix. The tables given indicate what parameter values were used to prepare each graph, and the graphs are similarly labelled. Though not strictly needed (since OR and AND curves are essentially mirror images), some AND curves for useful cases are listed in Table A.1 and shown in Section A.3.[3] The top half of Table A.2 deals with equal weighting cases for OR queries. Low p-values are most useful, so a number of those are included and only a few examples with $p > 5$ are given: Finally, the bottom part of Table A.2 lists cases with 1:2 and 1:3 relative weight ratios to illustrate interactions betwen p-value and relative weighting choices.

---

[1]UNIX is a trademark of Bell Laboratories.

[2]Specifically, every point in a fine grid laid over the unit square was treated as a document. Using the p-norm query associated with the graph a similarity for each such document was computed and if that was (within a small tolerance) equal to one of the similarity values of interest then the appropriate point was plotted.

[3]When one rotates a graph 180° in order to transform an OR curve to an AND curve, all coordinates and similarities must also be complemented with respect to 1.0. Hence, including AND figures for cases which might occur commonly is done as a courtesy to the reader.

The interested reader is encouraged to examine each of the figures, compare them, and then utilize them for reference in future preparation of p-norm queries.

A.2. List of Figures

Table A.1: Figures for AND queries $<A,a>\ AND^p\ <B;1>$

Figure	P-Value: p	Relative Weight: a
A-1	1	1
A-2	1.5	1
A-3	2	1
A-4	2.5	1
A-5	3	1
A-6	4	1
A-7	5	1
A-8	10	1

Table A.2: Figures for OR queries $<A,a>\ OR^p\ <B,1>$

Figure	P-Value: p	Relative Weight: a
A-9	1	1
A-10	1.5	1
A-11	2	1
A-12	2.5	1
A-13	3	1
A-14	4	1
A-15	5	1
A-16	10	1
A-17	50	1
A-18	1	.5
A-19	2	.5
A-20	5	.5
A-21	1	.33
A-22	2	.33
A-23	5	.33

A.3. Actual Figures

Each graph shows points of form (d_A, d_B), since the two axes are labelled for terms A and B. Similarity contours have an associated similarity value, so the real number given near a curve is connected with all parts of that curve.

Figure A.1: Graph for Query $<A,1>$ AND^1 $<B,1>$

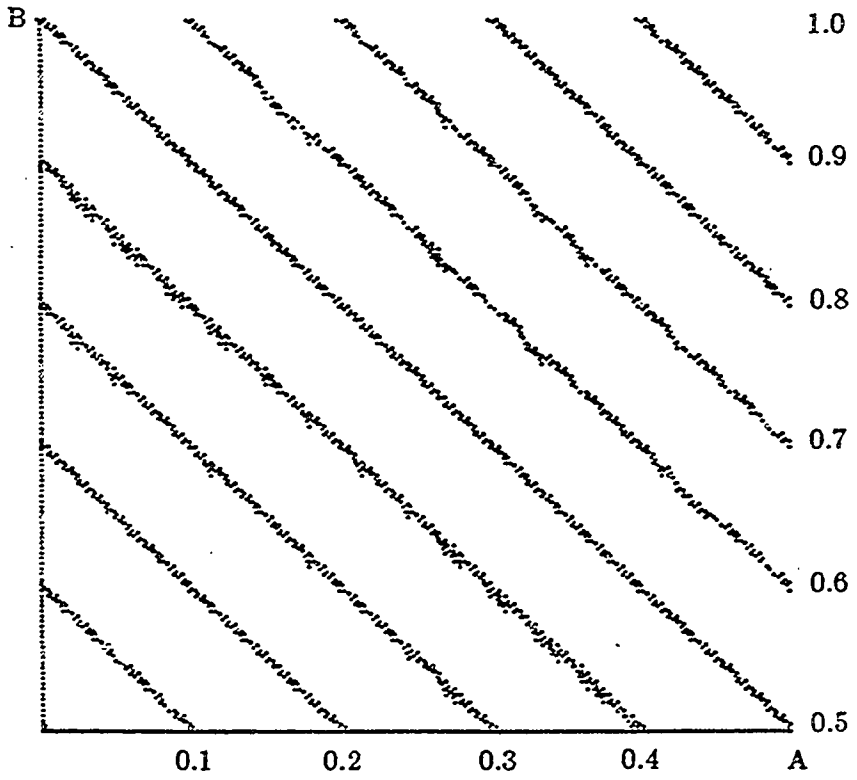


Figure A.2: Graph for Query $<A,1>\ AND^{1.5}\ <B,1>$

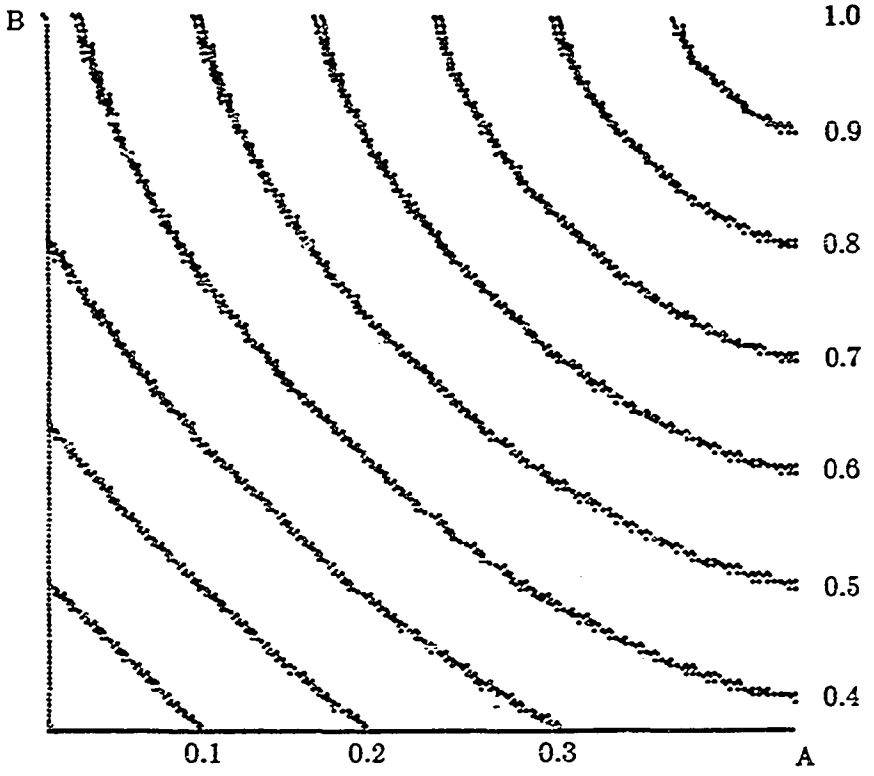


Figure A.3: Graph for Query $<A,1>\ AND^{2}\ <B,1>$

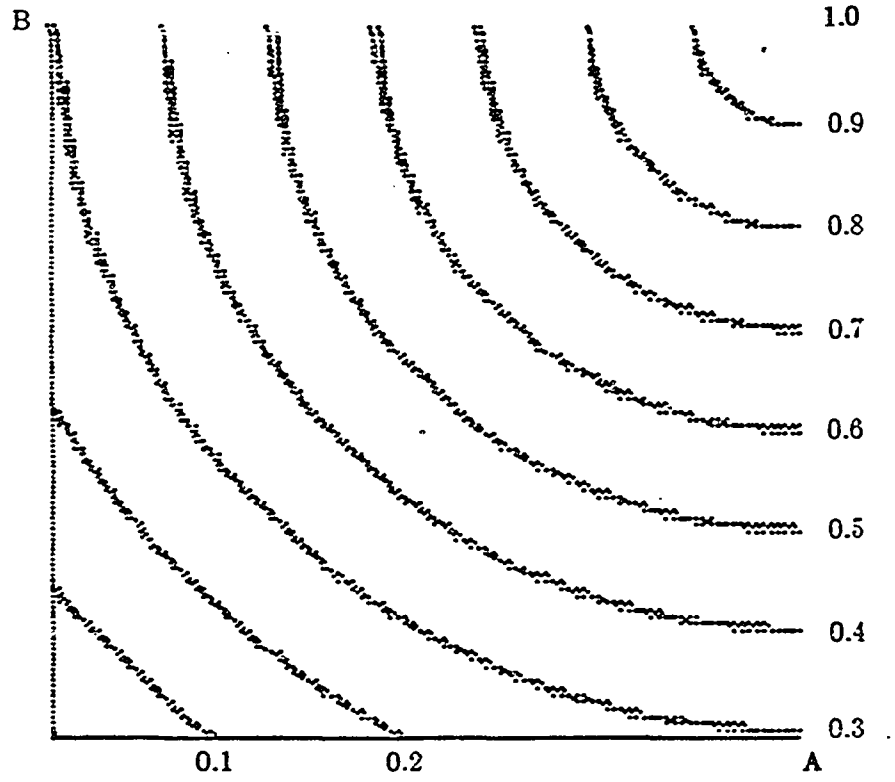


Figure A.4: Graph for Query $<A,1>\ AND^{2.5}\ <B,1>$

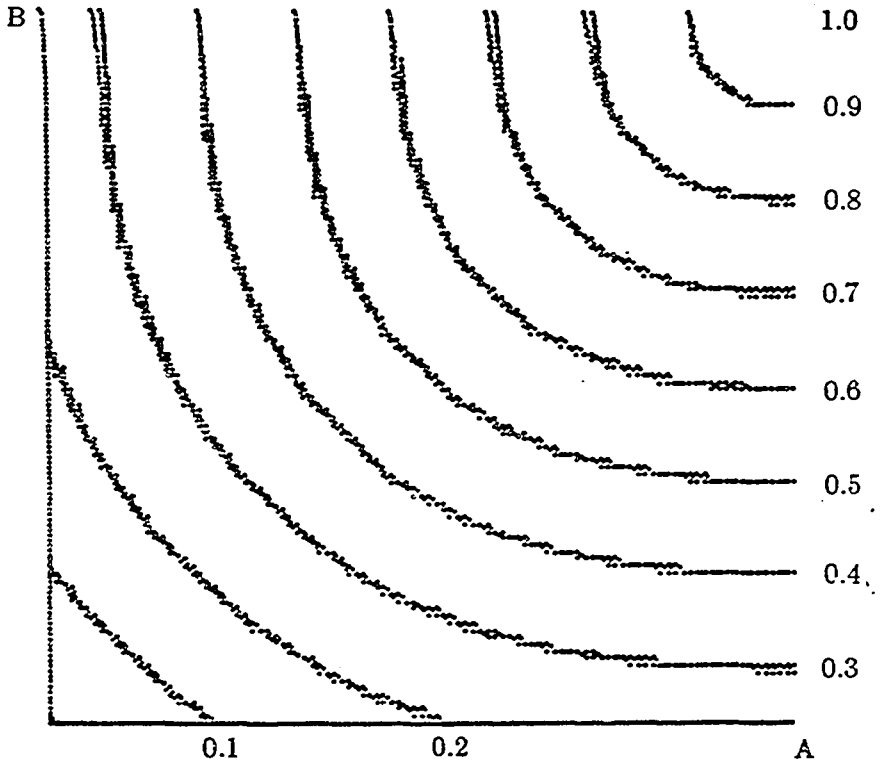


Figure A.5: Graph for Query $<A,1>\ AND^{3}\ <B,1>$

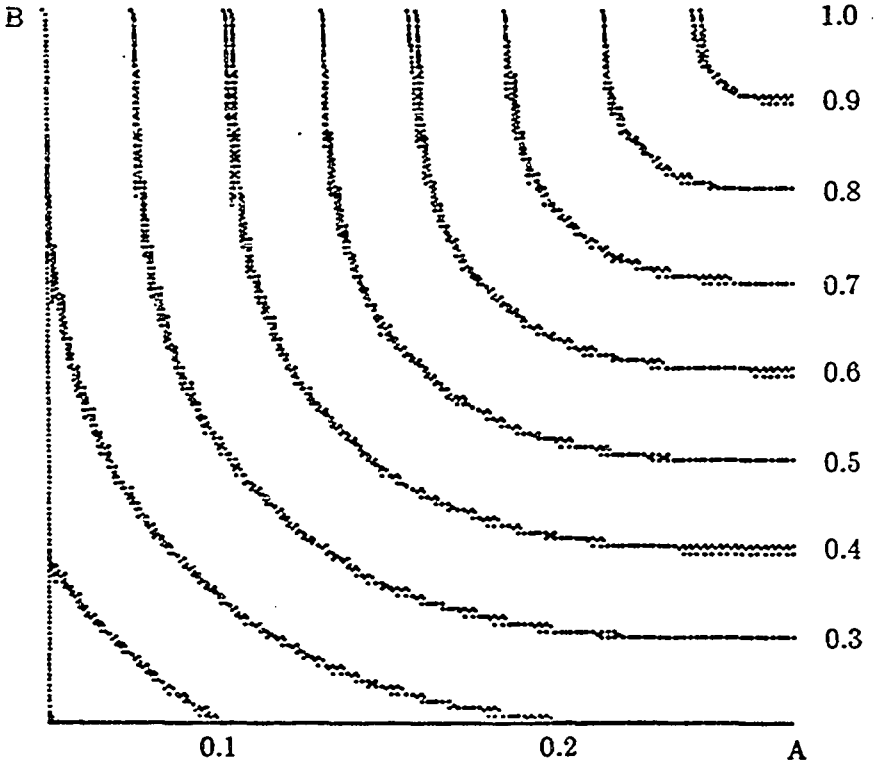


Figure A.6: Graph for Query $<A,1>\ AND^4\ <B,1>$

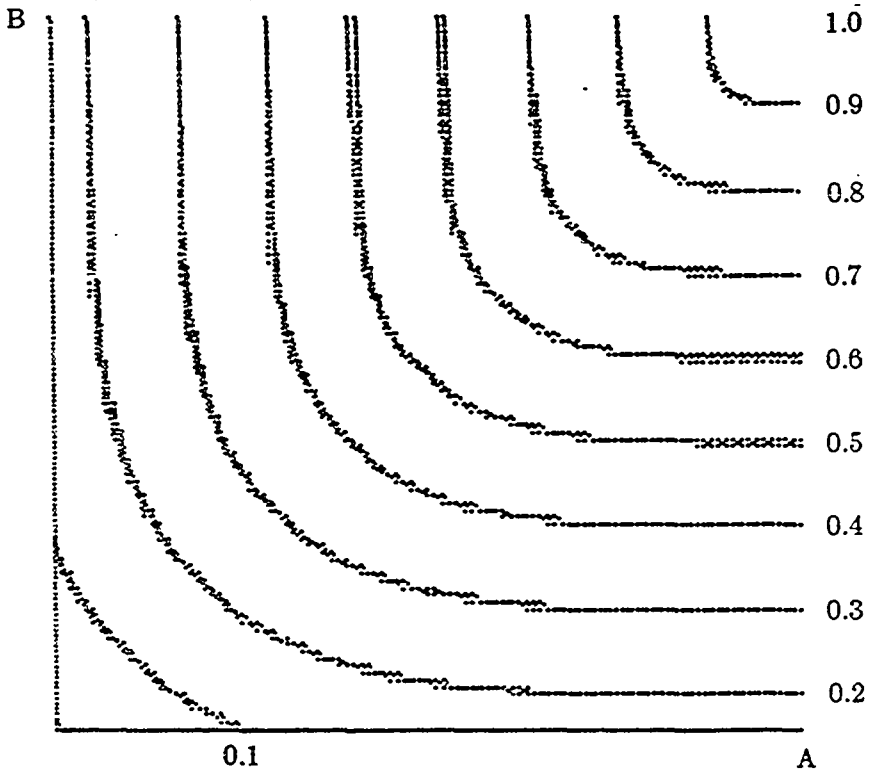


Figure A.7: Graph for Query $<A,1>\ AND^5\ <B,1>$

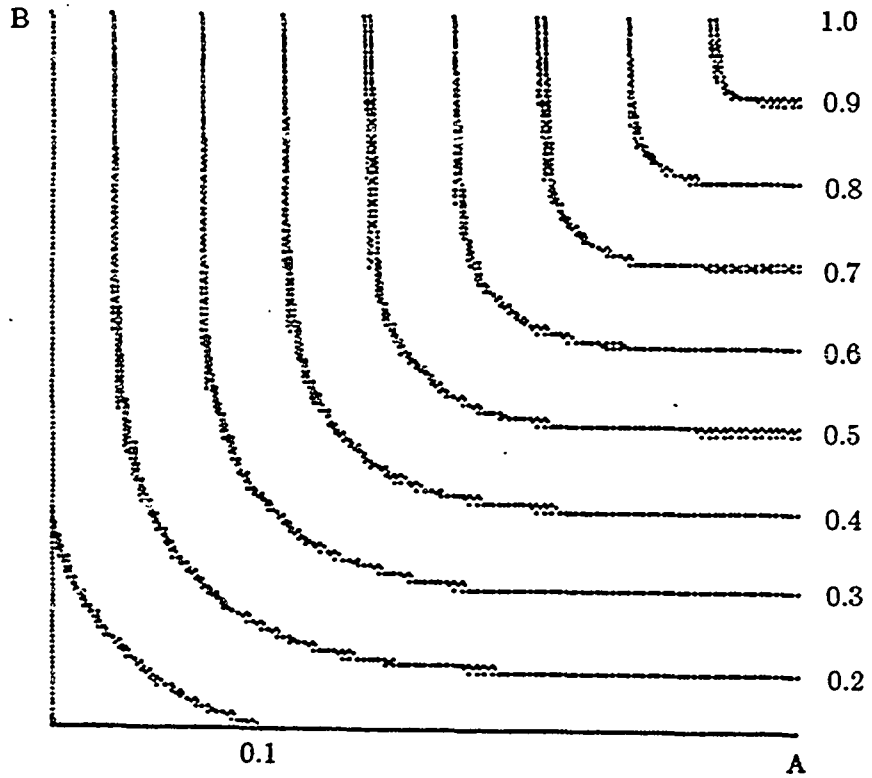


Figure A.8: Graph for Query $<A,1>\ AND^{10}\ <B,1>$

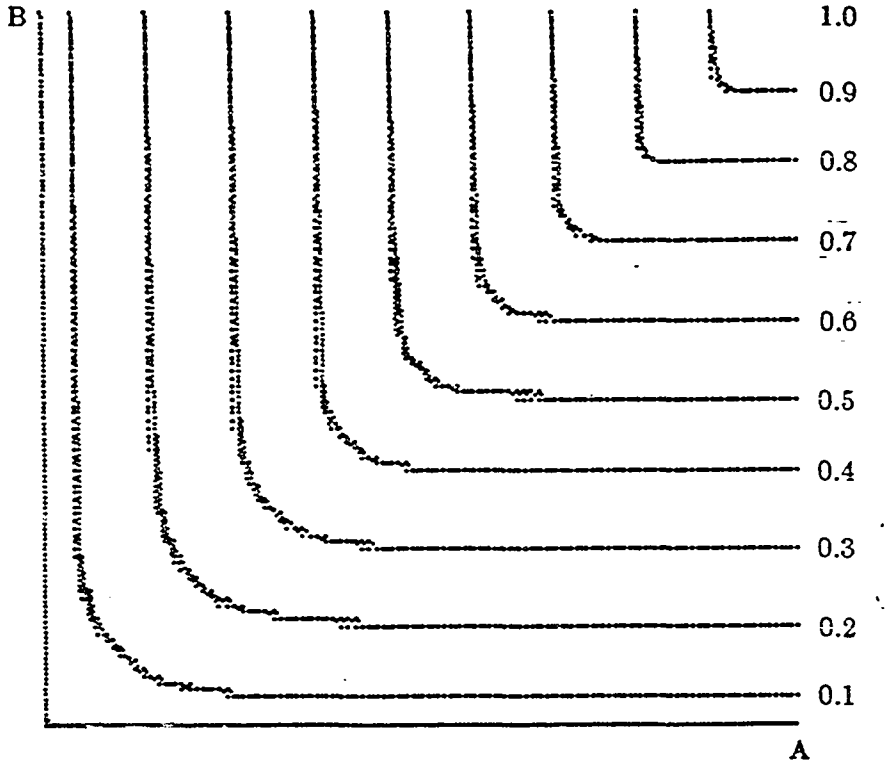


Figure A.9: Graph for Query $<A,1>\ OR^{1}\ <B,1>$

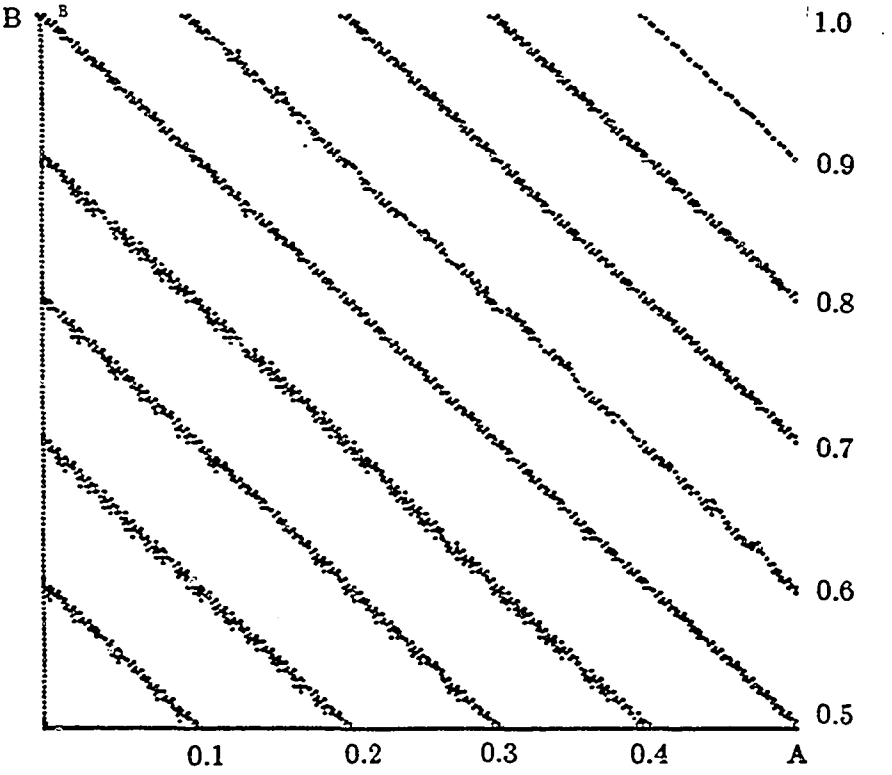


Figure A.10: Graph for Query $<A,1>\ OR^{1.5}\ <B,1>$

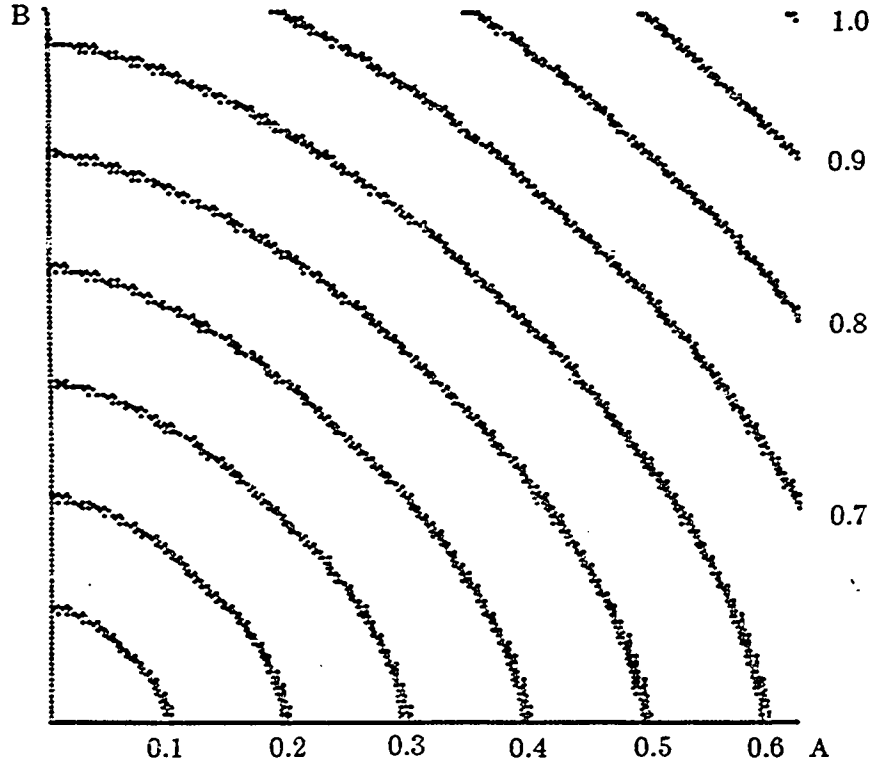


Figure A.11: Graph for Query $<A,1>\ OR^{2}\ <B,1>$

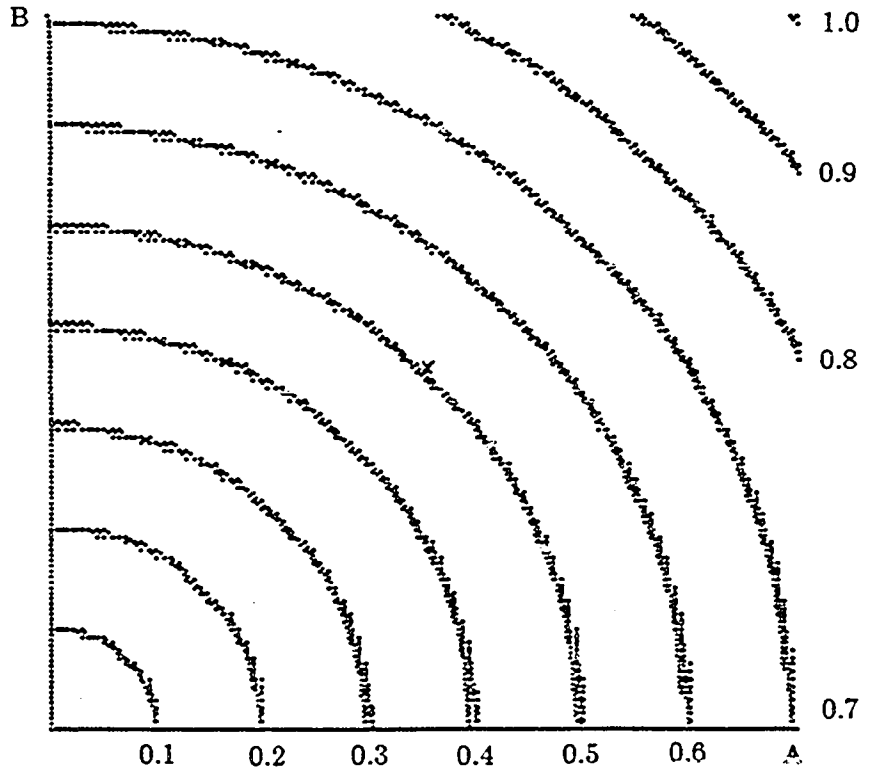


Figure A.12: Graph for Query $<A,1>\ OR^{2.5}\ <B,1>$

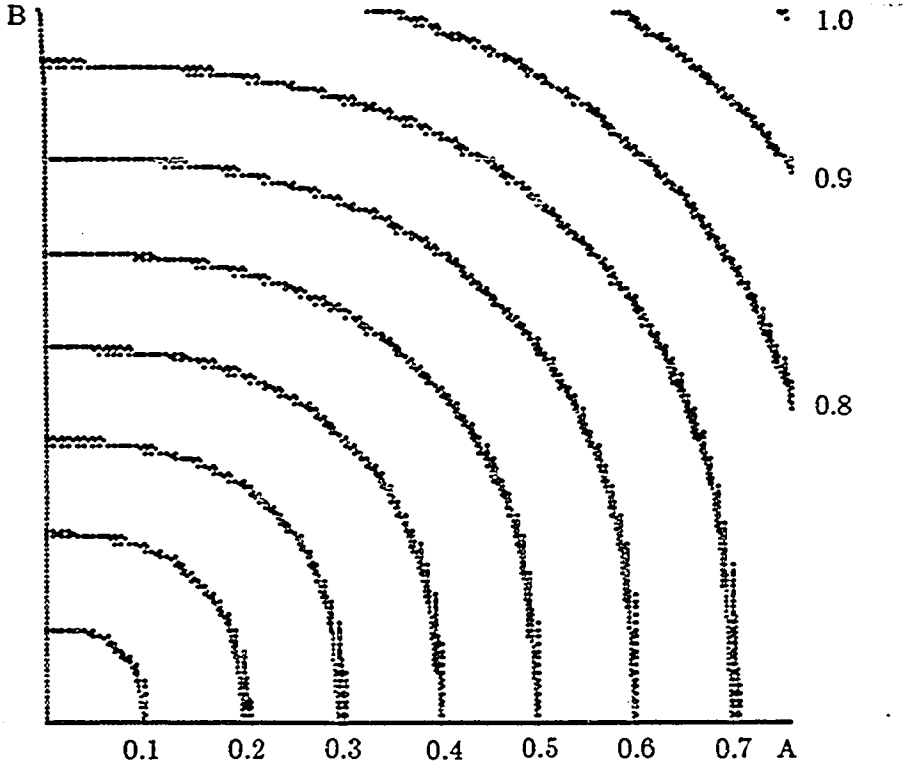


Figure A.13: Graph for Query $<A,1>\ OR^{3}\ <B,1>$

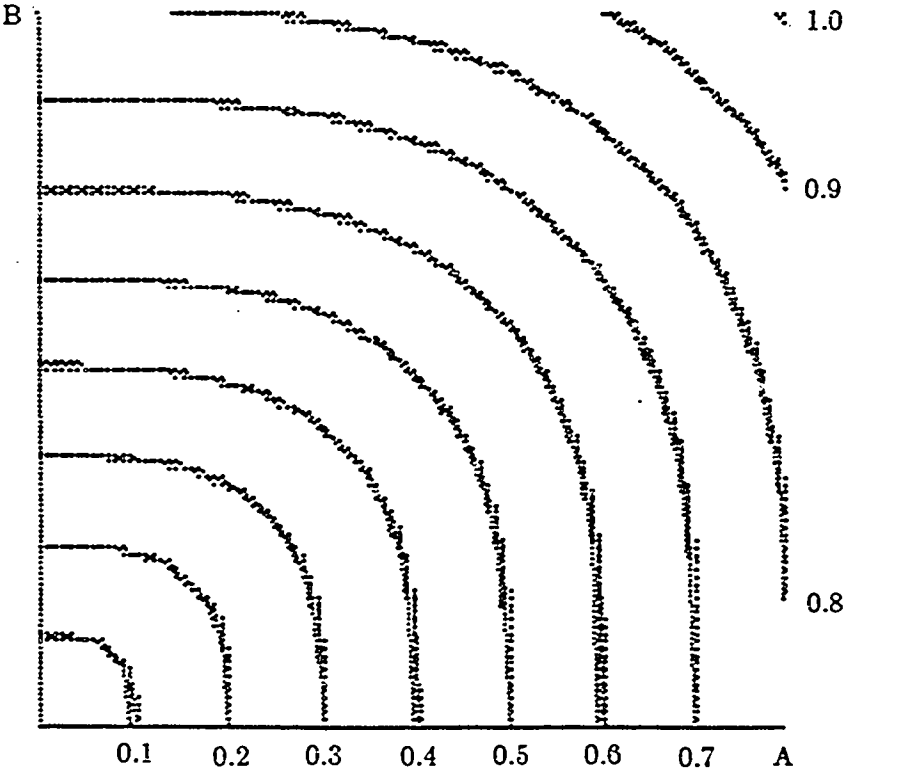


Figure A.14: Graph for Query $<A,1>\ OR^4\ <B,1>$

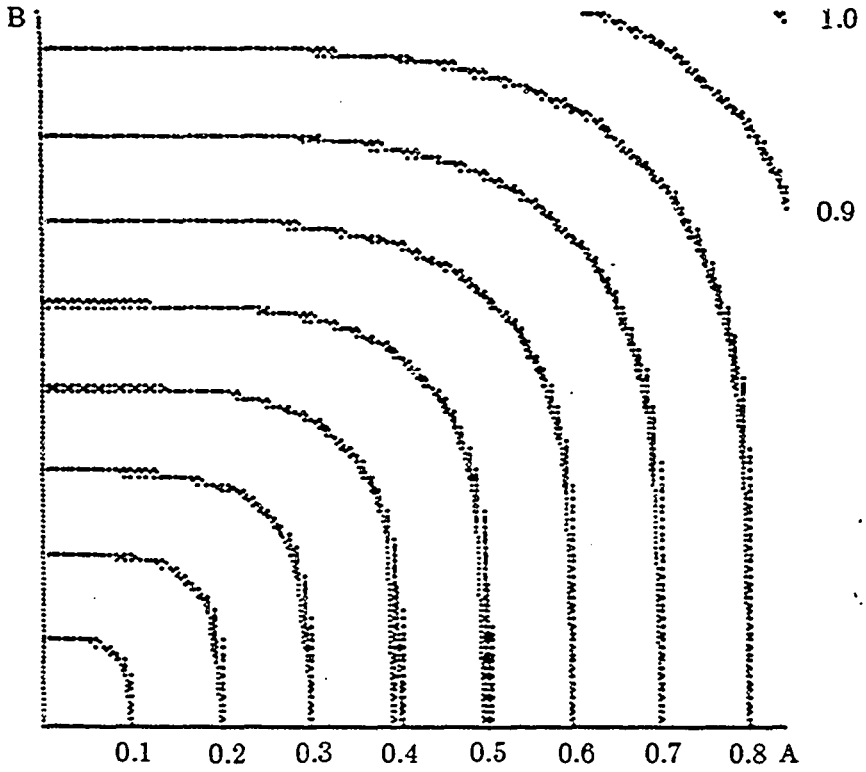


Figure A.15: Graph for Query $<A,1>\ OR^5\ <B,1>$

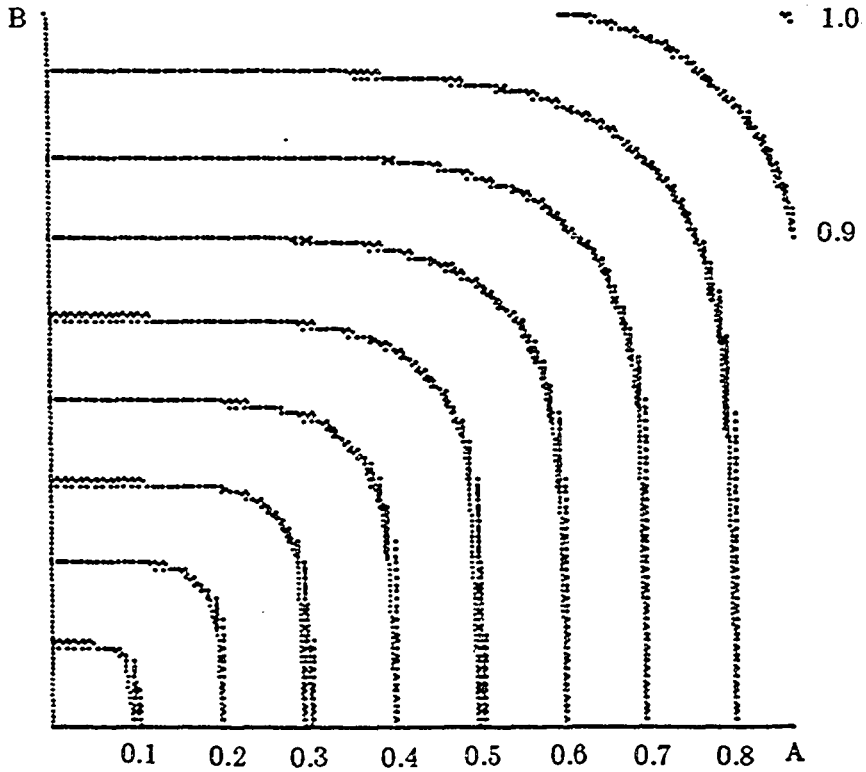


Figure A.16: Graph for Query $<A,1>\ OR^{10}\ <B,1>$

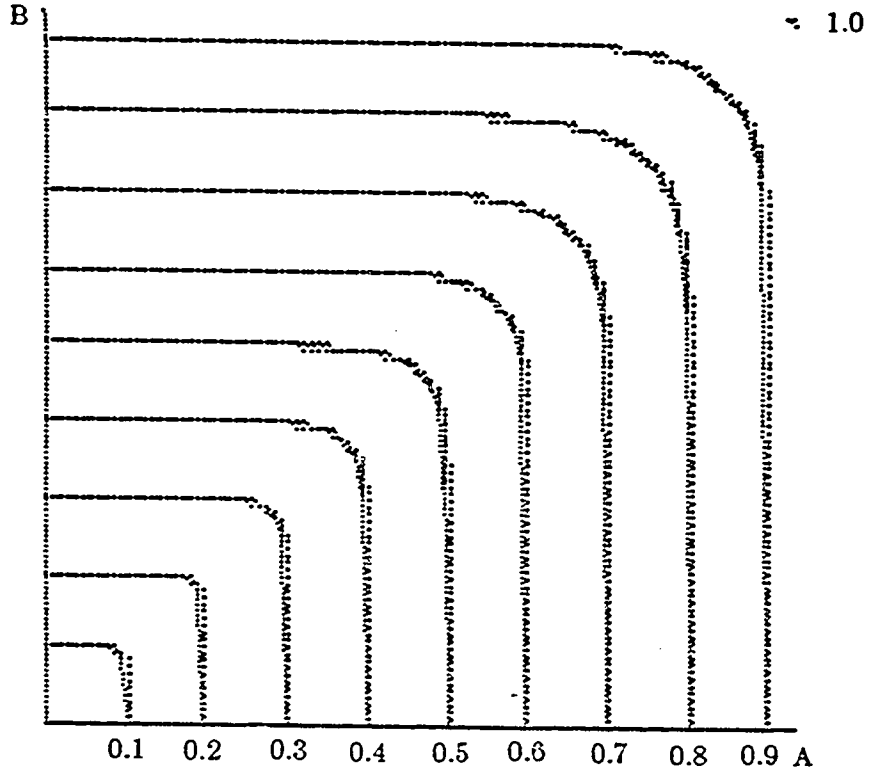


Figure A.17: Graph for Query $<A,1>\ OR^{50}\ <B,1>$

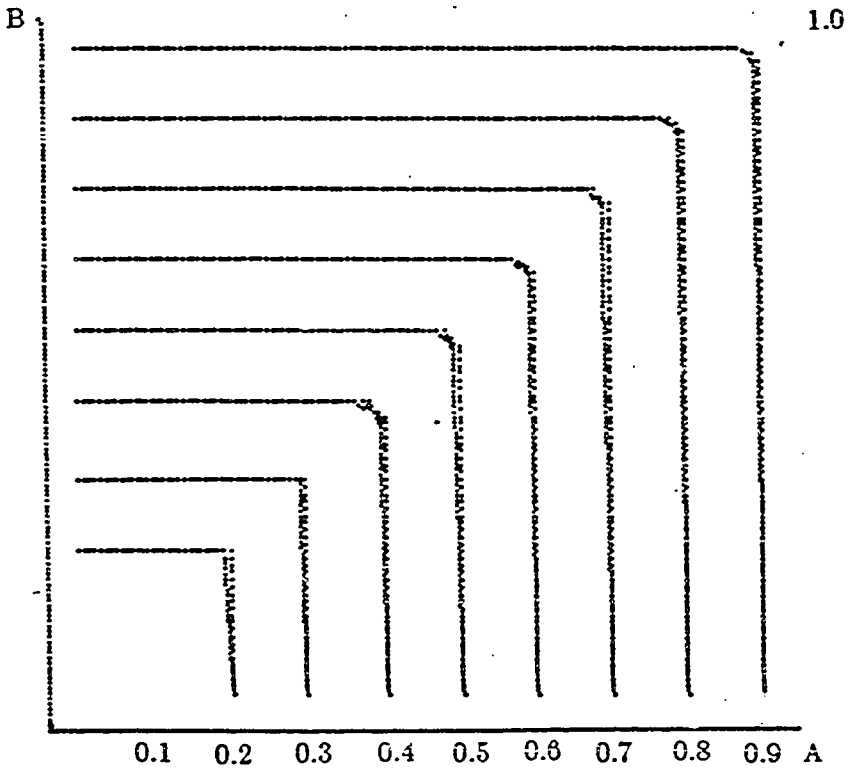


Figure A.18: Graph for Query $<A,.5>\ OR^1\ <B,1>$

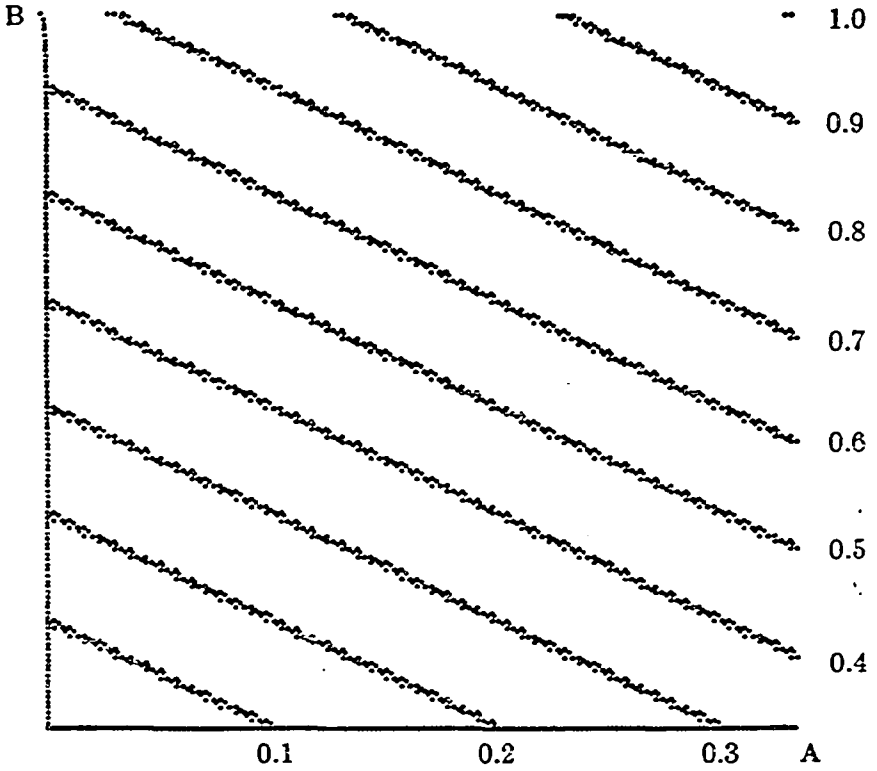


Figure A.19: Graph for Query $<A,.5>\ OR^2\ <B,1>$

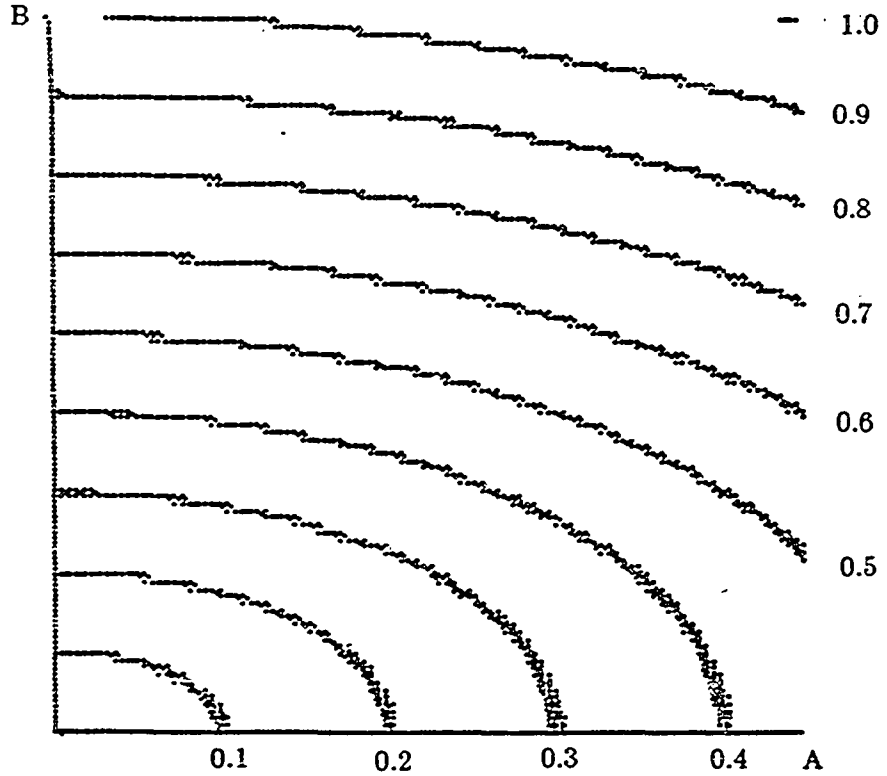


Figure A.20: Graph for Query $<A,.5>\ OR^5\ <B,1>$

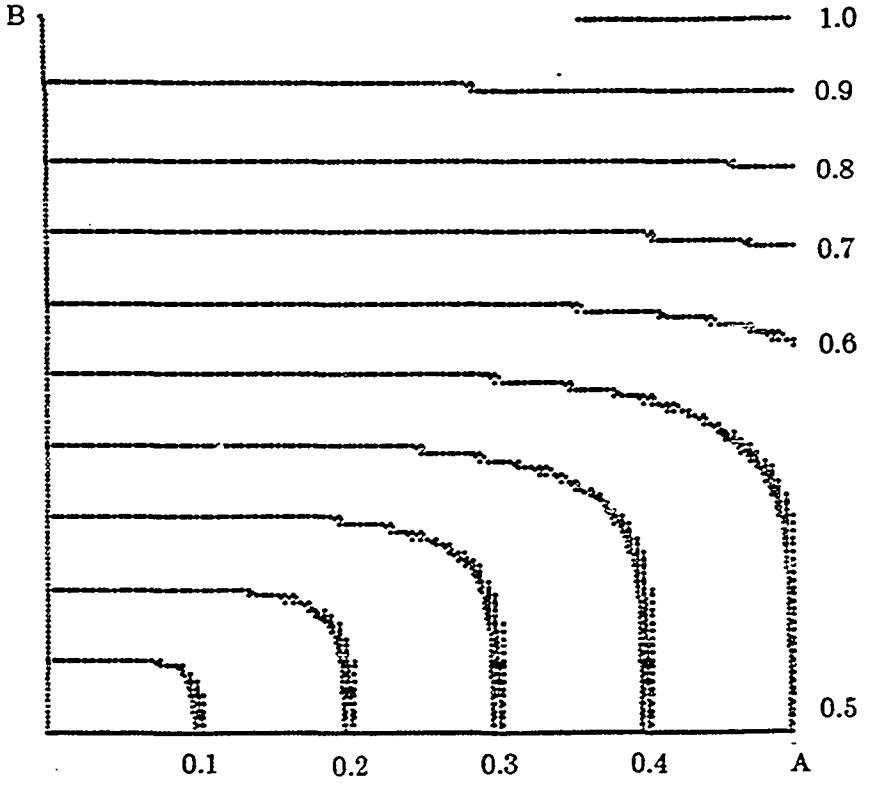


Figure A.21: Graph for Query $<A,.33>\ OR^1\ <B,1>$

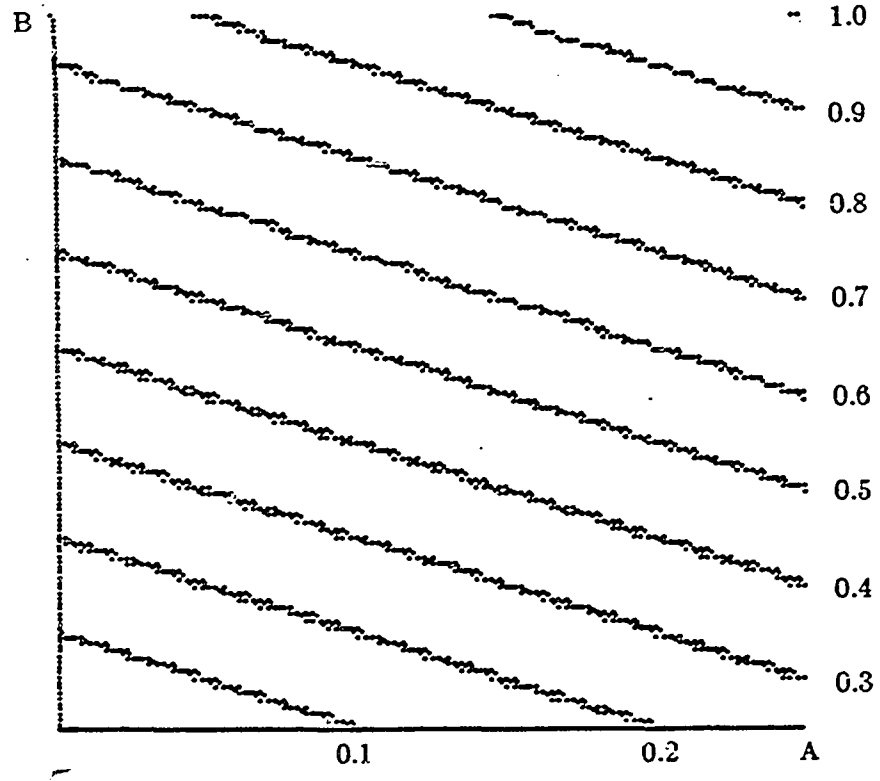


Figure A.22: Graph for Query $<A,.33>\ OR^2\ <B,1>$

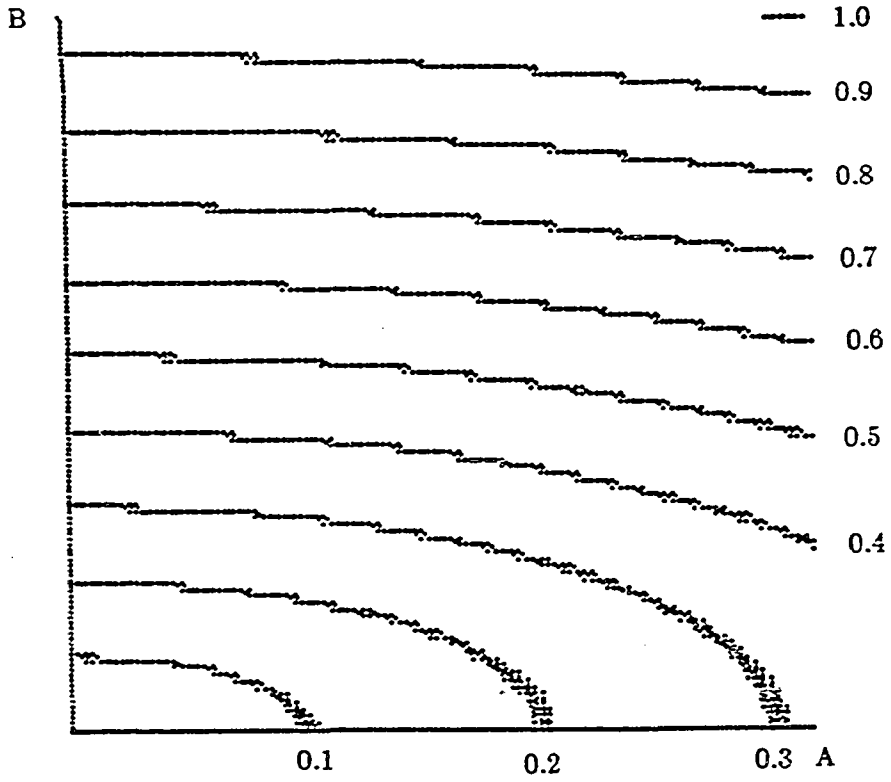
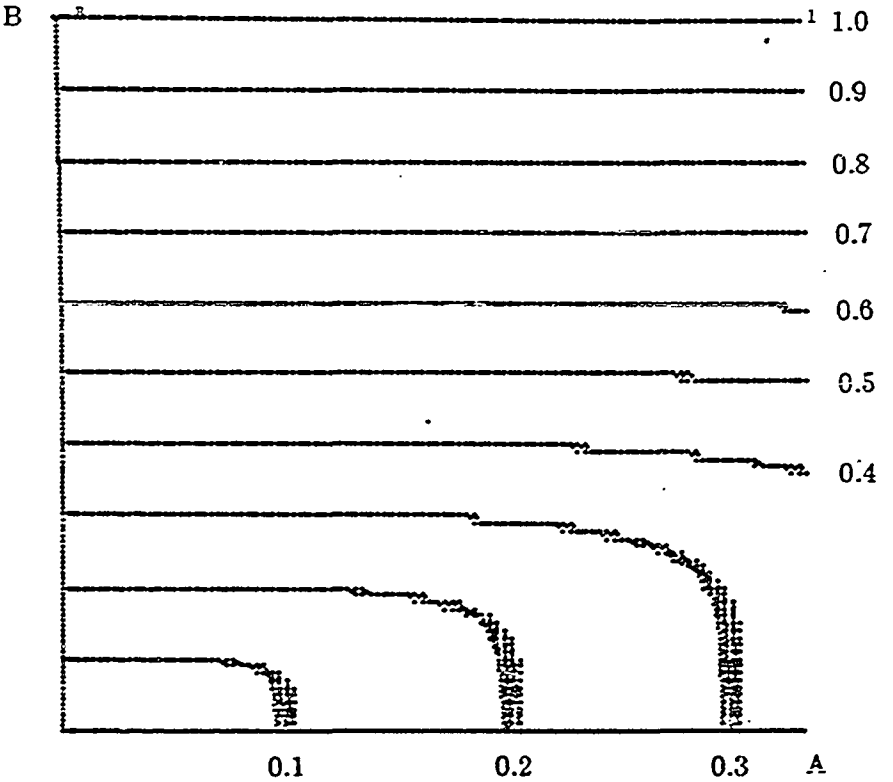


Figure A.23: Graph for Query $<A,.33>\ OR^5\ <B,1>$



APPENDIX B

P-NORM RANKING BEHAVIOR

This Appendix supplements Chapter 2 by providing proofs for equations presented in Section 2.3. As such it aims to clarify the ranking behavior of the p-norm similarity function through an analytical treatment of the basic defining equations of Wu [1981] for particular cases of interest.

In the simple but common case of having a two term query clause with binary query weights (see Sections 2.3.2 and B.1), there is a straightforward relationship between the choice of Boolean operator, the p-value, and the resulting similarity. For queries with more than two terms, things are harder to show. Hence, to lead into more complex equations, Sections 2.3.3 and B.2 deal with the almost trivial case of having constant weight assigned to all document terms.

For the general case of having multi-term queries, relative query weights, and weighted document terms, no simple relationship is possible, as was mentioned in Section 2.3.4. The proof in Section B.3 demonstrates this by counterexample.

Interestingly enough, though, experimental evidence in Chapter 3 shows that it is typically not beneficial to use relative query weights. And when binary query weights are employed, the result for two term queries can be generalized (as stated in Section 2.3.5) to multi-term queries. A short proof of this is given in Section B.4 and an alternative, more geometric but also much longer version, is presented in Section B.5.

B.1. Two Terms, Binary Query Weights

Lemma:

$$X\ AND^{\infty}\ Y \leq X\ AND^{p}\ Y \leq X\ AND^{1}\ Y$$
$$= X\ OR^{1}\ Y \leq X\ OR^{p}\ Y \leq X\ OR^{\infty}\ Y$$

Proof:

(1) Appealing to the p-norm operator definitions, rewrite this, for x being the (non-negative) weight of term X in a document, and y being the (non-negative) weight of term Y, as

$$\min(x,y) \leq 1-\left[\frac{(1-x)^p+(1-y)^p}{2}\right]^{\frac{1}{p}} \leq \frac{x+y}{2} \leq \left[\frac{x^p+y^p}{2}\right]^{\frac{1}{p}} \leq \max(x,y).$$

(2) When $x=y$, substitute, factor, and reduce to x.

(3) Assume, therefore, $x \neq y$ and also assume, without loss of generality, $y>x$. Now it is well known that

$$\min(x,y) \leq \frac{x+y}{2} \leq \max(x,y).$$

(4) Hence, it must first be shown that

$$\frac{x+y}{2} \leq \left[\frac{x^p+y^p}{2}\right]^{\frac{1}{p}} \leq \max(x,y). \qquad \text{(B-1)}$$

Note that if $x=0$, then (B-1) holds. Therefore assume $x>0$. Let $y = x(1+\epsilon)$ for $\epsilon>0$, since $y>x$. Then

$$\frac{x+y}{2} = x\left(1+\frac{\epsilon}{2}\right)$$

and

$$\max(x,y) = y = x(1+\epsilon).$$

Now

$$\left[\frac{x^p + y^p}{2}\right]^{\frac{1}{p}} = \left[\frac{x^p + x^p(1+\epsilon)^p}{2}\right]^{\frac{1}{p}} = x\left[\frac{1+(1+\epsilon)^p}{2}\right]^{\frac{1}{p}}$$

so, after factoring out $x>0$, it needs only be shown that

$$1+\frac{\epsilon}{2} \leq \left[\frac{1+(1+\epsilon)^p}{2}\right]^{\frac{1}{p}} \leq 1+\epsilon.$$

Since all terms are >1, they can be raised to the p^{th} power, yielding the equivalent form

$$(1+\frac{\epsilon}{2})^p \leq \left[\frac{1+(1+\epsilon)^p}{2}\right] \leq (1+\epsilon)^p. \qquad \text{(B-2)}$$

The validity of (B-2) can be shown in two steps:

(a) Assume p is integral and expand each of the terms:

$$1+\sum_{i=1}^{p}(\tfrac{p}{i})\epsilon^i(1/2)^i \leq 1+\sum_{i=1}^{p}(\tfrac{p}{i})\epsilon^i(1/2) \leq 1+\sum_{i=1}^{p}(\tfrac{p}{i})\epsilon^i(1)$$

Each term is of the same form. For $1\leq i\leq p$

$$(1/2)^i \leq 1/2 \leq 1,$$

so the expansion is true and (B-2) holds for integers $p\geq 1$.

(b) Since (B-2) holds for integers $p\geq 1$ and since all of the terms in the expansion above are continuous in p, then (B-2) holds in general for $p\geq 1$. Therefore, it can be concluded that (B-1) holds.

(5) Finally, to complete the proof of the lemma, its needs be shown that

$$\min(x,y) \leq 1-\left[\frac{(1-x)^p+(1-y)^p}{2}\right]^{\frac{1}{p}} \leq \frac{x+y}{2} \qquad \text{(B-3)}$$

Substituting $x' = 1-x$, $y' = 1-y$, (B-3) becomes

$$\min(1-x', 1-y') \leq 1-\left[\frac{x'^p+y'^p}{2}\right] \leq \frac{1-x'+1-y'}{2}.$$

But $0 \leq x,y \leq 1$ so $0 \leq x', y' \leq 1$. Thus the above can be rewritten as

$$1-\max(x', y') \leq 1-\left[\frac{x'^p+y'^p}{2}\right]^{\frac{1}{p}} \leq 1-\frac{x'+y'}{2}.$$

Subtracting 1, multiplying by -1 (and changing the directions of the inequalities) gives

$$\max(x', y') \geq \left[\frac{x'^p+y'^p}{2}\right]^{\frac{1}{p}} \geq \frac{x'+y'}{2}.$$

But that is the same as (B-1), already shown above in part 4 of this proof.

Q.E.D.

B.2. Document Terms with Constant Weight

Lemma:

Given

$$D = (d_0, d_0, \ldots, d_0)$$

and

$$Q_{OP(p)} = <t_1,w_1> OP^p <t_2,w_2> \cdots OP^p <t_n,w_n>,$$

then

$$SIM(Q_{OR(p_1)},D) = SIM(Q_{OR(p_2)},D)$$

$$= SIM(Q_{AND(p_3)},D) = SIM(Q_{AND(p_4)},D)$$

where $1 \leq p_1,p_2,p_3,p_4 \leq \infty$

Proof:

$SIM(Q_{OR(p_i)},D)$ by definition of OR

$= \left[\frac{\sum w_i^{p_1} d_0^{p_1}}{\sum w_i^{p_1}}\right]^{\frac{1}{p_1}}$ but numerator and denominator can be factored to give

$= \frac{\left[\sum w_i^{p_1}\right]^{\frac{1}{p_1}} (d_0^{p_1})^{\frac{1}{p_1}}}{\left[\sum w_i^{p_1}\right]^{\frac{1}{p_1}}}$ which reduces to

$= d_0$

Similarly,

$SIM(Q_{AND(p_1)},D)$ by definition of AND

$= 1 - \left[\frac{\sum w_i^{p_1}(1-d_0)^{p_1}}{\sum w_i^{p_1}}\right]^{\frac{1}{p_1}}$ which can be factored to give

$= 1 - \frac{\left[\sum w_i^{p_1}\right]^{\frac{1}{p_1}} \left[(1-d_0)^{p_1}\right]^{\frac{1}{p_1}}}{\left[\sum w_i^{p_1}\right]^{\frac{1}{p_1}}}$ which reduces to

$= 1-(1-d_0) = d_0$

Q.E.D.

B.3. Similarity Computations for Three Peculiar Cases

Lemma:

There exist query Q', documents D_1, D_2, D_3, and p-values p_1, p_2 such that

$$SIM(Q'_{AND(p_1)}, D_1) < SIM(Q'_{AND(p_2)}, D_1)$$

and

$$SIM(Q'_{AND(p_1)}, D_2) = SIM(Q'_{AND(p_2)}, D_2)$$

and

$$SIM(Q'_{AND(p_1)}, D_3) > SIM(Q'_{AND(p_2)}, D_3).$$

Proof:

(1) Some Q' and $D_1 = (d_1, d_2)$ must be found such that

$$SIM(Q'_{AND(1)}, D_1) < SIM(Q'_{AND(10)}, D_1).$$

Choose

$$Q'_{AND(p)} = <d_1, 0.5> \; AND^p \; <d_2, 1.0>$$

One solution would be to find d_1, d_2 so that

$$SIM(Q'_{AND(1)}, D_1) = 0.4$$

and

$$SIM(Q'_{AND(10)}, D_1) = 0.5.$$

The first constraint gives the equation

$$\frac{\frac{d_1}{2} + d_2}{\frac{3}{2}} = 0.4$$

so,

$$d_1 = \frac{6}{5} - 2\,d_2.$$

Substituting for d_1 in the second constraint equation gives

$$\left[\frac{(\frac{3}{5} - d_2)^{10} + d_2^{10}}{(1/2)^{10} + 1}\right]^{\frac{1}{10}} = 1/2$$

which can be rewritten as

$$d_2^{10} + (0.6 - d_2)^{10} - (1/2)^{10} - (1/2)^{20} = 0.sp.5$$

Clearly, $d_2 \approx 0.5$ since the AND^{10} curve is almost flat in this region. Using iterative methods yields $d_2 = 0.50004875$ which gives an error of only -1×10^{-9}. Hence, $D_1 \approx (0.2, 0.5)$ will behave as desired.

(2) Next, $D_2 = (d_1, d_2)$ must be found such that

$$SIM(Q'_{AND(1)}, D_2) = SIM(Q'_{AND(10)}, D_2) = 0.5.$$

The obvious guess is $d_1 = d_2 = 0.5$. Substituting, one sees that the above equality holds.

(3) Finally, $D_3 = (d_1, d_2)$ must be found such that

$$SIM(Q'_{AND(1)}, D_3) > SIM(Q'_{AND(10)}, D_3).$$

One solution is to have

$$SIM(Q'_{AND(1)}, D_3) = 0.6$$

and

$$SIM(Q'_{AND(10)}, D_3) = 0.5.$$

The first constraint yields

$$\frac{\frac{d_1}{2} + d_2}{\frac{3}{2}} = 0.6$$

so,

$$d_1 = \frac{9}{5} - 2\,d_2$$

Substituting in the second constraint equation gives

$$\left[\frac{(\frac{9}{10} - d_2)^{10} + d_2^{10}}{(1/2)^{10} + 1}\right]^{\frac{1}{10}} = 1/2$$

or,

$$d_2^{10} + (0.9 - d_2)^{10} - (1/2)^{10} - (1/2)^{20} = 0$$

An iterative solution leads to d_2=0.4933274, which yields an error of only 1×10^{-9}.

Hence $D_3 \approx$(0.814,0.493) will behave as desired.

All three points, D_1, D_2, and D_3, have been materialized as required.

Q.E.D.

B.4. Short Proof of OR Query Similarity Ranking

Theorem:

$$SIM(Q_{CR(p_1)}, D) \leq SIM(Q_{OR(p_2)}, D)$$

where

$$1 \leq p_1 \leq p_2 \leq \infty$$

and

$$Q_{OR(p)} = t_1 \; OR^p \; t_2 \; OR^p \; \cdots \; OR^p \; t_n$$

Proof:

Consider surfaces in n-space such that

$$SIM(Q_{OR(p)}, D) = S_0 \text{ for } 0 < S_0 < 1.$$

Note that one can safely ignore S_0=0 and S_0=1 since the lemma above then becomes trivial. Similarly, it is only necessary to consider cases where $n > 1$.

Points on the S_0 surface represent document-query combinations that all yield the same similarity. Therefore, examine the variation of document weights with respect to p-values. If increasing p-value forces a decrease in document weight for each term of our space, in order to maintain constant similarity, then increasing p-value with constant document weights would cause an increase in similarity. Thus, to show the lemma, one need only prove, for $1 \leq j \leq n$,

$$\frac{\partial d_j}{\partial p} \leq 0.$$

That can be done as follows:

Since

$$SIM(Q_{OR(p)}, D) = S_0$$

then

$$\partial(SIM) = 0.$$

In order to apply the chain rule, define functions $g()$ and $h()$ such that

$$SIM = f(g,h) = f(g, \frac{1}{p}) = g^{\frac{1}{p}}$$

so

$$h(p) = \frac{1}{p}.$$

Then,

$$\partial h = -\frac{1}{p^2}\,\partial p$$

$$g(d_j,p) = \frac{1}{n}\sum d_i^p = \frac{1}{n}\sum_{i\neq j} d_i^p + \frac{1}{n} d_j^p$$

and

$$\partial g = \frac{1}{n}\sum_{i\neq j} d_i^p \ln d_i\,\partial p + \frac{1}{n}\partial(d_j^p)$$

but

$$\partial(d_j^p) = p\, d_j^{p-1}\,\partial d_j + d_j^p \ln d_j\,\partial p.$$

Hence,

$$\partial g = \frac{1}{n}\sum d_i^p \ln d_i\,\partial p + \frac{p}{n} d_j^{p-1}\partial d_j.$$

Now, SIM is held constant, so

$$0 = \partial(SIM) = \frac{\partial f}{\partial g}\partial g + \frac{\partial f}{\partial h}\partial h$$

$$= \left[\frac{1}{p} g^{(\frac{1}{p}-1)}\right]\left[\frac{1}{n}\sum d_i^p \ln d_i \partial p + \frac{p}{n} d_j^{p-1}\partial d_j\right] + \left[g^{\frac{1}{p}} \ln g\right]\left[-\frac{1}{p^2}\partial p\right].$$

Solving for $\frac{\partial d_j}{\partial p}$

$$\frac{\partial d_j}{\partial p} = \frac{(g^{\frac{1}{p}})(\ln g)(\frac{1}{p^2}) - (\frac{1}{p}\cdot g^{\frac{1}{p}}\cdot g^{-1})(\frac{1}{n}\sum d_j^p \ln d_j)}{(\frac{p}{n} d_j^{p-1})(\frac{1}{p}\cdot g^{\frac{1}{p}}\cdot g^{-1})}$$

cancelling and multiplying numerator and denominator by $n \cdot g$,

$$= \frac{\frac{n}{p}\cdot g \ln g - \sum d_i^p \ln d_i}{p\, d_j^{p-1}}$$

substituting for g,

$$= \frac{(\frac{n}{p})(\frac{1}{n}\sum d_i^p)\ln(\frac{1}{n}\sum d_i^p) - \sum d_i^p \ln d_i}{p\, d_j^{p-1}}$$

factoring,

$$= \frac{\sum d_i^p\left[\frac{1}{p}\ln(\frac{1}{n}\sum d_k^p) - \ln d_i\right]}{p\, d_j^{p-1}}$$

combining ln in the numerator and substituting in the SIM definition,

$$\frac{\partial d_j}{\partial p} = \frac{\sum(d_i^p \ln \frac{SIM}{d_i})}{p d_j^{p-1}}$$

Now, $d_i \geq SIM$, so $\ln \frac{SIM}{d_i} < 0$ if $d_i > 0$. Also, note that $d_i \ln \frac{SIM}{d_i} = 0$ if $d_i = 0$.

Consequently,

$$\frac{\partial d_j}{\partial p} \leq 0$$

Q.E.D.

B.5. Geometric Proof of OR Query Similarity Ranking

Theorem:

$$SIM(Q_{OR(p_1)}, D) \leq SIM(Q_{OR(p_2)}, D)$$

where

$$1 \leq p_1 \leq p_2 \leq \infty$$

and

$$Q_{OR(p)} = t_1 \; OR^p \; t_2 \; OR^p \; \cdots \; OR^p \; t_n$$

Proof:

Several Lemmas will lead to the main result. Consider surfaces of points in n-space, each point representing a document. A surface is defined by

$$SIM(Q_{OR(p)}, D) = S_0$$

As in Section B.5 above, assume $n \geq 2$ and $0 < S_0 < 1$.

Lemma 1:

The surface for

$$SIM(Q_{OR(1)}, D) = S_0 = SIM(Q_{AND(1)}, D)$$

is a hyperplane through

$$D_0 = (S_0, \cdots, S_0)$$

with the perpendicular being the line

$$L = [(0, \cdots, 0), (1, \cdots, 1)].$$

Proof:

(1) The point $(S_0, \cdots, S_0)$ can be shown to be on the surface as follows:

$$SIM(Q_{OR(1)}, D) = \frac{d_1 + \cdots + d_n}{n} = SIM(Q_{AND(1)}, D) = S_0$$

If $d_1 = d_2 = \cdots = d_n = S_0$, then

$$\frac{S_0 + \cdots + S_0}{n} = S_0$$

and so $(S_0, \cdots, S_0)$ is on the surface.

(2) The equation

$$\frac{1}{n}(d_1 + \cdots + d_n) = S_0$$

is the equation of a hyperplane in n-space.

(3) The line $[(0, \cdots, 0), (1, \cdots, 1)] = L$ has equations $d_1 = d_2 = \cdots = d_n$. From calculus, the equation of the normal line to a surface at point $(d_{o1}, \cdots, d_{on})$ is

$$\frac{d_1 - d_{o1}}{\frac{\partial SIM}{\partial d_1}} = \frac{d_2 - d_{o2}}{\frac{\partial SIM}{\partial d_2}} = \cdots = \frac{d_n - d_{o3}}{\frac{\partial SIM}{\partial d_n}}$$

But, for this hyperplane and perpendicular

$$\frac{\partial SIM}{\partial d_i} = 1 \text{ and } d_{io} = S_0$$

so, $d_1 - S_0 = \cdots = d_n - S_0$ which yields the desired result for L.

Q.E.D.

Lemma 2:

The above mentioned hyperplane is the tangent plane at $(S_0, \cdots, S_0)$ to surfaces $SIM(Q_{OR(p)}, D) = S_0$ and to surfaces $SIM(Q_{AND(p)}, D) = S_0$ when $1 \leq p < \infty$. If $n, p > 1$ then the only intersection is the point $(S_0, \ldots, S_0)$. Furthermore, the surfaces for $Q_{OR(p)}$ lie totally on the $(0, \cdots, 0)$ side of the hyperplane, while the surfaces for $Q_{AND(p)}$ lie totally on the $(1, \cdots, 1)$ side of the hyperplane.

Proof:

(1) It is only necessary to demonstrate the desired facts for $Q_{OR(p)}$; the other results for $Q_{AND(p)}$ follow by symmetry. For the sake of brevity, call the $Q_{OR(p)}$ query Q_0.

(2) To find the tangent plane, use partial differentials and the chain rule:

$$\frac{\partial SIM}{\partial d_j} = \frac{\partial SIM}{\partial f} \cdot \frac{\partial f}{\partial d_j}$$

where

$$SIM = (n^{-\frac{1}{p}})(\sum d_i^p)^{\frac{1}{p}} = (n^{-\frac{1}{p}})(f)^{\frac{1}{p}}$$

for

$$f = \sum d_i^p.$$

Now,

$$\frac{\partial SIM}{\partial f} = (n^{-\frac{1}{p}})(\tfrac{1}{p})(f)^{(\frac{1}{p}-1)} = \left[\frac{1}{n^{\frac{1}{p}} \cdot p}\right](\sum d_i^p)^{(\frac{1}{p}-1)}$$

and,

$$\frac{\partial f}{\partial d_j} = p\ d_j^{p-1}$$

Hence,

$$\frac{\partial SIM}{\partial d_j} = \left[\left[\frac{1}{n^{\frac{1}{p}} \cdot p}\right](\sum d_i^p)^{(\frac{1}{p}-1)}\right](p\ d_j^{p-1}) = \left[\frac{d_j^{p-1}}{n^{\frac{1}{p}}}\right](\sum_{i=1}^{n} d_i^p)^{(\frac{1}{p}-1)}$$

(3) From calculus, the equation of the tangent plane to surface $F(x_1, \cdots, x_n)=0$ at $(x_{o1}, \cdots, x_{on})$ is

$$\frac{\partial F}{\partial x_1}(x_1-x_{o1}) + \cdots + \frac{\partial F}{\partial x_n}(x_n-x_{on}) = 0.$$

Now,

$$F(d_1, \cdots, d_n) = SIM(Q_0, D)-S_0 = 0$$

and the point of interest is $(S_0, \cdots, S_0)$, so the tangent plane has equation

$$\sum_{j=1}^{n} \frac{\partial SIM}{\partial d_j}(d_j-S_0) = 0.$$

Substituting the result of (2) for $\frac{\partial SIM}{\partial d_j}$ gives

$$\sum_{j=1}^{n}\left[\frac{d_j^{p-1}}{n^{\frac{1}{p}}}\right](\sum_{i=1}^{n} d_i^p)^{(\frac{1}{p}-1)}(d_j-S_0)=0=\frac{i}{n^{\frac{1}{p}}}(\sum_{i=1}^{n} d_i^p)^{(\frac{1}{p}-1)}\sum_{j=1}^{n} d_j^{p-1}(d_j-S_0).$$

Now, from Lemma 1 above, when $p=1$ the surfaces and hyperplane are all coincident. Hence nothing more need be proven. When $1<p<\infty$, the above has solution

$$d_1=S_0,\ d_2=S_0,\ \cdots,\ d_n=S_0,$$

so hyperplane

$$d_1=d_2=,\ \cdots,=d_n=S_0$$

is the tangent plane at $(S_0, \cdots, S_0)$.

(4) The surface and tangent plane intersect at $(S_0, \cdots, S_0)$. To show that the surface of

$$SIM(Q_0, D) = S_0$$

lies on the $(0, \cdots, 0)$ side of the hyperplane, it is necessary to give a detailed argument based on slopes. Steps (a) through (d) below accordingly conclude the proof of Lemma 2.

(a) As mentioned earlier, assume $n, p>1$.

(b) Select $1\leq j,k\leq n$ and consider $\frac{\partial d_j}{\partial d_k}$. Do implicit differentiation.

$$SIM = (n^{-\frac{1}{p}})(\sum d_i^p)^{\frac{1}{p}} = (n^{-\frac{1}{p}})f^{\frac{1}{p}} \text{ where } f = \sum d_i^p.$$

Then

$$\frac{\partial SIM}{\partial f} = (\frac{1}{n^{\frac{1}{p}}\cdot p})(\sum d_i^p)(\frac{1}{p}-1).$$

Consider d_j as a function of d_k,

$$\frac{\partial f}{\partial d_k} = \frac{\partial}{\partial d_k}(d_j^p) + \frac{\partial}{\partial d_k}(d_k^p)$$

$$= (p\ d_j^{p-1})\frac{\partial d_j}{\partial d_k} + (p\ d_k^{p-1}).$$

Now,

$$\frac{\partial}{\partial d_k}(SIM) = 0.$$

Applying the chain rule,

$$\left[\frac{1}{n^{\frac{1}{p}} \cdot p}(\sum d_i^p)^{(\frac{1}{p}-1)}\right]\left[(p\ d_j^{p-1})\frac{\partial d_j}{\partial d_k} + p\ d_k^{p-1}\right] = 0.$$

When $d_1 = d_2 = \cdots = d_n = 0$, similarity is 0 for any query, so that document point can be ignored. Consider instead the righthand factor:

$$\left[(p\ d_j^{p-1})\frac{\partial d_j}{\partial d_k} + p\ d_k^{p-1}\right] = 0$$

which implies that

$$\frac{\partial d_j}{\partial d_k} = -\left[\frac{d_k^{p-1}}{d_j^{p-1}}\right] = -\left[\frac{d_k}{d_j}\right]^{p-1}.$$

(c) For $n=2$ the situation is illustrated by Figure B.2 and Table B.1.

(d) Finally, prove for $n, p > 1$ that

$$SIM(Q_0, D) = S_0$$

lies on the $(0, \cdots, 0)$ side of the hyperplane.

Figure B.2: Illustration of Surface and Plane

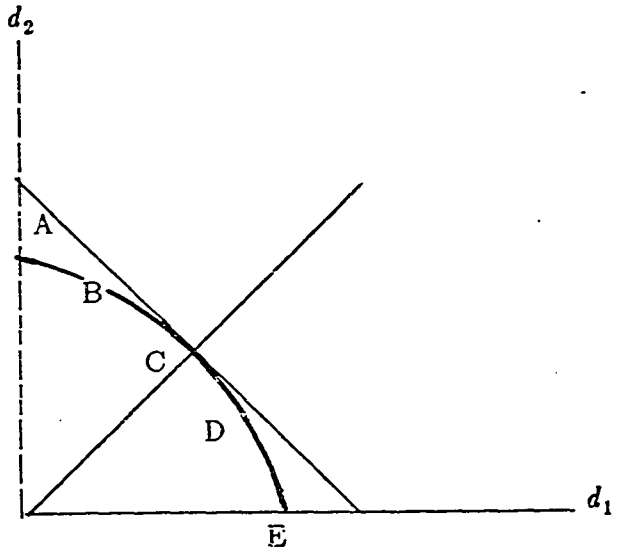


Table B.1: Values for Surface and Plane

Illustrative Information For Figure B.2					
Variable	Value At Point:				
Considered:	A	B	C	D	E
d_1	0.707	0.25	0.5	0.968	0.0
d_2	0.0	0.968	0.5	0.25	0.707
slope $= \frac{\partial d_2}{\partial d_1}$	0.0	-0.258	-1.0	-3.87	$-\infty$

(i)

$$SIM(Q_0,D) = S_0$$

goes through $(S_0, \cdots, S_0)$ where it intersects the hyperplane.

(ii) For any pair $1 \leq j,k \leq n$, it follows from (b) above that

$$\frac{\partial d_k}{\partial d_j} = -\left[\frac{d_j}{d_k}\right]^{p-1}.$$

(iii) At $(S_0, \cdots, S_0)$

$$\frac{\partial d_k}{\partial d_j} = -1$$

for both the hyperplane and the surface. Trivially this is true for the hyperplane, and substituting point D_0 in the equation for the surface yields the same result too.

(iv) $SIM(Q_0, D) = S_0$ is continuous.

(v) For $d_j \leq d_k$,

$$\frac{\partial d_k}{\partial d_j} = -\left[\frac{d_j}{d_k}\right]^{p-1}$$

ranges from 0 at $d_j = 0$ to -1 at $d_j = d_k$. Hence, when other d_i are fixed, as one moves away from $(S_0, \cdots, S_0)$ to smaller d_j values, d_k for the surface is less than d_k for the hyperplane. More formally, let

δd_j = the absolute value of change in d_j

$d_{k,surf} = d_k$ for the surface at $d_j = S_0 - \delta d_j$

$d_{k,plane} = d_k$ for the hyperplane at $d_j = S_0 - \delta d_j$

$$slope_{surf} = \text{average}\frac{\partial d_k}{\partial d_j} \text{ for the surface} \in (-1, 0]$$

$$\text{where } d_j \in [S_0 - \delta d_j, S_0]$$

$$slope_{plane} = \text{average}\frac{\partial d_k}{\partial d_j} \text{ for the hyperplane} = -1$$

so

$$slope_{surf} > slope_{plane}.$$

Then,

$$d_{k,surf} = (S_0 - \delta d_j \cdot slope_{surf})$$
$$< (S_0 - \delta d_j \cdot slope_{plane}) = d_{k,plane}$$

To clarify this argument, consider figure B.3 below which includes a graph of the overall area and an enlargement of the region of interest.

(vi) A similar claim to (v) can be made. For $d_j \geq d_k$, where

$$\frac{\partial d_k}{\partial d_j} = -\left[\frac{d_j}{d_k}\right]^{p-1}$$

ranges from -1 at $d_j = d_k$ to $-\infty$ at $d_k = 0$, then as one moves away from $(S_0, \cdots, S_0)$ to larger d_j values, d_k for the surface becomes less than d_k for the hyperplane.

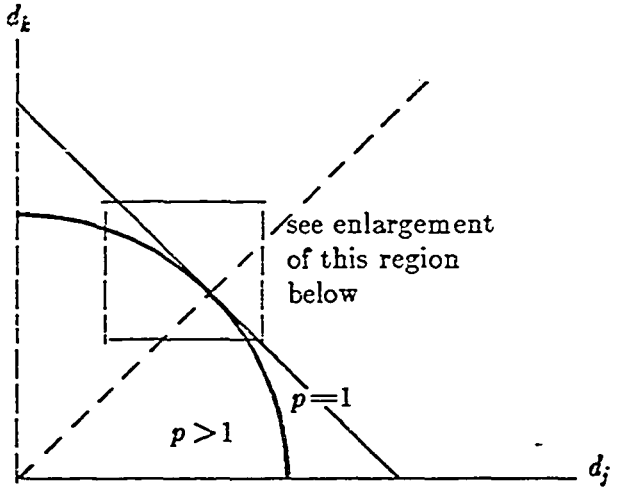
Proof:

$$slope_{surf} < slope_{plane}$$

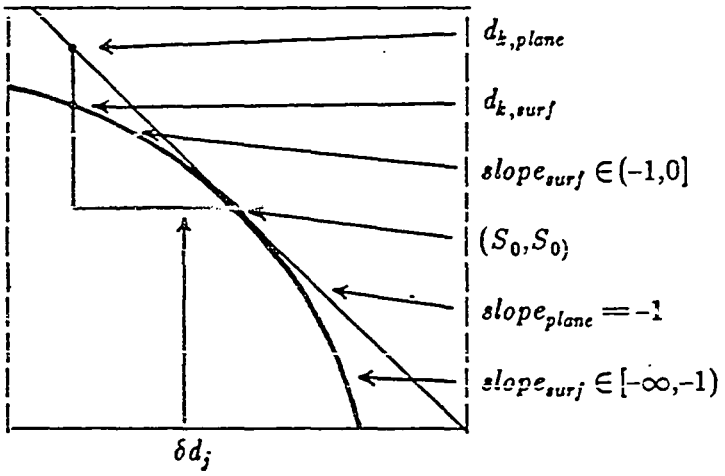
so

Figure B.3: Graph and Enlargement

Graph:



Enlargement:



$$d_{k,surf} = (S_0 + \delta d_j \cdot slope_{surf})$$
$$< (S_0 + \delta d_j \cdot slope_{plane})$$
$$= d_{k,plane}$$

(vii) Now consider some point

$$D_1 = (d_{11}, d_{12}, \cdots, d_{1n}) \neq (S_0, \cdots, S_0) = D_0$$

Since $D_1 \neq D_0$,

$$\exists\, h,k,\ 1 \leq h,k \leq n$$

such that

$$d_{1h} \neq S_0 \neq d_{1k}.$$

One can then apply (v) or (vi) for d_{1h} and d_{1k}. Hence, the point lies closer to $(0,0, \cdots, 0)$ than does the hyperplane. Since the point was arbitrarily chosen on the hyperplane, it may be concluded that the surface $SIM(Q_0, D) = S_0$ lies on the $(0,0, \cdots, 0)$ side of the hyperplane.

Q.E.D.

Lemma 3:

The surface for

S_2: $SIM(Q_{0(p=p_2)}, D) = S_0$

lies closer to the origin than

S_1: $SIM(Q_{0(p=p_1)}, D) = S_0$,

for

$$1 \leq p_1 < p_2 \leq \infty,$$

except in that they touch at

$$D_0 = (S_0, \cdots, S_0).$$

Proof:

(1) Intersection at D_0 has already been discussed.

(2) From Lemma 2 above,

$$\frac{\partial d_k}{\partial d_j} = -\left[\frac{d_j}{d_k}\right]^{p-1}.$$

Now, $p_1 < p_2$ so

$$\frac{\partial d_{k,S_1}}{\partial d_j} = -\left[\frac{d_j}{d_k}\right]^{p_1-1} \text{ and } \frac{\partial d_{k,S_2}}{\partial d_j} = -\left[\frac{d_j}{d_k}\right]^{p_2-1}.$$

(a) When $d_j \leq d_k$, then

$$\left[\frac{d_j}{d_k}\right]^{p_1-1} > \left[\frac{d_j}{d_k}\right]^{p_2-1}$$

and so

$$\frac{\partial d_{k,S_1}}{\partial d_j} < \frac{\partial d_{k,S_2}}{\partial d_j}$$

Hence $slope_{S_2} > slope_{S_1}$, and

$$d_{k,S_2} = (S_0 - \delta d_j \cdot slope_{S_2}) < (S_0 - \delta d_j \cdot slope_{S_1}) = d_{k,S_1}$$

(b) When $d_j \geq d_k$ then

$$\left[\frac{d_j}{d_k}\right]^{p_1-1} < \left[\frac{d_j}{d_k}\right]^{p_2-1}.$$

Once again, using slopes

$$d_{k,S_2} = (S_0 + \delta d_j \cdot slope_{S_2}) < (S_0 + \delta d_j \cdot slope_{S_1}) = d_{k,S_1}$$

(c) Now, the above holds for $i \leq j, k \leq n$, so one can argue as for Lemma 2. The conclusion is that for

$$D_1 = (d_{11}, \cdots, d_{1n}) \neq (S_0, \cdots, S_0) = D_0,$$

where $SIM(Q_{0(p=p_1)}, D_1) = S_0$ so that it lies on S_1, then

$$\exists\, h,k,\ 1 \leq h,k \leq n$$

such that

$$d_{1h} \neq S_0 \neq d_{1k}.$$

Now, apply one of the two cases (a) or (b) above. Since the argument was independent of index, it may be concluded that the surface S_2 lies closer to the origin than S_1 except at the point of intersection, $D_0 = (S_0, \cdots, S_0)$.

With the three lemmas above shown, it is straightforward to complete the proof of the theorem given at the beginning of this section (B.6):

$$SIM(Q_{0(p=p_1)}, D) < SIM(Q_{0(p=p_2)}, D)$$

for

$$1 \leq p_1 \leq p_2 \leq inf,\ D \neq D_0.$$

Choose $D_1 \neq D_0$. Let $s_1 = SIM(Q_{0(p=p_1)}, D_1)$. Then D_1 lies on some surface S_1 such that $SIM(Q_{0(p=p_1)}, D) = s_1$. According to Lemma 3, except at the single point where the two surfaces touch, and where things are equal, the surface S_2 such that $SIM(Q_{0(p=p_2)}, D) = s_1$ lies closer to the origin than does S_1. Hence, D_1 must also lie on some surface S_3, further from the origin than S_2. For $Q_{0(p=p_2)}$, being on a surface further from the origin means that a higher similarity results. Hence

$$SIM(Q_{0(p=p_2)}, D) > s_1 = SIM(Q_{0(p=p_1)}, D)$$

Q.E.D.

APPENDIX C

COLLECTION CHARACTERISTICS

C.1. Overview

Any work proposing new methods and attempting to validate theories experimentally must utilize appropriate test data. Though some have proposed automatically constructing test collections to use in evaluating differing retrieval systems [Tague, Nelson & Wu 1981], it is unlikely that such a scheme will be acceptable to critics for other than efficiency comparisons.

For this thesis, therefore, where effectiveness has been the main criteria of success, it has been necessary to have realistic document collections suitable for each type of experiment. In other words, it was necessary to begin with the text form of document titles and abstracts, an appropriate number of natural language queries, and relevance judgments relating the documents and queries. For p-norm and Boolean feedback comparisons, Boolean queries were also necessary. Finally, for extended vector runs, additional types of information about documents were required, such as author names or co-citation lists.

Five collections have been utilized in this study. The CACM and ISI collections are brand new - devised specially for extended vector experiments. Boolean queries for these two as well as for the other three collections were also created anew, to enable comparisons between vector, Boolean, and p-norm techniques. Relevance judgments were required for the new CACM and ISI collections, and additional ones were also needed for the INSPEC collection.

The CACM and ISI collections are described in detail in [Fox 1983b] so only general information about them will be included here. The other three document collections have been used before, so the focus in subsequent sections will be on summary information about all of the collections and about peculiarities of the new query collections.

C.2. Document Collections

Table C.1 summarizes essential data about each of the document collections. Generally they are made up of the title and abstract of monographs or journal articles. A number of subjects are dealt with, from the "soft" social science like material that makes up a fair proportion of the ISI collection, to the terse, medical articles used in Medlars studies.

Table C.1: Document Collections Summary

Short Name	No. of Docs.	No. of Terms	Av.No. Terms	Subvectors Included	Subject Matter	Years Covered
ADI	82	886	27.1	tm	Documentation	1963
CACM	3,204	10,446	40.1	au,bi,bc, cc,cr, ln,tm	Computer Science	1958 -1979
INSPEC	12,684	14,683	35.4	tm	Electrical Engineering	1979
ISI	1,460	7,392	104.9	au,cc,tm	Information Science	1969 -1977
Medlars	1,033	8,750	55.8	tm	Medicine	to 1969

(1) The small ADI collection, of 82 articles from the proceedings of the 1963 meeting of the American Documentation Institute (later, ASIS – American Society for Information Science), covers early publications about librarianship, microforms, and other technical matters discussed at the time.

(2) The CACM documents include all articles in issues of the *Communications of the ACM* from the first issue in 1958 to the last number of 1979. A considerable range of computer science literature is covered by those 3204 entries in the publication that for many years served as the premier periodical in the field.

(3) The 12,684 INSPEC documents constitute just a small part of the roughly 2 million entries published since 1969. This subset was selected by a group at Syracuse University and used there for some experiments which focused on the effects of searchers and document representations [Katzer et al. 1982]. INSPEC, which stands for Information Services in Physics, Electrotechnology, Computers and Control, covers three *Science Abstracts* publications, *Electrical and Electronics Abstracts*, *Computer and Control Abstracts*, and *Physics Abstracts*. INSPEC is prepared by The Institution of Electrical Engineers, Hertfordshire, England. The documents selected are those covered in part of one release tape: INSPEC-1 c 1979 (even issues 7902-7924). A copy of those documents was made available to the Cornell University Department of Computer Science, with the permission of Jeff Pache of IEE. The content focuses mainly on electrical engineering and computer science subjects. The documents were indexed using SMART procedures, except that to make storage and processing more manageable all terms occurring only once in the collection were omitted from the dictionary and vectors.

(4) The 1460 ISI entries were selected based on co-citation information relating to a study conducted by Dr. Henry Small of the Institute for Scientific Information © (ISI ©). They are the items that could be located at the Cornell University library out of a total of 1627 names listed in the field of information science. Each was published between 1969 and 1977 and received at least five citations.

(5) The 1033 Medlars articles were selected out of a large medical collection available at the National Library of Medicine. These documents were used for many studies since the inception of the SMART project, the most notable being a comparison of Boolean and vector methods that was reported in [Salton 1969].

C.3. Query Collections in General

Since considerable effort was made to study the characteristics and behavior of various query formulations, it is worthwhile to examine the five different query collections and all the versions present for each. Table C.2 gives statistics relating to the queries and the number of relevant documents for each query. The ADI collection is omitted since it was only used for preliminary testing.

To give some idea as to the average length of each query, that value is given for the cosine version based on the original natural language (NL). The ISI value is low, because only the 35 queries for which both vector and Boolean logic (BL) forms are available were considered, and those queries are rather short. The INSPEC queries are much longer, since users were not very experienced and so provided full paragraphs describing their interests, with many unimportant words thrown in.

Query generality, that is the number of relevant documents per query, is illustrated in the next two columns of the table. The first number is given in absolute terms and the

Table C.2: Query and Relevant Document Characteristics

Collection Name	No. of Queries	Av. Length Cos Query	Rels Per Query		Rels in Top 10	
			Av.No.	Av. %	Av.No.	Av. %
CACM	52	11.4	15.3	0.5	1.9	19
INSPEC	77	16.0	33.0	0.3	2.0	9
ISI	35,76	8.1	49.8	3.4	1.7	4
Medlars	30	10.4	23.2	2.3	3.8	18

second is a percentage of the total number of documents. CACM queries have very few relevant documents, both in actual numbers and as a percentage. INSPEC have roughly twice as many, and the values probably are fairly typical. Medlars queries have slightly fewer relevants, but the percentage of documents that are relevant is much higher. And the ISI collection seems to have too many relevant documents per query, both in absolute numbers and as a percentage; those queries were much too vague to give small, sharply defined relevant sets.

The final two columns of Table C.2 illustrate the retrieval behavior of the queries using cosine correlation by considering the top 10 ranked documents. For Medlars, almost 4 of the first 10 are relevant on average, indicating that the queries are likely to have high precision. Further, almost 20% of the relevant documents are retrieved there, showing that recall is probably good also. For CACM, the recall also seems high, but the precision is roughly half. In the other two collections, around 2 relevant documents are found in the

top 10 retrieved, so precision is probably low there as well. And there is even a lower percentage of the number of relevant documents that are identified there. For ISI, in particular, only 4% of the relevant documents are found in the top 10, indicating that recall will be rather low. Since there are so many relevant documents per query, that is very likely to be the case.

Table C.3 gives descriptive details about the five different groups of query collections.

Table C.3: Query Collections Summary

Collection Name	NL Queries No.	NL Queries Description	BL Queries No.	BL Queries Description
ADI	35	Written by 2 Harvard computer science students	35	4 forms by author 1 form each by 2 librarians
ADI-2c	19	Two clause linearized	19	Written by author
ADI-2t	13	Two term linearized	13	Written by author
CACM	52	By Cornell and other computer personnel	52	1 form each by 2 computer science grad. students
INSPEC	77	By students, faculty at Syracuse Univ.	77	1 form by one of 7 Syracuse searchers
ISI	76	From ADI, ISPRA sets and SIGIR Forum abstracts	35	From ADI part only (see ADI coll. above)
Medlars	30	From NLM files	30	From NLM searchers and then expanded using MESH

There is usually one set of natural language queries for each of the five, but in several cases there are several Boolean forms based on each natural language query.

The ADI collection originally had 35 natural language queries. For the initial tests of p-norm techniques, a first searcher (this author) and two Cornell University library employees who conducted searches at the main library, each rephrased the natural language questions into a Boolean representation. In addition, the first searcher also devised 3 other p-norm versions of the Boolean logic queries.

For other ADI tests, 13 two term and 19 two clause Boolean queries were devised by this author, and relevance judgments were also made. Since the Boolean forms were the original questions, the subsequent natural language version was simply the string of terms linearized (flattened) from each Boolean expression.

52 CACM queries were submitted from a variety of sources, and two students in the Cornell graduate level information retrieval course, who were responsible for creating the entire query collection, each proposed Boolean forms. The INSPEC queries were selected from the various ones devised by seven different searchers working at Syracuse University.

For ISI, the same 35 natural language and Boolean queries used for ADI were employed. Multiple concept type testing, however, required more natural language queries, so 41 additional ones were selected from a set of queries for the ISPRA collection and from some new queries based on abstracts published in issues of the *ACM SIGIR Forum*.

Medlars natural language queries were as provided from the National Library of Medicine (NLM) files and Boolean forms were later constructed at Cornell. They were based on Boolean expressions employed by searchers, but adapted to the information in document representations by expanding Medical Subject Headings category names by substituting

terms listed under those in the MESH thesaurus [National Library of Medicine 1968].

Further collection specific details about the various collections are provided in the following subsections dealing with each query collection one by one.

C.4. ADI Queries

C.4.1. ADI Natural Language Queries

35 queries were prepared years ago by two Harvard students. A discussion of the behavior of the queries is given in [Keen 1971]. Table C.4 shows some examples of the queries; 4 of the 35 are included. It can be seen that short phrases are typically utilized.

Table C.4: Examples of ADI Natural Language Queries

(1)	What problems and concerns are there in making up descriptive titles? What difficulties are involved in automatically retrieving articles from approximate titles? What is the usual relevance of the content of articles to their titles?
(9)	What possibilities are there for automatic grammatical and contextual analysis of articles for inclusion in an information retrieval system?
(19)	Techniques of machine matching and machine searching systems. Coding and matching methods.
(35)	Government supported agencies and projects dealing with information dissemination.

C.4.2. Boolean Queries

Table C.5 shows samples from the set of 35 Boolean queries formed by this author based on the corresponding natural language queries described above.

The alternate version of query 1 shown at the bottom of Table C.5 was constructed using the principles of lexical relations. Thus, each relatively low frequency term in the ori-

Table C.5: ADI Prefix Form Boolean Queries by Searcher 1

a- Original Queries

```
(1) AND ( titles,
        OR ( automatically, retrieving, problems, concerns,
             descriptive, approximate, difficulties,
             content, relevance, articles) )

(9) AND ( OR ( information, retrieval, system, automatic,
             possibilities, inclusion, analysis),
        OR ( grammatical, contextual) )

(19) AND ( OR ( matching, searching, coding),
        OR ( methods, machine, techniques, systems) )

(35) AND ( government,
        OR ( information, dissemination, agencies, projects) )
```

b- Expanded Version of Query 1 Above, Adding Lexically Related Words

```
(1) AND ( titles,
        OR ( automatically, retrieving, problems,
             concerns, descriptive,
        OR ( approximate, <OR(compared,matching,comparison,
                              resemble,comparative),.5>),
        OR ( difficulties,<OR(solve,investigation,
                              test,problem),.5>),
        OR ( content,     <OR(contains,substantial,include,
                              ideas,item,subject-matter,
                              subject),.5>),
        OR ( relevance,   <OR(relevancy,relevant,relationship,
                              comparison,applicability),.5>),
        OR ( articles,    <OR(writing,author,journal,book,
                              paragraph,collection,paper,
                              report),.5>)
           ) ).
```

---

ginal form of query 1 was replaced by an entire clause, where the original word, fully weighted, is ORed with a .5 weighted OR clause that includes all lexically related words to

the one being considered. Thus, "approximate" is replaced by the fourth and fifth lines of the new version of query 1, as shown in part (b).

For early p-norm testing, several other specialized query collections were required. In order to identify a proper weighting technique, and to allow graphing of the appropriate portion of the complete vector space, 13 two term queries were randomly chosen so as to represent the various possible combinations of term frequencies. Table C.6 shows these queries as used. Relevance judgments were made for each by this author.

Since two term queries are not very realistic, longer queries were also formulated. Two different ideas were each expressed by a clause, and the resulting two clause query expressed a real interest relevant to the subject matter. Relevance judgments were also

Table C.6: 13 ADI Two Term Queries

No.	Query Terms
1	government agencies
2	form (of) articles
3	printing articles
4	library books
5	scientific definitions
6	distribution (of) journals
7	education (of) personnel
8	journal publication
9	library mechanization
10	recognition (or) printing
11	training programs
12	recognition (or) transformation
13	responses (to) requests

made by this author, so the fairly unusual situation existed of starting with a Boolean expression as the desired query rather than beginning with a natural language statement. Table C.7 gives the 19 two clause queries that were constructed in the above mentioned manner.

C.5. INSPEC Queries

C.5.1. Obtaining Boolean Query Collection

In the Syracuse experimental study, 84 natural language queries were submitted by Syracuse students, faculty, and staff working in Electrical Engineering, Computer Science, and Information Studies departments.

In order to compare the effects of different searchers and different search representations, 7 different searchers familiar with Diatom[1] were chosen, and 7 different representations were selected. Each document had a title, abstract, list of descriptors, etc. so one search was done against the title, another against title and abstract, a third against descriptors, etc. A Latin square design was employed with each searcher using a specific search form on a given query.

The representation searching the free text of the title and abstract fields was chosen as most suitable for helping obtain an appropriate Boolean query to use at Cornell. Even though the selected 84 queries were all of one representation, they came from 7 different searchers. Since the experiments planned were to average the results for the entire set and compare different methods that way, having a number of different searchers participating was probably an advantage rather than a disadvantage.

---

[1]Diatom is a Syracuse system simulating DIALOG (trademark of LMSC, Inc.).

Table C.7: 19 ADI Two Clause Boolean Queries

No.	Two Clause Infix Form
1	(catalogue OR catalog) AND (mechanization OR automation OR computerization)
2	(information AND retrieval) AND (system OR mechanized OR automatic OR machine)
3	(utility OR user OR man OR human) AND (system OR interface OR display)
4	(syntactic OR structural) AND (matching OR identification OR analysis OR transformation)
5	(information AND science) AND (education OR training)
6	searching AND (strategy OR method OR technique)
7	(conventional OR manual) AND (index OR subject)
8	(selective AND dissemination) OR sdi
9	(vocabulary OR descriptors OR concepts) AND indexing
10	(users AND relevant) AND (judged OR regarded)
11	(composition OR photocomposition OR typesetting) AND (computer OR automatic)
12	(thesaurus OR synonyms) OR (term AND relationship)
13	(graduate OR university OR school OR academy) AND (program OR course OR education OR training)
14	(distribution OR copying OR reproduction) AND (journals OR periodicals OR papers OR information)

Table C.7 continued: 19 ADI Two Clause Boolean Queries

No.	Two Clause Infix Form
15	(logic OR calculus) AND (language OR linguistics)
16	(titles OR abstracts) AND (citation OR reference)
17	(microfiche OR microfilm) AND (access OR reproduction)
18	(information OR retrieval) AND (specialist OR personnel OR professional)
19	(statistical OR significant) AND (educational OR results OR effect)

Figure C.1 shows the natural language form of query 101, followed by an example of the search process carried out for that query with Diatom.

To use the Diatom queries at Cornell a number of changes were called for. First, it should be noted that at Syracuse an entire search was carried out, where various sets were retrieved and eventually the results of one of the sets was selected for printing. Since a single query was needed for the Cornell tests the complete expression that yielded the printed set had to be constructed. Figure C.2 shows the prefix notation for the query that resulted from the search given in Figure C.1.

A second difference between the Diatom searches and the Cornell Boolean queries relates to the connectives employed. The Cornell documents were stored as simple vectors, so the standard AND, OR, and NOT operators were easily implemented. However, no positional information was included, and the full text of abstracts and titles were not retained,

Figure C.1: Query 101 Natural Language and Diatom Search

Natural Language Form from User:

101) I am interested in the support of high level and very high level computer languages by means of hardware and firmware (microprogramming). This can include the design of languages such as APL, SETL, or ADA and the architecture of the machines designed to support the high level language (HLL). The topics may include the design of the intermediate language (or machine architecture) or the data structures or other theoretical results useful in the direct execution of HLL.

Original Diatom Search Script:

```
[1]     S COMPUTER? OR PROGRAMMING OR INTERMEDIATE
[2]     S HIGH(F)LEVEL
[3]     S APL OR SETL OR ADA
[4]     S LANGUAGE?
[5]     C (1 OR 2 OR 3) AND 4
[6]     S HARDWARE OR FIRMWARE OR MICROPROGRAMMING
[7]     S (MACHINE OR COMPUTER) AND ARCHITECTURE
[8]     S DATA AND STRUCTURE
[9]     C 6-8/OR
[10]    C 5 AND 9
        T10/6/1-5
[11]    C 2 AND 4
[12]    C 11 AND 9
        T12/6/1-5
[13]    C 3 AND 4
[14]    C 13 AND 9
[15]    C 3 AND 9
[16]    S INTERMEDIATE AND LANGUGE?
[17]    S INTERMEDIATE AND LANGUAGE?
[18]    C 17 AND 9
[19]    C 18 NOT 12
        T19/2/1-3
[20]    C 12 OR 19
[21]    S DESIGN OR SUPPORT
[22]    C 21 OR 10
[23]    C 5 AND 9
[24]    C 10 AND 21
[25]    C 23 NOT 20
[26]    C 23 NOT (12 OR 19)
        T25/6/1-5
[27]    C 20 OR 25
        PRINT 26/1/1-27
```

Figure C.2: Prefix Form of Query 101, Constructed from Diatom Search

```
#and(#and(#and(#or(#or( computer, programming,
                        intermediate ),
                    #or(#and( high, level ),
                    #or( apl, setl, ada ))),
                language ),

          #or (#or( hardware, firmware, microprogramming ),
               #and(#or( machine, computer ), architecture ),
               #and( data, structure ))),

     #not(#or(#and(#and(#and( high  , level ), language ),
                    #or (#or( hardware, firmware,
                              microprogramming ),
                         #and(#or( machine, computer ),
                              architecture ),
                         #and( data, structure ))),

               #and(#and(#and( intermediate, language ),
                         #or (#or( hardware, firmware,
                                   microprogramming ),
                              #and(#or( machine, computer ),
                                   architecture ),
                              #and( data, structure ))),

                    #not(#and(#and(#and( high, level ),
                                   language ),
                              #or (#or( hardware, firmware,
                                        microprogramming ),
                                   #and(#or( machine, computer),
                                        architecture ),
                                   #and(data, structure)))))))));
```

so metrical operators (e.g., within n words, adjacent to, in same sentence or paragraph) were not implementable. All special operators in the Diatom queries were therefore replaced by the less precise AND connector.

A third difference is that Diatom allows word truncation while SMART uses stemming. In general, the full word which best fit in the Diatom expression was therefore substituted so that the SMART indexing could replace it with the appropriate word stem from

the dictionary. A small amount of other editing was also performed at this stage.

Finally, only 77 out of the original 84 queries were utilized. The ones omitted each had some type of problem, such as retrieving no relevant documents, or resulting from searches that were riddled with errors or incomprehensible expressions. But the 77 remaining queries seemed to be a reasonable set to work with.

One final note concerns the relative lengths of the natural language and Boolean queries. In general, the natural language statements were very long, not very specific, and filled with unimportant words. The Boolean queries, however, were fairly specific and reasonably well thought out.

C.5.2. Obtaining Relevance Judgments

The files provided by Syracuse included the relevant documents for each selected query based on users having examined the documents retrieved by each of the 7 searches carried out. A total of 6679 relevance judgments were made, yielding 1760 relevant documents, for an overall precision of 26.4%.

Documents not retrieved by any of the 7 searches were not presented to the users so exhaustive relevance information was not available. For the p-norm experiments and for vector comparison runs as well, it was expected that additional relevant documents might be retrieved via those types of searches.

Consequently, in addition to the set S of retrieved documents based on Syracuse searches, two other sets were formed and relevance judgments made on them. Set C was of 3850 documents retrieved using cosine correlation and vector methods. Set P, 3702 documents, was based on a p-norm search - tf*idf document weights, binary weights elsewhere, and $p = 1$ everywhere, were the parameters chosen. Documents in sets C and P

were among the top 50 retrieved by the appropriate query, with a nonzero similarity required in all cases.

The union of sets C and P was formed, and several people in the SMART group who were familiar with the subject area made relevance judgmnts on the 4991 newly retrieved documents in set $(C \cup P) - S$.

Table C.8 shows counts and percentages relating to the three sets S, C, and P. Half of the chart relates to the retrieved sets, and the other half to the relevant portions of those retrieved sets. All together, 11,670 relevance judgments were made, yielding 2543 relevant documents. Since it was based on 7 searches, the Syracuse search set retrieved over 57% of the total number retrieved, and found almost 70% of the relevant documents. The other two search methods did about equally well. There was more overlap between the Boolean and p-norm results, as would be expected since the p-norm queries were originally based on the Syracuse search statements. In terms of documents retrieved only by a single method, the Syracuse set was largest, followed by the vector method using cosine correlation, and then by p-norm. All in all, it did seem worthwhile to use the three different search strategies to arrive at a closer approximation to having complete relevance information.

All relevance judgments were made on a 4 point scale (1=relevant, 2=probably relevant, 3=probably not relevant, 4=not relevant). Since SMART evaluations call for binary relevance information, it was decided that the above scale be transformed, with 1-2 meaning relevant and 3-4 meaning not relevant.

Table C.8: Counts and Percentages for Retrieved and Relevant Retrieved Documents by Various Search Methods

Key:

s means Syracuse, c means cosine, and p means p-norm.
∪ means union, ∩ means intersection, - means difference.

Retrieved only.			Retrieved and relevant.		
Set	No.	Percent	Set	No.	Percent
s	6679	57.2	s	1760	69.2
c	3850	33.0	c	966	38.0
p	3762	32.2	p	1029	40.5
s ∪ c	9603	82.3	s ∪ c	2245	88.3
s ∪ p	9130	78.2	s ∪ p	2166	85.2
c ∪ p	6768	58.0	c ∪ p	1585	62.3
s ∪ c ∪ p	11670	100.0	s ∪ c ∪ p	2543	100.0
s ∩ c	926	7.9	s ∩ c	481	18.9
s ∩ p	1311	11.2	s ∩ p	623	24.5
c ∩ p	844	7.2	c ∩ p	410	16.1
s ∩ c ∩ p	460	3.9	s ∩ c ∩ p	302	11.9
s ∩ c - p	466	4.0	s ∩ c - p	179	7.0
s ∩ p - c	851	7.3	s ∩ p - c	321	12.6
c ∩ p - s	384	3.3	c ∩ p - s	108	4.2
∪ of all ∩	2161	18.5	∪ of all ∩	910	35.8
only s	4902	42.0	only s	958	37.7
only c	2540	21.8	only c	377	14.8
only p	2067	17.7	only p	298	11.7

C.6. Medlars Queries

C.6.1. Obtaining Boolean Query Collection

Since the Medlars document representations used were made up of titles and abstracts, and the natural language queries used the same vocabulary, vector processing of both was possible immediately.

However, the Boolean queries formulated by the Medlars searchers referred to MESH thesaurus categories as well as free text document terms. Since this extra descriptor-type data about the documents was not available without going to the National Library of Medicine files, and since such data could not then be used for vector processing, it was decided that the Boolean queries should instead be modified to compensate.

Consider the query forms shown in Figure C.3. At the top is the natural language form of query 1 of the Medlars set. Then comes the NLM search formulation, referring to various categories. Finally following it, however is only the beginning portion of the corresponding Boolean formulation.

Several comments are in order about the Boolean version. First, the notation uses "#and" for the AND operator, and "#or" for the OR operator.

Second, the original searcher provided query contained a number of MESH thesaurus categories. Those were replaced by the disjunction of entries listed under each category. For example, query 1 referred to "vertebrates", so subentries such as "human, animals, apes" were substituted. When a phrase like "guinea pigs" was found, it was replaced by the conjunction, as in

#and (guinea, pigs).

Clearly, a single thesaurus entry could result in a large number of replacement terms which is why only a small portion of the first Boolean query is shown.

Some of the subentries out of the MESH thesaurus were words not in the document collection. The query parsing routine looked up each word stem in the term dictionary, and so those were omitted from the query. For example, there was an entry between "birds" and "canaries" that did not appear in the Medlars term dictionary.

Figure C.3: Forms of Medlars Query Number 1

Natural Language:

The crystalline lens in vertebrates, including humans.

Medlars Search Formulation, Referring to Categories in MESH:

[{lens,crystalline} OR crystallins]
AND
[human OR {any vertebrate term} OR
{any animal disease term}
OR {chick embryo}]
AND NOT
[surgery OR {surgery,operative} OR {cataract extraction}
OR {enzymatic zonulolysis} OR {cryogenic surgery} OR
prosthesis OR {drug therapy} OR {drug incompatibility}
OR {drug synergism} OR {drug tolerance} OR
{any invertebrate term}
{OR the subheadings:}
{drug therapy} OR surgery OR {therapeutic use}
OR {administration & dosage}]

Beginning of Expanded Search Formulation:

```
#and ( #or ( #and (lens, crystalline), crystallins),
     #or ( human, amphibia, animals, apes,
          birds, canaries,
          cats, cattle, chickens,
          dogs, fishes, frogs,
          goats, #and (guinea, pigs), hamsters,
          horses, lizards, mammals,
          mice, monkeys, poultry, primates,
          rabbits, rats, rodents, sheep,
          snakes, turkeys,
          vertebrates, abortion,
          #and (animal, disease), avian,
          #and (bird, diseases), bovine,
          #and (cat, diseases),
          #and (cattle, diseases),
          #and (renal, infections),
          #and (dog, diseases), . . .
```

C.6.2. Retrieval Performance

The natural language queries performed fairly well. They were not too short, and used rather precise terminology, so both recall and precision could be expected to be fairly high.

The Boolean queries, however, were a bit more problematic. While the original Medlars Boolean queries searched against the document representations available at NLM gave about the same performance as vector methods [Salton 1969], the new queries were rather different in being much less effective for searching.

Apparently, it is a different process for an indexer to assign a thesaurus category to a document or to choose that category for searching than it is to try to match any subentry implied by that category with a term that actually must appear in the document title or abstract. Perhaps the document would be indexed under a category without any of the subentry terms actually appearing, or the indexer would not assign the category even though some subentry did appear.

More important though, probably, is the fact that the "expanded" queries have too many terms added in OR constructions. Whereas thesaurus categories often have fairly well defined limitations of frequency of occurrence in documents, there is no such control over randomly selected document terms. And when dozens of terms are used as was done for query 1, in an OR construction, the number of documents matching the resulting clause may be rather large. The specificity will most likely be greatly reduced and so precision should be expected to decrease. Ideally one would like to replace a single term by the disjunction of a small number of terms with similar search characteristics - searchonyms [Attar & Fraenkel 1977]. The lexical relation experiments showed that at best one can get

mild improvements for low frequency terms when synonyms or other lexically related terms are included. However, expansion of thesaurus categories causes addition of terms with widely different frequency characteristics, in relationships lacking the collection specific correlation of searchonyms or the specific linguistic connections identified by lexical functions.

BIBLIOGRAPHY

[Alberga 1967]
Alberga, C. N. String Similarity and Misspellings. *Communications of the ACM*, Vol. 10, No. 6, June 1967.

[Allen & Cady 1982]
Allen, David M. and Foster B. Cady. *Analyzing Experimental Data by Regression.* Lifetime Learning Publications, Belmont, California, 1982.

[Anderson & McLean 1974]
Anderson, Virgil L. and Robert A. McLean. *Design of Experiments: A Realistic Approach.* Marcel Dekker, New York, 1974.

[Apresyan, Mel'chuk & Zholkovsky 1969]
Apresyan, Yu. D., I. A. Mel'chuk and A. K. Zholkovsky. Semantics and Lexicography: Towards a New Type of Unilingual Dictionary. In *Studies in Syntax and Semantics*, edited by F. Kiefer, D. Reidel, Dordrecht, Holland, 1969, pages 1-33.

[Arms & Arms 1978]
Arms, W. Y. and C. R. Arms. Cluster Analysis Used on Social Science Journal Citations. *Journal of Documentation*, Vol. 34, No. 1, March 1978, pages 1-11.

[Artandi 1971]
Artandi, Susan. Document Retrieval and the Concept of Sets. *Journal of the American Society for Information Science*, Vol. 22, No. 4, July-August 1971.

[Astrahan et al. 1976]
Astrahan, M. M. et al. System R: Relational Approach to Database Management. *ACM Transactions on Database Systems*, Vol. 1, No. 2, June 1976.

[Attar & Fraenkel 1977]
Attar, Rony and Aviezri S. Fraenkel. Local Feedback in Full-Text Retrieval Systems. *Journal of the ACM*, Vol. 24, No. 3, July 1977, pages 397-417.

[Attar & Fraenkel 1981]
Attar, Rony and Aviezri S. Fraenkel. Experiments in Local Metrical Feedback in Full-Text Retrieval Systems. *Information Processing and Management*, Vol. 17, No. 3, 1981, pages 115-126.

[Barker, Veal & Wyatt 1972]
Barker, F. H., D. C. Veal and B. K. Wyatt. Toward Automatic Profile Construction. *Journal of Documentation*, Vol. 28, No. 1, March 1972.

[Bartle 1966]
Bartle, R. G. *The Elements of Integration*. John Wiley and Sons, New York, 1966.

[Bayer & McCreight 1972]
Bayer, R. and E. McCreight. Organization and Maintenance of Large Ordered Indexes. *Acta Informatica*, Vol. 1, No. 3, 1972, pages 173-189.

[Becker & Chambers 1981]
Becker, Richard A. and John M. Chambers. *S: A Language and System for Data Analysis*. Bell Laboratories, 1981.

[Bichteler & Parsons 1974]
Bichteler, Julie and Ronald G. Parsons. Document Retrieval by Means of an Automatic Classification Algorithm for Citations. *Information Storage and Retrieval*, Vol. 10, No. 7-8, July/August 1974, pages 267-278.

[Bichteler & Eaton 1977]
Bichteler, J. and E. A. Eaton III. Comparing Two Algorithms for Document Retrieval Using Citation Links. *Journal of the American Society for Information Science*, Vol. 28, July 1977, pages 192-195.

[Bichteler & Eaton 1980]
Bichteler, Julie and Edward A. Eaton III. The Combined Use of Bibliographic Coupling and Cocitation for Document Retrieval. *Journal of the American Society for Information Science*, Vol. 31, No. 4, July 1980.

[Bobrow & Collins, eds. 1975]
Bobrow, D. G. and A. Collins, editors. *Representation and Understanding*. Academic Press, New York, 1975.

[Boehm, Demers & Donahue 1980]
Boehm, Hans, A. Demers and J. Donahue. An Informal Description of Russell. Technical Report 80-430, Cornell University Department of Computer Science, Ithaca, New York, 1980.

[Bookstein 1978]
Bookstein, Abraham. On the Perils of Merging Boolean and Weighted Retrieval Systems. *Journal of the American Society for Information Science*, Vol. 29, No. 3, May 1978, pages 156-158.

[Bookstein 1980]
Bookstein, Abraham. Fuzzy Requests: An Approach to Weighted Boolean Searches. *Journal of the American Society for Information Science*, Vol. 31, No. 4, July 1980, pages 240-247.

[Bookstein 1981]
Bookstein, Abraham. A Comparison of Two Systems of Weighted Boolean Retrieval. *Journal of the American Society for Information Science*, Vol. 32, No. 4, July 1981, pages 275-279.

[Bolc, ed. 1978]
Bolc, L., editor. *Natural Language Communication via Computers*. Lecture Notes in Computer Science, Springer Verlag, Berlin, 1978.

[Borko & Bernier 1978]
Borko, Harold and Charles L. Bernier. *Indexing Concepts and Methods*. Academic Press, New York, 1978.

[Boyer & Moore 1977]
Boyer, R. S. and J. S. Moore. A Fast String Searching Algorithm. *Communications of the ACM*, Vol. 20, No. 10, October 1977, pages 762-772.

[Brauen 1971]
Brauen, T. Document Vector Modification. In *The SMART Retrieval System, Experiments in Automatic Document Processing*, edited by Gerard Salton, Prentice Hall, Inc., Englewood Cliffs, N.J., 1971.

[Brittain & Line 1973]
Brittain, J. Michael and Maurice B. Line. Sources of Citations and References for Analysis Purposes: A Comparative Assessment. *Journal of Documentation*, Vol. 29, No. 1, March 1973.

[Buell & Kraft 1981]
Buell, Duncan A. and Donald H. Kraft. Threshold Values and Boolean Retrieval Systems. *Information Processing and Management*, Vol. 17, No. 3, 1981, pages 127-136.

[Cane 1964]
Cane, Mark. Adapting SMART to the M.I.T. Compatible Time-Sharing System. Information Storage and Retrieval, Vol. 8, Cornell University Department of Computer Science, Ithaca, New York, 1964.

[Carpenter & Narin 1973]
Carpenter, Mark P. and Francis Narin. Clustering of Scientific Journals. *Journal of the American Society for Information Science*, Vol. 24, No. 6, Nov.-Dec. 1973.

[Cawkell 1976]
Cawkell, A. E. Citations, Obsolescence, Enduring Articles, and Multiple Authorships. *Journal of Documentation*, Vol. 32, No. 1, March 1976, pages 53-58.

[Chamberlin et al. 1976]
Chamberlin, Donald D. et al. SEQUEL 2: A Unified Approach to Data Definition, Manipulation, and Control. *IBM Journal Research and Development*, November 1976.

[Chamberlin 1976]
Chamberlin, Donald D. Relational Data-Base Management Systems. *ACM Computing Surveys*, Vol. 8, No. 1, 1976, pages 43-66.

[Chamberlin et al. 1981]
Chamberlin, Donald D. et al. Support for Repetitive Transactions and Ad Hoc Queries in System R. *ACM Transactions on Database Systems*, Vol. 6, No. 1, 1981, pages 70-94.

[Chambers 1977]
Chambers, John M. *Computational methods for data analysis.* John Wiley & Sons, New York, 1977.

[Chang, Cirillo & Razon 1971]
Chang, Y. K., C. Cirillo and J. Razon. Evaluation of Feedback Retrieval Using Modified Freezing, Residual Collections, and Test and Control Groups. In *The SMART Retrieval System, Experiments in Automatic Document Processing*, edited by Gerard Salton, Prentice Hall, Inc., Englewood Cliffs, N.J., 1971.

[Cleverdon & Keen 1968]
Cleverdon, C. W. and E. M. Keen. *Factors Determining the Performance of Indexing Systems.* Aslib Cranfield Research Project, volume 1 and 2, Cranfield, England, 1968.

[Codd 1970]
Codd, E. F. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, Vol. 13, No. 6, 1970, pages 377-387.

[Codd et al. 1978]
Codd, E. F. et al. Rendezvous: Version 1. Report RJ 2144, IBM Research Laboratory, San Jose, California, January 1978.

[Codd 1979]
Codd, E. F. Extending the Database Relational Model to Capture More Meaning. *ACM Transactions on Database Systems*, Vol. 4, No. 4, 1979, pages 397-434.

[Crawford 1981]
Crawford, Robert G. The Relational Model in Information Retrieval. *Journal of the American Society for Information Science*, Vol. 32, No. 1, 1981, pages 51-64.

[Croft 1977]
Croft, W. Bruce. Clustering Large Files of Documents Using the Single-link Method. *Journal of the American Society for Information Science*, Vol. 28, No. 6, November 1977, pages 341-344.

[Croft 1978a]
Croft, W. Bruce. *Organizing and Searching Large Files of Document Descriptions*. Cambridge University Ph.D. Thesis, Cambridge, England, 1978.

[Croft 1978b]
Croft, W. Bruce. A File Organization for Cluster-based Retrieval. In First International Conference on Information Storage and Retrieval. *ACM SIGIR Forum*, Vol. 13, No. 1, Summer 1978, pages 65-82.

[Croft 1979]
Croft, W. Bruce. Using Boolean Queries with a Clustered File Organization. *Journal of the American Society for Information Science*, Vol. 30, No. 6, November 1979, pages 358-360.

[Croft & Harper 1979]
Croft, W. Bruce and D. Y. Harper. Using Probabalistic Models of Document Retrieval without Relevance Information. *Journal of Documentation*, Vol. 35, No. 4, December 1979, pages 285-295.

[Croft 1980]
Croft, W. Bruce. A Model of Cluster Searching Based on Classification. *Information Systems*, Vol. 5, No. 3, 1980, pages 189-195.

[Croft 1982]
Croft, W. Bruce. An Overview of Information Systems. *Information Technology: Research and Development*, Vol. 1, No. 1, January 1982.

[Cronin 1981]
Cronin, Blaise. The Need for a Theory of Citing. *Journal of Documentation*, Vol. 37, No. 1, March 1981, pages 16-24.

[Cullingford 1978]
Cullingford, Richard Edward. Script Application: Computer Understanding of Newspaper Stories. Research Report 116, Yale University Department of Computer Science, New Haven, Connecticut, January 1978.

[Cummings & Fox 1973]
Cummings, L. J. and D. A. Fox. Some Mathematical Properties of Cycling Strategies Using Citation Indexes. *Information Storage and Retrieval*, Vol. 9, No. 12, December 1973, pages 713-719.

[Dahl, Dijkstra & Hoare 1972]
Dahl, O. J., E. W. Dijkstra and C. A. R. Hoare. *Structured Programming*. Academic Press, New York, 1972.

[Date 1982]
Date, C. J. *An Introduction to Database Systems. Volume I.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1982.

[Date 1983]
Date, C. J. *An Introduction to Database Systems. Volume II.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1983.

[Dattola 1971]
Dattola, Robert T. Experiments with a Fast Algorithm for Automatic Classification. In *The SMART Retrieval System, Experiments in Automatic Document Processing*, edited by Gerard Salton, Prentice Hall, Inc., Englewood Cliffs, N.J., 1971.

[Dattola 1979]
Dattola, Robert T. FIRST: Flexible Information Retrieval System for Text. *Journal of the American Society for Information Science*, Vol. 30, No. 1, 1979, pages 9-14.

[Demers & Donahue 1983]
Demers, Alan and James Donahue. Making Variables Abstract: An Equational Theory for Russell. Technical Report 82-534, Cornell University Department of Computer Science, Ithaca, New York, 1983.

[Denning et al. 1981]
Denning, Dorothy E. et al. The Proposed New Computing Reviews Classification Scheme: A Report of the Computing Reviews Category Revision Committee. *Communications of the ACM*, Vol. 24, No. 7, July 1981, pages 419-433.

[Dillon & Desper 1980]
Dillon, Martin and James Desper. The Use of Automatic Relevance Feedback in Boolean Retrieval Systems. *Journal of Documentation*, Vol. 36, No. 3, September 1980, pages 197-208.

[Duda & Hart 1973]
Duda, R. O. and Peter E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.

[Ehrich 1982]
Ehrich, R. W. DMS: A System for Defining and Managing Human-Computer Dialogues. In Proceedings of Analysis, Design, and Evaluation of Man-Machine Systems IFAC/IFIP/IFORS/IEA Conference, Federal Republic of Germany, 1982, pages 367-374.

[Evens & Smith 1979]
Evens, Martha W. and Radul N. Smith. A Lexicon for a Computer Question-Answering System. *American Journal of Computational Linguistics*, Microfiche 83 1979.

[Feldman & Sutherland, eds. 1979]
Feldman, Jerome A. and William R. Sutherland, editors. Rejuvenating Experimental Computer Science: A Report to the National Science Foundation and Others. *Communications of the ACM*, Vol. 22, No. 9, September 1979, pages 497-502.

[Findler, ed. 1979]
Findler, N. V., editor. *Associative Networks: Representation and Use of Knowledge by Computers.* Academic Press, New York, 1979.

[Fisher & Siegel 1972]
Fisher, Robert A. and Morris M. Siegel. A Method for Incorporating Boolean Relationships into Queries in the SMART System. Information Storage and Retrieval, Vol. 21, Cornell University Department of Computer Science, Ithaca, New York, 1972.

[Fox 1980]
Fox, Edward A. Lexical Relations: Enhancing Effectiveness of Information Retrieval Systems. *ACM SIGIR Forum*, Vol. 15, No. 3, Winter 1980, pages 5-36.

[Fox 1981]
Fox, Edward A. Implementing SMART for Minicomputers Via Relational Processing with Abstract Data Types. In Joint Proceedings of SIGSMALL Symposium on Small Systems and SIGMOD Workshop on Small Data Base Systems. *ACM SIGSMALL Newsletter*, Vol. 7, No. 2, October 1981.

[Fox 1983a]
Fox, Edward A. Some Considerations for Implementing the SMART Information Retrieval System Under UNIX. Technical Report 83-560, Cornell University Department of Computer Science, Ithaca, New York, July 1983.

[Fox 1983b]
Fox, Edward A. Characterization of Two New Experimental Collections in Computer and Information Science Containing Textual and Bibliographic Concepts. Technical Report 83-561, Cornell University Department of Computer Science, Ithaca, New York, July 1983.

[Garfield 1964]
Garfield, Eugene. Science Citation Index: A New Dimension in Indexing. *Science*, Vol. 144, No. 3619, May 1964, pages 649-654.

[Garfield 1970]
Garfield, Eugene. Citation Measures as an Objective Estimate of Creativity. *Current Contents*, No. 34, August 1970, pages 4-5.

[Garfield 1977]
Garfield, Eugene. Bibliographies, Citations, and Citation Abstracts. In *Essays of an Information Scientist, Vol. 2*, ISI Press, Philadelphia, 1977.

[Garfield 1979]
Garfield, Eugene. *Citation Indexing: Its Theory and Application in Science, Technology, and Humanities.* John Wiley and Sons, New York, 1979.

[Geller, Cani & Davies 1981]
Geller, Nancy L., John S. de Cani and Robert E. Davies. Lifetime-Citation Rates: A Mathematical Model to Compare Scientists' Work. *Journal of the American Society for Information Science*, Vol. 32, No. 1, January 1981, pages 3-15.

[Geschke,Jr. & Satterwaite 1977]
Geschke, Charles M., James H. Morris Jr. and Edwin H. Satterwaite. Early Experience with Mesa. *Communications of the ACM*, Vol. 20, No. 8, 1977, pages 540-553.

[Grauer & Messier 1971]
Grauer, R. T. and M. Messier. An Evaluation of Rocchio's Clustering Algorithm. In *The SMART Retrieval System, Experiments in Automatic Document Processing*, edited by Gerard Salton, Prentice Hall, Inc., Englewood Cliffs, N.J., 1971.

[Gries & Gehani 1977]
Gries, David and Narain Gehani. Some Ideas on Data Types in High-Level Languages. *Communications of the ACM*, Vol. 20, No. 6, 1977, pages 414-420.

[Grimes 1975]
Grimes, Joseph E. *The Thread of Discourse.* Janua Linguarum, series minor 207, Mouton, The Hague, 1975.

[Grimes 1980]
Grimes, Joseph E. Reference Spaces in Text. In Nobel Symposium on Text Processing. Edited by Sture Allén, Språkdata, Gothenburg, 1980.

[Guttag 1977]
Guttag, John. Abstract Data Types and the Development of Data Structures. *Communications of the ACM*, Vol. 20, No. 6, 1977, pages 396-404.

[Hall & Dowling 1980]
Hall, Patrick A.V. and Geoff R. Dowling. Approximate String Matching. *ACM Computing Surveys*, Vol. 12, No. 4, 1980, pages 381-402.

[Harper & Van Rijsbergen 1978]
Harper, D. J. and C. J. Van Rijsbergen. An Evaluation of Feedback in Document Retrieval Using Co-Occurrence Data. *Journal of Documentation*, Vol. 34, No. 3, September 1978, pages 189-216.

[Harris 1977]
Harris, L. R. User Oriented Database Query with the ROBOT Natural Language Query System. *International Journal of Man-Machine Studies*, Vol. 9, 1977, pages 697-713.

[Hartigan 1975]
Hartigan, J. A. *Clustering Algorithms*. John Wiley and Sons, New York, 1975.

[Haskin 1980]
Haskin, R. Hardware for Searching Very Large Text Databases. In Proceedings of Fifth Workshop on Computer Architecture for Non-Numeric Processing, Association for Computing Machinery, New York, March 1980, pages 49-56.

[Haskin & Lorie 1981]
Haskin, Roger L. and Raymond A. Lorie. On Extending the Functions of a Relational Database System. Research Report RJ3182 (38988), IBM Research Laboratory, San Jose, CA, 1981.

[Heaps 1978]
Heaps, H. S. *Information Retrieval: Computational and Theoretical Aspects*. Academic Press, New York, 1978.

[Hendrix et al. 1978]
Hendrix, G. G. et al. Developing a Natural Language Interface to Complex Data. *ACM Transactions on Database Systems*, Vol. 3, No. 2, June 1978, pages 105-147.

[Herr[illegible] 1978]
Herrmach[illegible], Gertrud. Can Retrieval of Information from Citation Indexes be Simplified? Multiple Mention of a Reference as a Characteristic of the Link between Cited and Citing Article. *Journal of the American Society for Information Science*, Vol. [illegible], November 1978.

[IBM Corporation 1979]
IBM Corporation. *Storage and Information Retrieval System for OS/VS (STAIRS/VS): Program Reference Manual*. Publication SH12-5400, IBM Corporation, White Plains, New York, 1979.

[Ide 1971]
Ide, E. New Experiments in Relevance Feedback. In *The SMART Retrieval System, Experiments in Automatic Document Processing*, edited by Gerard Salton, Prentice Hall, Inc., Englewood Cliffs, N.J., 1971.

[Ide & Salton 1971]
Ide, E. and Gerard Salton. Interactive Search Strategies and Dynamic File Organizations. In *The SMART Retrieval System, Experiments in Automatic Document Processing*, edited by Gerard Salton, Prentice Hall, Inc., Englewood Cliffs, N.J., 1971.

[Ivie 1977]
Ivie, Evan L. The Programmer's Workbench - A Machine for Software Development. *Communications of the ACM*, Vol. 20, No. 10, 1977, pages 746-753.

[Jardine & Sibson 1968]
Jardine, N. and R. Sibson. A Model for Taxonomy. *Mathematical Biosciences*, Vol. 2, No. 2-3, May 1968, pages 465-482.

[Jardine & Van Rijsbergen 1971]
Jardine, N. and C. J. Van Rijsbergen. The Use of Hierarchic Clustering in Information Retrieval. *Information Storage and Retrieval*, Vol. 7, No. 5, December 1971, pages 217-240.

[Katzer et al. 1982]
Katzer, Jeffrey et al. *A Study of the Overlap Among Document Representations.* Syracuse University School of Information Studies, 1982.

[Kaufman 1977]
Kaufman, A. Progress in Modelling of Human Reasoning. In *Fuzzy Automata and Decision Processes*, edited by Madan M. Gupta, George N. Saridis and Brian R. Gaines, North-Holland, New York, 1977.

[Kay & McDaniel 1978]
Kay, Paul and Chad K. McDaniel. The Linguistic Significance of the Meaning of Basic Color Terms. *Language*, Vol. 54, No. 3, 1978.

[Keen 1971]
Keen, E. M. An Analysis of the Documentation Requests. In *The SMART Retrieval System, Experiments in Automatic Document Processing*, edited by Gerard Salton, Prentice Hall, Inc., Englewood Cliffs, N.J., 1971.

[Kempson 1977]
Kempson, Ruth M. *Semantic Theory.* Cambridge University Press, Cambridge, 1977.

[Kent 1979]
Kent, William. Limitations of Record-Based Information Models. *ACM Transactions on Database Systems*, Vol. 4, No. 1, 1979, pages 107-131.

[Kernighan & Ritchie 1978]
Kernighan, Brian W. and Dennis M. Ritchie. *The C Programming Language*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1978.

[Kessler 1962]
Kessler, M. M. An Experimental Study of Bibliographic Coupling Between Technical Papers. M.I.T. Report R-1, June 1962.

[Kessler 1963a]
Kessler, M. M. Bibliographic Coupling Between Scientific Papers. *American Documentation*, Vol. 14, No. 1, January 1963, pages 10-25.

[Kessler 1963b]
Kessler, M. M. Bibliographic Coupling Extended in Time: Ten Case Histories. *Information Storage and Retrieval*, Vol. 1, 1963, pages 169-187.

[Kessler, Ivie & Mathews 1964]
Kessler, M. M., E. L. Ivie and W. D. Mathews. The M.I.T. Technical Information Project: A Prototype System. In Proceedings American Documentation Institute, 1964.

[Kessler 1965a]
Kessler, M. M. The MIT Technical Information Project. *Physics Today*, March 1965.

[Kessler 1965b]
Kessler, M. M. Comparison of Results of Bibliographic Coupling and Analytic Subject Indexing. *American Documentation*, Vol. 16, No. 3, July 1965, pages 223-233.

[Koll 1979]
Koll, Matthew B. *The Concept Space in Information Retrieval Systems as a Model of Human Concept Relations*. Syracuse University Ph.D. Dissertation, 1979.

[Kwok 1975]
Kwok, K. L. The Use of Title and Cited Titles as Document Representation for Aautomatic Classification. *Information Processing and Management*, Vol. 11, No. 8-12, 1975, pages 201-206.

[Lacroix & Pirotte 1980]
Lacroix, Michel and Alain Pirotte. Associating Types with Domains of Relational Data Bases. In Proceedings of Workshop on Data Abstraction, Databases and Conceptual Modelling. Edited by Michael L. Brodie and Stephen N. Zilles, Association for Computing Machinery, Inc., 1980.

[Lancaster & Fayen 1973]
Lancaster, F. W. and E. G. Fayen. *Information Retrieval On-Line.* Melville Publishing Co., Los Angeles, CA, 1973.

[Lancaster 1979]
Lancaster, F. W. *Information Retrieval Systems: Characteristics, Testing and Evaluation 2nd Edition.* John Wiley and Sons, New York, 1979.

[Landauer 1981]
Landauer, Christopher. Message Extraction Through Estimation of Relevance. In *Information Retrieval Research,* edited by R. N. Oddy, S. E. Robertson, C. J. van Rijsbergen and P. W. Williams, Butterworths, London, 1981, pages 117-138.

[Lawani 1970]
Lawani, S. M. *The Aslib-Cranfield Studies on the Evaluation of Indexing Systems.* University of Ibadan Institute of Librarianship, Ibadan, Nigeria, 1970.

[Lawani 1981]
Lawani, S. M. Bibliometrics: Its Theoretical Foundations, Methods and Applications. *Libri,* Vol. 31, No. 4, 1981, pages 294-315.

[Lawani 1982]
Lawani, Stephen M. On the Heterogeneity and Classification of Author Self-Citations. *Journal of the American Society for Information Science,* September 1982.

[Lawani & Bayer 1983]
Lawani, Stephen M. and Alan E. Bayer. Validity of Citation Criteria for Assessing the Influence of Scientific Publications: New Evidence with Peer Assessment. *Journal of the American Society for Information Science,* Vol. 34, No. 1, 1983, pages 59-66.

[Ledgard & Taylor 1977]
Ledgard, Henry F. and Robert W. Taylor. Two Views of Data Abstraction. *Communications of the ACM,* Vol. 20, No. 6, 1977, pages 382-384.

[Lesk 1966]
Lesk, M. E. Design Considerations for Time Shared Automatic Documentation Centers. Information Storage and Retrieval, Vol. 11, Cornell University Department of Computer Science, Ithaca, New York, 1966.

[Lesk 1968]
Lesk, M. E. Design of a Revised On-Line Information Retrieval System. Information Storage and Retrieval, Vol. 14, Cornell University Department of Computer Science, Ithaca, New York, 1968.

[Liskov et al. 1977]
Liskov, Barbara et al. Abstraction Mechanisms in CLU. *Communications of the ACM*, Vol. 20, No. 8, 1977, pages 564-576.

[Liu 1976]
Liu, Jane W. S. Algorithms for Parsing Search Queries in Systems with Inverted File Organization. *ACM Transactions on Database Systems*, Vol. 1, No. 4, December 1976, pages 299-316.

[Lockemann et al. 1979]
Lockemann, Peter C. et al. Data Abstractions for Database Systems. *ACM Transactions on Database Systems*, Vol. 4, No. 1, 1979, pages 60-75.

[Lovins 1968]
Lovins, B. J. Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics*, Vol. 11, No. 1-2, March-June 1968, pages 11-31.

[MacCafferty & Gray, eds. 1979]
MacCafferty, M. and K. Gray, editors. *The Analysis of Meaning.* Aslib, London, 1979.

[Macleod 1979]
Macleod, I. A. SEQUEL as a Language for Document Retrieval. *Journal of the American Society for Information Science*, Vol. 30, No. 5, 1979, pages 243-249.

[Martin 1978]
Martin, W. A. Some Comments on EQS: A Near Term Natural Language Data Base Query System. In Proceedings ACM 1978 Annual Conference, December 1978, pages 156-164.

[McCawley 1980]
McCawley, James D. *Everything That Linguists Have Always Wanted to Know About Logic: but Were Ashamed to Ask.* Univ. of Chicago Press, Chicago, 1980.

[McGill et al. 1976]
McGill, Michael J. et al. Syracuse Information Retrieval Experiment (SIRE): Design of an On-Line Bibliographic Retrieval System. *ACM SIGIR Forum*, Vol. 10, No. 4, Spring 1976, pages 37-44.

[McGill & Noreault 1977]
McGill, Michael J. and Terry Noreault. Syracuse Information Retrieval Experiment (SIRE): Rationale and Basic System Design. Report, Syracuse University School of Information Studies, Syracuse, NY, May 1977.

[McGill, Koll & Noreault 1979]
McGill, Michael J., Matthew Koll and Terry Noreault. *An Evaluation of Factors Affecting Document Ranking By Information Retrieval Systems.* Syracuse University School of Information Studies, 1979.

[Meadow & Cochrane 1981]
Meadow, Charles T. and Pauline A. Cochrane. *Basics of Online Searching.* John Wiley and Sons, New York, 1981.

[Mel'chuk 1973]
Mel'chuk, I. A. Towards a Linguistic 'Meaning <=> Text' Model. In *Trends in Soviet Theoretical Linguistics*, edited by F. Kiefer, D. Reidel, Dordrecht, Holland, 1973, pages 33-57.

[Michaels, Mittman & Carlson 1976]
Michaels, Ann S., Benjamin Mittman and C. Robert Carlson. A Comparison of the Relational and CODASYL Approaches to Data-Base Management. *ACM Computing Surveys*, Vol. 8, No. 1, 1976, pages 125-151.

[Michelson et al. 1971]
Michelson, D. et al. An Experiment in the Use of Bibliographic Data As a Source of Relevance Feedback in Information Retrieval. In *The SMART Retrieval System, Experiments in Automatic Document Processing*, edited by Gerard Salton, Prentice Hall, Inc., Englewood Cliffs, N.J., 1971.

[National Library of Medicine 1968]
National Library of Medicine. Medical Subject Headings 1968. *Index Medicus*, Vol. 9, No. 1, Part 2, Washington, D.C., January 1968.

[Nodtvedt 1980]
Nodtvedt, Einar. Information Retrieval in the Business Environment. Technical Report 80-447, Cornell University Department of Computer Science, Ithaca, New York, 1980.

[Noreault, Koll & McGill 1977]
Noreault, Terry, Matthew Koll and Michael J. McGill. Automatic Ranked Output from Boolean Searches in SIRE. *Journal of the American Society for Information Science*, Vol. 28, No. 6, November 1977, pages 333-339.

[O'Connor 1973]
O'Connor, John. Text Searching Retrieval of Answer-Sentences and Other Answer-Passages. *Journal of the American Society for Information Science*, Vol. 24, No. 6, 1973, pages 445-460.

[O'Connor 1980a]
O'Connor, John. Citing Statements: Recognition by Computer and Use to Improve Retrieval. In Proceedings of the 43rd ASIS Annual Meeting, Vol. 17, 1980.

[O'Connor 1980b]
O'Connor, John. Answer-Passage Retrieval by Text Searching. *Journal of the American Society for Information Science*, Vol. 31, No. 4, 1980, pages 227-239.

[O'Connor 1982]
O'Connor, John. Citing Statements: Computer Recognition and Use to Improve Retrieval. *Information Processing and Management*, Vol. 18, No. 3, July 1982, pages 125-131.

[Ortega 1972]
Ortega, James M. *Numerical Analysis: A Second Course*. Academic Press, New York, 1972.

[Oddy 1977]
Oddy, R. N. Information Retrieval Through Man-Machine Dialogue. *Journal of Documentation*, Vol. 33, No. 1, March 1977, pages 1-14.

[Paice 1980]
Paice, C. D. The Automatic Generation of Literature Abstracts: An approach based on the identification of self-indicating phrases. In *Information Retrieval Research*, edited by R. N. Oddy, S. E. Robertson, C. J. van Rijsbergen and P. W. Williams, Butterworths, London, 1980, pages 172-191.

[Paolini 1980]
Paolini, Paolo. Abstract Data Types and Data Bases. In Proceedings of Workshop on Data Abstraction, Databases and Conceptual Modelling. Edited by Michael L. Brodie and Stephen N. Zilles, Association for Computing Machinery, Inc., 1980.

[Peterson 1980]
Peterson, James L. Computer Programs for Detecting and Correcting Spelling Errors. *Communications of the ACM*, Vol. 23, No. 12, December 1980, pages 676-687.

[Plath 1976]
Plath, W. J. REQUEST: A Natural Language Question-Answering System. *IBM Journal of Research and Development*, Vol. 20, No. 4, July 1976, pages 326-335.

[Porter 1982]
Porter, M. F. Implementing a Probabalistic Information Retrieval System. *Information Technology: Research and Development*, Vol. 1, No. 2, April 1982.

[Preece 1974]
Preece, S. E. Clustering as an Output Option. In Proceeding ASIS National Conference, Vol. 10, 1974, pages 189-190.

[Radecki 1977]
Radecki, Tadeusz. Mathematical Model of Time-Efficient Information Retrieval System Based on the Theory of Fuzzy Sets. *Information Processing and Management*, Vol. 13, 1977, pages 109-116.

[Radecki 1979]
Radecki, Tadeusz. Fuzzy Set Theoretical Approach to Document Retrieval. *Information Processing and Management*, Vol. 15, 1979, pages 247-259.

[Radecki 1982]
Radecki, Tadeusz. Reducing the Perils of Merging Boolean and Weighted Retrieval Systems. *Journal of Documentation*, Vol. 38, No. 3, September 1982.

[Ritchie & Thompson 1974]
Ritchie, D. M. and K. Thompson. The UNIX Time Sharing System. *Communications of the ACM*, Vol. 17, No. 7, 1974, pages 365-375.

[Robertson & Sparck Jones 1976]
Robertson, S. E. and Karen Sparck Jones. Relevance Weighting of Search Terms. *Journal of the American Society for Information Science*, Vol. 27, No. 3, 1976, pages 129-146.

[Robertson 1978]
Robertson, Stephen E. On the Nature of Fuzz: A Diatribe. *Journal of the American Society for Information Science*, November 1978.

[Robertson, Van Rijsbergen & Porter 1980]
Robertson, Stephen E., C. J. Van Rijsbergen and M. F. Porter. Probabalistic Models of Indexing and Searching. In Proc. of ACM-BCS Symposium on Research and Development in Information Retrieval, Cambridge, England, 1980.

[Rocchio 1964]
Rocchio, Jr., Joseph J. Possible Time-Sharing Organization for a SMART Retrieval System. Information Storage and Retrieval, Vol. 7, Cornell University Department of Computer Science, Ithaca, New York, 1964.

[Rocchio 1966]
Rocchio, Jr., Joseph J. Document Retrieval Systems – Optimization and Evaluation. Doctoral thesis. Report ISR-10 to the National Science Foundation, Harvard Computation Laboratory, March 1966.

[Rocchio 1971]
Rocchio, Jr., Joseph J. Relevance Feedback in Information Retrieval. In *The SMART Retrieval System, Experiments in Automatic Document Processing*, edited by Gerard Salton, Prentice Hall, Inc., Englewood Cliffs, N.J., 1971.

[Rowe 1980a]
Rowe, Lawrence A. Data Abstraction from a Programming Language Viewpoint. In Proceedings of Workshop on Data Abstraction, Databases and Conceptual Modelling. Edited by Michael L. Brodie and Stephen N. Zilles, Association for Computing Machinery, Inc., 1980.

[Rowe 1980b]
Rowe, Lawrence A. Issues in the Design of Database Programming Languages. In Proceedings of Workshop on Data Abstraction, Databases and Conceptual Modelling. Edited by Michael L. Brodie and Stephen N. Zilles, Association for Computing Machinery, Inc., 1980.

[Rush, Salvador & Zamora 1971]
Rush, J. E., R. Salvador and A. Zamora. Automatic Abstracting and Indexing. II: Production of Indicative Abstracts by Application of Contextual Inference and Syntactic Coherence Criteria. *Journal of the American Society for Information Science*, Vol. 22, No. 4, July-August 1971, pages 260-274.

[Sager 1975]
Sager, N. Sublanguage Grammars in Science Information Processing. *Journal of the American Society for Information Science*, Vol. 26, No. 1, January-February 1975, pages 10-16.

[Salton 1963]
Salton, Gerard. Associative Document Retrieval Techniques using Bibliographic Information. *Journal of the ACM*, Vol. 10, No. 4, October 1963, pages 440-457.

[Salton & Lesk 1965]
Salton, Gerard and Michael E. Lesk. The SMART Automatic Document Retrieval System: An Illustration. *Communications of the ACM*, Vol. 8, No. 6, June 1965, pages 391-398.

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

[Salton & Lesk 1968]
Salton, Gerard and Michael E. Lesk. Computer Evaluation of Indexing and Text Processing. *Journal of the ACM*, Vol. 15, No. 1, January 1968, pages 8-36.

[Salton 1969]
Salton, Gerard. A Comparison between Manual and Automatic Indexing Methods. *American Documentation*, Vol. 20, No. 1, January 1969.

[Salton, ed. 1971a]
Salton, Gerard, editor. *The SMART Retrieval System: Experiments in Automatic Document Processing.* Prentice-Hall, Inc., Englewood Cliffs, N.J., 1971.

[Salton 1971b]
Salton, Gerard. Automatic Indexing Using Bibliographic Citations. *Journal of Documentation*, Vol. 27, No. 2, June 1971, pages 98-110.

[Salton 1971c]
Salton, Gerard. Relevance Feedback and the Optimization of Retrieval Effectiveness. In *The SMART Retrieval System, Experiments in Automatic Document Processing*, edited by Gerard Salton, Prentice Hall, Inc., Englewood Cliffs, N.J., 1971.

[Salton 1971d]
Salton, Gerard. Cluster Search Strategies and the Optimization of Retrieval Effectiveness. In *The SMART Retrieval System, Experiments in Automatic Document Processing*, edited by Gerard Salton, Prentice Hall, Inc., Englewood Cliffs, N.J., 1971.

[Salton 1972a]
Salton, Gerard. Dynamic Document Processing. *Communications of the ACM*, Vol. 15, No. 7, July 1972, pages 658-668.

[Salton 1972b]
Salton, Gerard. A New Comparison Between Conventional Indexing (Medlars) and Text Processing (SMART). *Journal of the American Society for Information Science*, Vol. 23, No. 2, 1972, pages 75-84.

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

[Salton, Wong & Yang 1975]
Salton, Gerard, A. Wong and C. S. Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, Vol. 18, No. 11, November 1975, pages 613-620.

[Salton, Yang & Yu 1975]
Salton, Gerard, C. S. Yang and C. T. Yu. A Theory of Term Importance in Automatic Text Analysis. *Journal of the American Society for Information Science*, Vol. 26, No. 1, January-February 1975, pages 33-44.

[Salton & Waldstein 1977]
Salton, Gerard and Robert Waldstein. Term Relevance Weights in On-Line Information Retrieval. Technical Report 77-316, Cornell University Department of Computer Science, Ithaca, New York, 1977.

[Salton & Bergmark 1977]
Salton, Gerard and Donna Bergmark. Clustered File Generation and Its Application to Computer Science Taxonomies. In *Information Processing 77*, edited by B. Gilchrist, IFIP, North-Holland Publishing Company, 1977, pages 441-445.

[Salton & Wong 1978]
Salton, Gerard and Anita Wong. Generation and Search of Clustered Files. *ACM Transactions on Database Systems*, Vol. 3, No. 4, December 1978, pages 321-346.

[Salton 1979]
Salton, Gerard. Suggestions for a Uniform Representation of Query and Record Content in Data Base and Document Retrieval. Technical Report 79-363, Cornell University Department of Computer Science, Ithaca, New York, 1979.

[Salton & Bergmark 1979]
Salton, Gerard and D. Bergmark. A Citation Study of the Computer Science Literature. Technical Report 79-364, Cornell University Department of Computer Science, Ithaca, New York, 1979.

[Salton 1980]
Salton, Gerard. The SMART System 1961-1976: Experiments in Dynamic Document Processing. In *Encyclopedia of Library and Information Science, Vol. 28*, 1980, pages 1-36.

[Salton & Bergmark 1980]
Salton, Gerard and D. Bergmark. Parallel Computations in Information Retrieval. Technical Report 80-439, Cornell University Department of Computer Science, Ithaca, New York, 1980.

[Salton & Wu 1981]
Salton, Gerard and Harry Wu. A Comparison of Inverted File Searching with Cluster Tree Search Operations. Unpublished Report, Cornell University Department of Computer Science, Ithaca, New York, 1981.

[Salton, Wu & Yu 1981]
Salton, Gerard, H. Wu and C. T. Yu. The Measurement of Term Importance in Automatic Indexing. *Journal of the American Society for Information Science*, Vol. 32, No. 3, May 1981, pages 175-186.

[Salton, Fox & Wu 1982]
Salton, Gerard, E. Fox and H. Wu. Extended Boolean Information Retrieval. Technical Report 82-511, Cornell University Department of Computer Science, Ithaca, New York, August 1982.

[Salton, Buckley & Fox 1982]
Salton, Gerard, C. Buckley and E. A. Fox. Automatic Query Formulations in Information Retrieval. Technical Report 82-524, Cornell University Department of Computer Science, Ithaca, New York, October 1982.

[Salton, Buckley & Fox 1983]
Salton, G., C. Buckley and E. A. Fox. Automatic Query Formulations in Information Retrieval. *Journal of the American Society for Information Science*, Vol. 34, No. 4, July 1983, pages 262-280.

[Salton, Fox, Buckley & Voorhees 1983]
Salton, Gerard, E. A. Fox, C. Buckley and E. Voorhees. Boolean Query Formulation with Relevance Feedback. Technical Report 83-539, Cornell University Department of Computer Science, Ithaca, New York, January 1983.

[Salton & McGill 1983]
Salton, Gerard and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[Sandison 1975]
Sandison, A. References/Citations in the Study of Knowledge. *Journal of Documentation*, Vol. 31, No. 4, September 1975, pages 195-198.

[Sanford 1976]
Sanford, David H. Competing Semantics of Vagueness: Many Values Versus Super-Truth. *Synthese*, Vol. 33, 1976, pages 195-210.

[Schank & Yale-A.I.-Project 1975]
Schank, Roger C. and Yale-A.I.-Project. SAM: A Story Understander. Research Report 43, Yale University Department of Computer Science, New Haven, Connecticut, 1975.

[Schank 1976]
Schank, Roger C. Research at Yale in Natural Language Processing. Research Report 84, Yale University Department of Computer Science, New Haven, Connecticut, 1976.

[Schank & Abelson 1977]
Schank, R. C. and R. P. Abelson. *Scripts, Plans, Goals and Understanding*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.

[Schank, Kolodner & DeJong 1981]
Schank, Roger C., Janet L. Kolodner and Gerald DeJong. Conceptual Information Retrieval. In *Information Retrieval Research*, edited by R. N. Oddy, S. E. Robertson, C. J. van Rijsbergen and P. W. Williams, Butterworths, London, 1981, pages 94-116.

[Schiminovich 1971]
Schiminovich, S. Automatic Classification and Retrieval of Documents by Means of a Bibliographic Pattern Discovery Algorithm. *Information Storage and Retrieval*, Vol. 6, No. 6, May 1971, pages 417-435.

[Schmidt 1977]
Schmidt, Joachim W. Some High Level Language Constructs for Data of Type Relation. *ACM Transactions on Database Systems*, Vol. 2, No. 3, 1977, pages 247-261.

[Shopiro 1979]
Shopiro, Jonathan E. Theseus - A Programming Language for Relational Databases. *ACM Transactions on Database Systems*, Vol. 4, No. 4, 1979, pages 493-517.

[Siegel 1956]
Siegel, Sidney. *Non-parametric Statistics for the Behavioral Sciences.* McGraw-Hill, New York, 1956.

[Small 1973]
Small, Henry G. Co-Citation in the Scientific Literature: A New Measure of the Relationship Between Two Documents. *Journal of the American Society for Information Science*, Vol. 24, No. 4, July-August 1973, pages 265-269.

[Small 1974]
Small, H. G. Multiple Citation Patterns in Scientific Literature: The Circle and Hill Model. *Information Storage and Retrieval*, Vol. 10, No. 11-12, 1974, pages 393-402.

[Small & Koenig 1977]
Small, Henry G. and Michael E. D. Koenig. Journal Clustering Using a Bibliographic Coupling Method. *Information Processing and Management*, Vol. 13, No. 5, 1977, pages 277-288.

[Small 1978]
Small, Henry G. Cited Documents as Concept Symbols. *Social Studies of Science*, Vol. 8, 1978, pages 327-340.

[Small 1980]
Small, Henry. Co-Citation Context Analysis and the Structure of Paradigms. *Journal of Documentation*, Vol. 36, No. 3, September 1980, pages 183-196.

[Small 1981]
Small, Henry. The Relationship of Information Science to the Social Sciences: A Co-Citation Analysis. *Information Processing and Management*, Vol. 17, No. 1, 1981, pages 39-50.

[Small,S. 1980]
Small, Steven. Word Expert Parsing: A Theory of Distributed Word-Based Natural Language Understanding. TR-954, Department of Computer Science, Univ. of Maryland, College Park, Maryland, September 1980.

[SMART Project 1974]
SMART Project. User's Manual for the SMART Information Retrieval System. Technical Report No. 71-95 Revised April 1974, Cornell University Department of Computer Science, Ithaca, New York, 1974.

[Smith & Smith 1977]
Smith, John Miles and Diane C.P. Smith. Database Abstractions: Aggregation. *Communications of the ACM*, Vol. 20, No. 6, 1977, pages 405-413.

[Sneath & Sokal 1973]
Sneath, P. H. A. and R. R. Sokal. *Numerical Taxonomy: The Principle and Practice of Numerical Classification.* W. H. Freeman, San Francisco, 1973.

[Snedecor & Cochran 1980]
Snedecor, George W. and William G. Cochran. *Statistical Methods: Seventh Edition.* The Iowa State University Press, Ames, Iowa, 1980.

[Sparck Jones 1970]
Sparck Jones, Karen. Some Thoughts on Classification for Retrieval. *Journal of Documentation*, Vol. 26, No. 2, June 1970, pages 89-101.

[Sparck Jones 1971]
Sparck Jones, Karen. *Automatic Keyword Classifications.* Butterworths, London, 1971.

[Sparck Jones 1972]
Sparck Jones, Karen. A Statistical Interpretation of Term Specificity and its Application in Retrieval. *Journal of Documentation*, Vol. 28, 1972, pages 11-21.

[Sparck Jones & Kay 1973]
Sparck Jones, Karen and Martin Kay. *Linguistics and Information Science.* Academic Press, New York, 1973.

[Sparck Jones 1974]
Sparck Jones, Karen. Automatic Indexing. *Journal of Documentation*, Vol. 30, No. 4, 1974, pages 393-432.

[Sparck Jones & Van Rijsbergen 1976]
Sparck Jones, Karen and C. J. Van Rijsbergen. Information Retrieval Test Collections. *Journal of Documentation*, Vol. 32, No. 1, March 1976.

[Sparck Jones 1979]
Sparck Jones, Karen. Experiments in Relevance Weighting of Search Terms. *Information Processing and Management*, Vol. 15, 1979, pages 133-144.

[Sparck Jones 1979]
Sparck Jones, Karen. Search Term Relevance Weighting Given Little Relevance Information. *Journal of Documentation*, Vol. 35, No. 1, 1979, pages 30-48.

[Sparck Jones 1980]
Sparck Jones, Karen. Search Term Relevance Weighting: Some Recent Results. *Journal of Information Science*, Vol. 1, 1980, pages 325-332.

[Stonebraker et al. 1976]
Stonebraker, Michael et al. The Design and Implementation of INGRES. *ACM Transactions on Database Systems*, Vol. 1, No. 3, September 1976.

[Tague, Nelson & Wu 1981]
Tague, Jean, Michael Nelson and Harry Wu. Problems in the Simulation of Bibliographic Retrieval Systems. In *Information Retrieval Research*, edited by R. N. Oddy, S. E. Robertson, C. J. van Rijsbergen and P. W. Williams, Butterworths, London, 1981, pages 236-255.

[Tahani 1976]
Tahani, Vahollah. A Fuzzy Model of Document Retrieval Systems. *Information Processing and Management*, Vol. 12, 1976, pages 177-187.

[Tahani 1977]
Tahani, Vahollah. A Conceptual Framework for Fuzzy Query Processing: A Step Toward Using Intelligent Database Systems. *Information Processing and Management*, Vol. 13, 1977, pages 289-303.

[Todd 1976]
Todd, S. J. P. The Peterlee Relational Test Vehicle - a system overview. *IBM Systems Journal*, Vol. 15, No. 4, 1976, pages 285-307.

[Tsichritzis 1982]
Tsichritzis, D. Form Management. *Communications of the ACM*, Vol. 25, No. 7, July 1982, pages 453-478.

[Van Rijsbergen 1974]
Van Rijsbergen, C. J. Further Experiments with Hierarchic Clustering in Document Retrieval. *Information Storage and Retrieval*, Vol. 10, No. 1, January 1974, pages 1-14.

[Van Rijsbergen & Croft 1975]
Van Rijsbergen, C. J. and W. B. Croft. Document Clustering: An Evaluation of Some Experiments with the Cranfield 1400 Collection. *Information Processing and Management*, Vol. 11, 1975, pages 171-182.

[Van Rijsbergen 1977]
Van Rijsbergen, C. J. A Theoretical Basis for the Use of Co-Occurrence Data in Information Retrieval. *Journal of Documentation*, Vol. 33, No. 2, June 1977, pages 106-119.

[Van Rijsbergen 1979]
Van Rijsbergen, C. J. *Information Retrieval: Second Edition.* Butterworths, London, 1979.

[Van Rijsbergen, Robinson & Porter 1980]
Van Rijsbergen, C. J., S. E. Robinson and M. F. Porter. New Models in Probabalistic Information Retrieval. British Library R & D Report No. 5587, Cambridge, England, 1980.

[Verhoeff, Goffman & Belzer 1961]
Verhoeff, J., W. Goffman and J. Belzer. Inefficiency of the Use of Boolean Functions for Information Retrieval Studies. *Communications of the ACM*, Vol. 4, No. 12, December 1961.

[Vernimb 1977]
Vernimb, Carlo. Automatic Query Adjustment in Document Retrieval. *Information Processing and Management*, Vol. 13, No. 6, 1977, pages 339-353.

[Wagner 1974]
Wagner, R. A. Order-n Correction for Regular Languages. *Communications of the ACM*, Vol. 17, No. 5, May 1974.

[Waller & Kraft 1979]
Waller, W. G. and Donald H. Kraft. A Mathematical Model of a Weighted Boolean Retrieval System. *Information Processing and Management*, Vol. 15, No. 5, 1979, pages 235-245.

[Walker, Karlgren & Kay, eds. 1977]
Walker, D. E., H. Karlgren and M. Kay, editors. *Natural Language in Information Science*. FID Publication 551, Skriptor, Stockholm, 1977.

[Walker 1979]
Walker, Donald E. Information Retrieval on a Linguistic Basis. In *Aspects of Automatized Text Processing, Papers in Textlinguistics Band 17*, edited by Sture Allén and János S. Petöfi, Helmut Buske Verlag, Hamburg, 1979.

[Waltz 1978]
Waltz, D. L. An English Language Question Answering System for a Large Relational Data Base. *Communications of the ACM*, Vol. 21, No. 7, July 1978, pages 526-539.

[Wasserman 1980]
Wasserman, Anthony I. The Extension of Data Abstraction to Database Management. In Proceedings of Workshop on Data Abstraction, Databases and Conceptual Modelling. Edited by Michael L. Brodie and Stephen N. Zilles, Association for Computing Machinery, Inc., 1980.

[Weinberg 1974]
Weinberg, Bella Hass. Bibliographic Coupling: A Review. *Information Storage and Retrieval*, Vol. 10, No. 5/6, 1974, pages 189-196.

[White 1981]
White, Howard D. Cocited Author Retrieval Online: An Experiment with the Social Indicators Literature. *Journal of the American Society for Information Science*, January 1981.

[Wiener 1965]
Wiener, Norbert. *Cybernetics: Control and Communication in the Animal and the Machine, Second Edition*. M.I.T. Press, New York, 1965.

[Williams, ed. 1982]
Williams, Martha E., editor. *Computer-Readable Databases: A Directory and Data Sourcebook 1982 Edition*. Knowledge Industry Publications, Inc., New York, 1982.

[Williamson & Williamson 1970]
Williamson, D. and R. Williamson. A Prototype On-Line Document Retrieval System. Information Storage and Retrieval, Vol. 18, Cornell University Department of Computer Science, Ithaca, New York, 1970.

[Williamson, Williamson & Lesk 1971]
Williamson, D., R. Williamson and M. E. Lesk. The Cornell Implementation of the SMART System. In *The SMART Retrieval System, Experiments in Automatic Document Processing*, edited by Gerard Salton, Prentice Hall, Inc., Englewood Cliffs, N.J., 1971.

[Williamson 1974]
Williamson, Robert Edward. *Real-Time Document Retrieval.* Cornell University Ph.D. Thesis, June 1974.

[Wirth 1971]
Wirth, N. Program Development by Stepwise Refinement. *Communcations of the ACM*, Vol. 14, No. 4, April 1971.

[Woodfill et al. 1981]
Woodfill, John et al. *INGRES: Version 7 Reference Manual.* University of California at Berkeley, 1981.

[Wu 1981]
Wu, Harry Chih Chien. *On Query Formulation in Information Retrieval.* Cornell University Ph.D. Thesis, Xerox University Microfilms, Ann Arbor, Michigan, January 1981.

[Wu & Salton 1981]
Wu, Harry and Gerard Salton. The Estimation of Term Relevance Weights Using Relevance Feedback. Unpublished Report, Cornell University Computer Science Department, Ithaca, New York, 1981.

[Yip 1979]
Yip, Man-Kam. *An Expert System for Document Retrieval.* M.I.T. M.S. Thesis, 1979.

[Yu & Salton 1976]
Yu, Clement and Gerard Salton. Precision Weighting: An Effective Automatic Indexing Method. *Journal of the ACM*, Vol. 23, No. 1, 1976, pages 76-88.

[Yu & Salton 1977]
Yu, Clement and Gerard Salton. Effective Information Retrieval Using Term Accuracy. *Journal of the ACM*, March 1977.

[Yu, Luk & Siu 1979]
Yu, C. T., W. S. Luk and M. K. Siu. On Models of Information Retrieval Processes. *Information Systems*, Vol. 4, No. 3, 1979, pages 205-218.

[Yu, Buckley, Lam & Salton 1983]
Yu, C. T., C. Buckley, K. Lam and G. Salton. A Generalized Term Dependence Model in Information Retrieval. Technical Report TR83-543, Cornell University Department of Computer Science, Ithaca, New York, February 1983.

[Zadeh 1965]
Zadeh, L. A. Fuzzy Sets. *Information and Control*, Vol. 8, 1965, pages 338-353.