A NEW CLUSTERING SCHEME
AND
ITS USE IN AN INFORMATION RETRIEVAL SYSTEM
INCORPORATING THE SUPPORT OF A DATABASE MACHINE

A DOCTOR OF PHILOSOPHY THESIS

in

Computer Engineering
Middle East Technical University

By

Fazlı CAN

November, 1984

Approval of the Graduate School of Natural and Applied Sciences.

Prof.Dr. Bilgin KAFTANOĞLU
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Doctor of Philosophy in Computer Engineering.

Prof.Dr. Ziya AKTAŞ
Chairman of the Department

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Doctor of Philosphy in Computer Engineering.

Assoc.Prof.Dr.Esen A.ÖZKARAHAN
Supervisor

Examining Committee in Charge :

Assoc.Prof.Dr. Mehmet BARAY

Prof.Dr. Ziya AKTAŞ

Assoc.Prof.Dr. Müslim BOZYİĞİT

Assoc.Prof.Dr. Ayşe KİPER

Assoc.Prof.Dr. Aydın KÖKSAL

ABSTRACT

A NEW CLUSTERING SCHEME AND ITS USE IN AN INFORMATION RETRIEVAL SYSTEM INCORPORATING THE SUPPORT OF A DATABASE MACHINE

CAN, Fazlı
Ph.D. in Computer Engineering
Supervisor: Assoc.Prof.Dr. Esen A. ÖZKARAHAN
Nov. 1984, 288 pages

The need for immediate and accurate access to the current literature at one side and the information explosion on the other side have caused the development of information retrieval systems. In this work, information retrieval problem is studied and new concepts and methodologies are proposed for its solution.

The new proposals are cover coefficient and cluster seed power concepts and the methodologies for estimating the number of clusters within a collection and the number of members within a cluster. These concepts and methodologies are used in a new single-pass clustering algorithm. A multi-pass clustering algorithm is introduced to show the validity of the cover coefficient concept for clustering purposes. In the thesis, the complexity analysis of the algorithms, a new centroid generation policy in connection with the new cover coefficient concept are presented. An algorithm for the maintenance of the clusters in expanding document collection environments and its complexity analysis are also presented.

The similarity and stability concepts for clustering algorithms are introduced, then the clustering algorithms are analyzed by a set of experiments with respect to these concepts. For the purpose of the experiments, a document collection of 167 articles from the ACM-TODS publications has been constructed. The characteristics of the collection, the findings of the experiments and some observed basic relationships are illustrated in detail.

In the thesis, an information system model which integrates the information retrieval and database management systems is proposed. Unlike the previous studies aimed at this purpose, which more or less reduce one system into the other, the proposed model aims to accomplish this integration by a synthesis of the techniques and methodologies of both systems. For this purpose, a database machine, the Relational Associative Processor (RAP), is enhanced with the new text retrieval instructions. Context sensitive free text retrieval operations are implemented by using the new instructions. In the model, a clustering subsystem and a conceptual data model are used for information retrieval purposes. The performance of the database machine in text retrieval operations and a comparative performance evaluation of the single-pass and the multi-pass clustering algorithms in information retrieval are presented.

Additional concepts/methodologies that utilize cover coefficient concepts are also introduced in the thesis.

Keywords: *Information (text,document) retrieval systems, cover coefficient, cluster seed power,clustering algorithms, complexity analysis of algorithms, similarity and stability of clustering algorithms, centroid generation, database management systems, text retrieval computers, RAP database machine, term discrimination value, document significance value.*

ÖZET

YENİ BİR KÜMELEME YÖNTEMİ
VE
VERİ TABANI BİLGİSAYARI DESTEKLİ
BİR BİLGİ ERİŞİM SİSTEMİNDE KULLANIMI

CAN, Fazlı
Doktora Tezi, Bilgisayar Müh. Bölümü
Tez Yöneticisi: Doç.Dr. Esen A.ÖZKARAHAN
Kasım 1984, 288 Sayfa

Son yayınlara anında ve doğru ulaşım gereksinimi ve bilgi patlaması, bilgi erişim sistemlerinin geliştirilmesine yol açmıştır. Bu çalışmada, bilgi erişim sorunu incelenmiş ve çözümü için yeni kavram ve yöntemler önerilmiştir.

Yeni öneriler, kapsama katsayısı ve küme çekirdek gücü kavramları ile bir derlemdeki küme sayısının ve kümelerin üye sayılarının kestirimini amaçlayan yöntemlerdir. Bu kavram ve yöntemler, yeni bir tek-geçişli kümeleme algoritmasında kullanılmıştır. Kapsama katsayısının kümeleme algoritmalarında kullanım geçerliliğini göstermek için çok-geçişli bir kümeleme algoritması önerilmiştir. Tezde, algoritmaların karmaşıklık çözümlemesi ve kapsama katsayısı kavramına ilişkin bir ortaç oluşturma yöntemi sunulmuştur. Büyüyen derlem ortamlarında kümelerin bakımında kullanılabilecek bir algoritma ve karmaşıklık çözümlemesi de ayrıca verilmiştir.

Kümeleme algoritmalarında benzerlik ve kararlılık kavramları anlatıldıktan sonra önerilen algoritmalar bir takım deneyler ile bu kavramlar açısından incelenmiştir. Deneyler için 167 ACM-TODS yayınından bir derlem oluşturulmuş; derlemin özellikleri, deneylerin bulguları ve ortaya çıkardığı temel ilişkiler ayrıntılı olarak anlatılmıştır.

Tezde, bilgi erişim ve veritabanı yönetim sistemlerini birleştiren bir bilgi sistemi modeli önerilmiştir. Bir sistemi ötekine indirgeyen daha önceki çalışmalardan farklı olarak, önerilen model bu birleştirmeyi her iki sistemin yöntemlerinin bir sentezi olarak gerçekleştirmektedir. Bu amaçla, bir veritabanı bilgisayarı olan Relational Associative Processor (RAP), yeni belge erişim komutlarının eklenmesiyle genişletilmiştir. Bağlama duyarlı özgür belge erişim işlemleri, yeni komutların kulanımıyla RAP üzerinde gerçekleştirilmiştir. Önerilen yapıda, bilgi erişim amaçları için kavramsal bir bilgi modeli ve kümeleme alt sistemi kullanılmaktadır. Veritabanı bilgisayarının belge erişim işlemlerindeki başarısı ile tek-geçişli ve çok-geçişli kümeleme algoritmalarının bilgi erişimdeki başarılarının karşılaştırmalı olarak değerlendirilmesi ayrıca sunulmuştur.

Kapsama katsayısı kavramını kullanan kimi kavram ve yöntemler de tezde anlatılmaktadır.

Anahtar Sözcükler: *bilgi (metin, belge) erişim sistemleri, kapsama katsayısı, küme çekirdek gücü, kümeleme algoritmaları, algoritmaların karmaşıklık çözümlemesi, kümeleme algoritmalarında benzerlik ve kararlılık, ortaç üretimi, veritabanı yönetim sistemleri, belge erişim bilgisayarları, RAP veritabanı bilgisayarı, terim ayırt etme değeri, belge önem değeri.*

ACKNOWLEDGEMENTS

I am grateful to my supervisor Dr.Esen A. Özkarahan for his appreciation, guidance, and the research environment provided by his assistance.

I would also like to thank Dr.William E. Lewis for his assistance in providing me the financial support in the form of a research assistantship and an industrial fellowship from Arizona State University. I should also thank Dr (late) Lee Wilkinson of Intel, for his assistance in providing me the summer hire engineer position at the Intel Corporation.

Special thanks are due to Dr. Mehmet Baray for his valuable suggestions and corrections in some sections of the manuscript.

I would also like to thank Dr. Ziya Aktaş for his assistance and understanding for the finalization of the thesis.

I am grateful to the Department of Electrical Engineering and the Department of Computer Engineering of Middle East Technical University for their understanding at the various stages of my dissertation work.

I should also thank my family, for all their support in completing my Ph.D. degree.

The author also wishes to express his thanks to Meral Taşdemir for her careful typing. (The author is an admirer of her American style job responsibility.)

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

TABLE OF CONTENTS

Page

Page

Page

I. INTRODUCTION

The expansion of the scientific literature, which is usually referred to as "information explosion", is a generally accepted fact. Because of its size not all of the available information can be examined. As a result researchers are usually unaware about the findings of others. On the other hand, industry and government are waiting for the immediate solutions to their problems. This led to the implementation of the information systems, especially information retrieval systems. In short, an information retrieval system provides fast and accurate access to bibliographic data. By doing so, it helps to researchers by making them aware of the current literature and the current state of knowledge.

This thesis is about information retrieval systems. Its main contributions will be illustrated in Chapters 3 through 8.

During the dissertation work, the author had the opportunity of studying and enjoying the facilities of two different academic institutions located in two different countries. After the observations made in these institutions, it became the idea of the author that, the best way of achieving good works from dissertations is to have good research facilities and a mechanism of rewarding. The latter will give the motivation, the former will provide the results.

1.1. OUTLINE OF THE THESIS

The thesis consists of nine chapters. A brief summary of Chapters 2 through 9 is given in the following.

Chapter 2 is an overview chapter. Firstly, a definition for information systems is given. In this chapter, information systems are divided into three parts. The objective of these systems, the methodology of information representation, and systems' characteristics are given in a comparative manner. In this chapter information retrieval systems are presented in detail. The reasons of existence of information retrieval systems and information explosion are reviewed. Performance evaluation criteria and the hardware solution for information retrieval systems are also illustrated. In the last section of Chapter 2, the characteristics of the databases used in these systems and some commercial and experimental information retrieval systems are also presented.

Chapter 3 covers the new clustering algorithms and the related concepts. In this chapter, firstly the notion of clustering problem is introduced. This is followed by a classification of clustering algorithms, an illustration of evaluation criteria of clustering algorithms, and an overview of generation of cluster representatives. The use of clustering in information retrieval is also emphasized in this chapter. The remainder of the chapter is devoted to the presentation of some new concepts for clustering and two new clustering algorithms. The first one is a single-pass algorithm, the second one is a multi-pass algorithm. The initiation of the algorithms depends on the same concepts. However, document assignment to the cluster seeds is done by using two different criteria. For document assignment to the seeds, the single-pass algorithm uses the new concept called "cover coefficient"; the multi-pass algorithm uses a conventional "similarity" concept. A new cluster representative generation policy, the complexity analysis of the algorithms are also presented. An algorithm for cluster maintenance and the characteristics of the illustrated algorithms are presented in detail.

Chapter 4 contains the similarity and stability analysis of the two algorithms presented in Chapter 3. Firstly, an informal definition for similarity and stability of clustering algorithms are given. Secondly, two metrics for similarity and stability analysis are illustrated. This is followed by experiments whose main purpose is to test the similarity and stability of the two clustering algorithms. A document collection from 167 ACM-TODS (Association for Computing Machinery-Transactions on Database Systems) was constructed for the experiments. The general characteristics of the document collection and the findings of the experiments and the basic relationships observed in connection with indexing and clustering are illustrated in detail.

Chapter 5 first gives a chronological overview of the RAP database machine, which is a cellular parallel associative processor for database management, current version of which is called RAP.3. This overview also presents a brief timing analysis of the RAP.3. The relationally complete RAP Assembler Language is briefly explained in this chapter. This is followed by the illustration of the text retrieval operations with the RAP database machine. For this purpose, the text string and the physical data structure formats in RAP, the new RAP instructions for information retrieval, the realization of these instructions, and the performance evaluation of RAP in information retrieval instructions are presented.

Chapter 6 is devoted to an information system model which integrates the information retrieval and database management systems. The model relies on a clustering subsystem for database partitioning and relation fragmentation. For the implementation, the single-pass algorithm, which is presented in Chapter 3 is proposed. The support architecture for the search operations is the RAP.3 database machine.

Chapter 7 contains a comparative performance analysis of the single-pass and the multi-pass clustering algorithms in information retrieval. For this purpose the experimental document collection, which is introduced in Chapter 4, is used. The performance of the single-pass algorithm validates its usage for information retrieval purposes.

Chapter 8 contains some additional concepts/methodologies that utilize cover coefficient concept. These are the following: some methodologies for testing the correspondence of document and term clusters, the use of the coupling coefficient concept for term discrimination purposes, an approach for finding the optimum weights for indexing terms, and a method of using the new concepts introduced in the thesis for the construction of effective indexing vocabularies.

The summary of the thesis along with the contributions of the thesis and directions for further research are given in Chapter 9.

2. INFORMATION SYSTEMS

A generally accepted functional definition of an information system is the following: An information system retrieves an information from its database according to a used query. The form of the information depends on the type of the information system. There are variety of systems that satisfy this definition. Basically, there are three types of information systems. These are Information (Text or Document) Retrieval Systems (IRS), Database Management Systems (DBMS), and Question Answering Systems (QAS). In this study our concern is IRS and a combination of IRS and DBMS. In the text, Information Retrieval (IR), Text Retrieval (TR), and Document Retrieval (DR) phrases will be used interchangeably. Some other kind of information systems, are management information systems and office information systems.

A description of an information system can be given as in Figure 2.1. In this figure "Data" represents the raw information which is stored in the database of the information system. Here, the database means the collection of "organized" data which is used to respond to the queries submitted to the system. The "Updates" indicates the addition of new data, or deletion of the old data, or modifications on the already existing data. The "Query", which is submitted by a

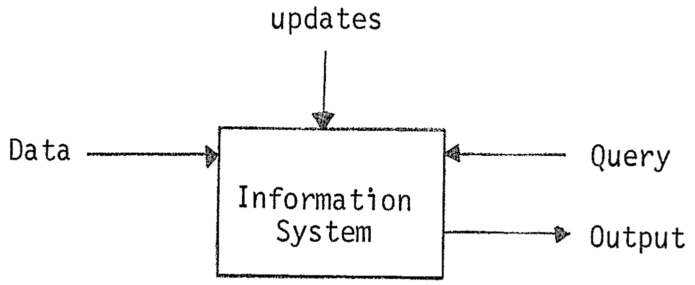


Figure 2.1 - Description of an information system

user, indicates the information need of the user. The "Output" is produced according to the specifications of the query. This functional description for an information system is valid for all varieties. However, each system will have its unique features in each component.

In an IR system the output will be the citations or whereabouts of a document which satisfies the user query. However, in case of a DBMS, the output is an factual information, and in a QAS system a knowledge is produced for the user query (or more correctly, user question).

2.1. INFORMATION SYSTEMS OTHER THAN IR SYSTEMS

In this section, a brief description of the information systems other than IR systems will be given.

2.1.1. Database Management Systems

A database can be defined as a collection of structured operational data stored in the secondary storage and used by the application systems of a particular organization. A DBMS is the software that provides all the accesses and manipulations on the database of an enterprise |28|.

In a DBMS, information is kept in data fields with some specific type and length constraints. These data fields (or the attributes) are stored together to form record types. It is evident that there might be more than one record type. A record type's value extension is called a record occurrence. All of these record occurrences will form the database of an enterprise.

For the organization and manipulations on an enterprise's data there are various file management methods |56|. However, a DBMS is different than a file management system. Its major distinctive features are the following |28|:

(a) Data independence is achieved. This achievement is in two respects : Physical data independence and logical data independence. Within this context, physical data independence means the ability to modify the storage structures and the related access paths for the stored data without affecting the applications and users. Similarly, logical data independence provides the capability of a DBMS to support various view of the database and modifications of these views or the underlying conceptual schema without affecting the physical structures.

(b) The amount of redundancy in the stored data is reduced, since the data is stored only once.

(c) Problems of inconsistency because of redundancy can be avoided to a certain extent.

(d) The stored data can be shared by different applications, and new applications can be developed to operate on the existing database.

(e) Standars can be enforced.

(f) Security restrictions can be applied.

(g) Data integrity can be maintained.

Remembering the features of a file management system the above characteristics of a DBMS will be more meaningful. In a file management system, every application has its own private file and correspondingly stored data and related programs. However, in the case of a DBMS the data of the organization is stored only once and the system shades the storage structure from the users. Each user is concerned with some part of the stored data which is relevant to him. The user do not need to know the physical data structures in order to write programs. This is achieved by data independence.

Furthermore, a DBMS provides a data definition language (DDL) to map the logical data structures into the physical data structures and a data manipulation language (DML) or a query language to specify data management operations on the database.

DBMSs can be classified according to their data models. The most recognized data models are the following:

a) the relational data model ;

b) the hierarchical data model ;

c) the network data model.

The relational data model is based on the mathematical theory of relations. A relation can be defined as follows : Given a collection of sets $D_1, D_2, \ldots, D_n$ (not necessarily distinct), R is a relation on these n sets if it is a set of ordered n-tuples $<d_1, d_2, \ldots, d_n>$ such that $d_1 \varepsilon D_1, d_2 \varepsilon D_2, \ldots, d_n \varepsilon D_n$. Sets $D_1, D_2, \ldots, D_n$ are the domains of R. The value of n indicates the degree of R. In other words, R is a relation on the sets $D_1, D_2, \ldots, D_n$ if it is a subset of the Cartesian product $D_1 \times D_2 \times \ldots \times D_n$ |18, 28|. In the relational model terminology, record fields are referred to as an "attribute". Associations among relations (record types) are provided by data values in attributes having intersecting domains.

The relational data model consists of relations. Unlike the mathematical relations, database relations are time varying since tuples may be inserted, deleted or updated.

In the hierarchical data model, data is arranged into tree structures. Each record type represents a node of a tree. The links between the nodes specify 1:N connections.

The rules to be obeyed in forming a hierarchical definition tree are the following |30|:

a) There is a single root record type.

b) The root record type may have any number of child record types.

c) Each child record type may also have any number of child record types.

d) For one occurrence of a given record type there may be any number of occurrences of each of its children.

e) No child record occurrence can exist without its parent.

The hierarchical definition tree forms a template to arrange the data. If information structure is hierarchical by nature, then it would be suitable to use the hierarchical data model, otherwise it has some disadvantages |30|.

In the network model there are two important concepts : record types, and the set types. Firstly record types are formed. The relationships between record types are represented by links. A link connecting two record types constitude a set type. Each set type has two record types : Owner record type, and member record type. Within a set type, there is 1:N relationship between the owner and member record occurrences. This means that each set occurrence has an owner and zero or more member record occurrences. For the implementation of many-to-many relationships, a link record type being the member of two set types can be introduced.

An owner record type of a set type may be a member record type of another set type.

The network data model is more general than the hierarchical data model, since certain restrictions of the hierarchical data model are removed (a record type my have more than one owner).

As can be seen from these descriptions a DBMS contains structured data.

For a query submitted, a factual information is provided by a DBMS (DBMSs are also referred to as fact retrieval systems). The same query can be formulated in different, but semantically equivalent ways.

The query processing is completely deterministic in a DBMS. Therefore, semantically equivalent queries will always generate the same output. A typical query to a DBMS can be the following : which is the largest city of the USA? If the database of the DBMS is equipped with the related information the "fact" (New York) will be returned.

2.1.2. Question Answering Systems

QA systems provide access to factual information about some specific area in a natural language environment. In QAS environment the natural language system queries of the users are analyzed. In this analysis linguistic methods are used. The meaning of the user query is understood by the system. The information stored in the system, i.e. knowledge, is searched by using heuristic methods. Which requires an inference. Following this a natural language response is generated and given back to the user. In the heuristic search, a QAS makes a deductive reasoning.

The representation of knowledge is a critical task and it can be done in a number of ways in a QAS. The most commons are the following :

(a) Semantic nets : In this method, graph theory is used. The nodes of the system represent the concepts, branches among the nodes specify the relationships between concepts.

(b) Procedural representation : Instead of having a complicated data structure for representing the knowledge, the same thing is done by procedures. Programs or procedures know the things, concepts. They simulate the thinking process. The artificial intelligence programming language PLANNER is an example for this approach |10|.

(c) Production based representation : In this representation the thinking process is defined as a prespecified sequence of transformation rules. A classical production system has three major components |10|. (1) global data base: stores the rules having the general form IF <condition> THEN <action> and facts or assertions about the particular problem being solved; (2) a rule base that contains the general knowledge about the problem domain; and (3) a rule interpreter that carries out the problem solving process.

It seems that the production based representation is the most hot subject in QAS or expert systems. There are various research activities on this subject, particularly on special architecture suited for production systems |64|. A brief introduction for the other kind of representations (such as predicate calculus and frames) are provided in reference |97|.

A QAS usually works in an interactive mode it gives some knowledge to the user, and gets some knowledge from the user. By this interaction process it extends both the knowledge of the user and knowledge stored in the system. A user can ask a question that requires deductive reasoning, such as : Why New York is the biggest city in the USA? If the QAS under use is equipped with necessary information, the system will produce a reasonable answer.

2.2. INFORMATION RETRIEVAL SYSTEMS

IRSs consider the retrieval of textual data, i.e. the information being processed consists of documents (such as papers, technical reports, patents, court decisions). Since their area of concentration is documents or text of documents, in general they are also referred to as document retrieval or text retrieval systems. In this section IR systems will be introduced in a somewhat detailed manner. Before introducing the IR systems the reader will be familiarized with the reasons of existence for them which is information explosion.

2.2.1. Information Explosion

The proliferation of the scientific literature, which is generally referred to as "information explosion", is a real problem for the scientists. The most remarkable example for this is the increase in the size of a collection of typical series. The number of different periodicals in typical library collections grew in number from 1000 in 1960 to about 1700 in 1974. This increase corresponds to a shelf length growth from 67 to 124 meters. By looking at this, one may say that the scientific literature doubled in bulk within 15 years |108|.

It is estimated that for an established field of science, it will take about 50 years to double its literature. A new field can grow enourmously, such that doubling will occur within every 4 or 5 years for several decades |108|. In short, it can easily be said that there is a huge increase in the size of scientific literature. Because of this, a scientist should narrow down his field of interest assuming that the amount of knowledge that can be assimilated by a scientist remains fixed. On the other hand, the immediate needs of technology and unavoidable growth of scientific literature necessitates a fast an accurate access to the literature. The answer for this need is the computerized access to the already existing information in an efficient and effective manner. This means that an individual should be able to reach to the latest publication which is related to his interest as fast as end as accurate as possible.

2.2.2. Characteristics of Information Retrieval Systems

Characteristics of IR systems can be observed by looking at their implementations. The design and use of an IR system involves four basic activities: information analysis, information organization, query analysis and search, and information retrieval and disseminaiton |90|. In fact, these are the activites observed in any type of information system with the peculiarities relevant to the type of the information system. The forth of the activities is appearent, the rest will be explained in the remainder of this section.

A functional description of an IR system is depicted in Figure 2.2. The following explanation will follow the concepts introduced in that figure.

In terms of Figure 2.1, data are the document collection. Each document is fetched by the IR system with respect to its title, abstract, full text, and other relevant features of the document. In Figure 2.2 the phrase "selected documents" is used, since none of the databases for IR systems can be absolutely exhaustive. This means that they do not include all the documents of their specialization area. This is because some of the documents have very marginal contribution to its field. Furthermore, the inclusion of all the documents of a field requires the scanning of enormous number of periodicals.

The task of information analysis is also known as indexing. In this phase a conceptual analysis of the documents takes place and various subject indexes or identifiers are assigned to the documents. Corresponding to each document an indexing term vector is generated.

Each entry of this vector indicates either the existence (nonexistence) of the corresponding term, or the importance (weight) of the term for the document. The former is a binary indexing strategy, the latter is a weighted indexing strategy. The indexing process either can be done automatically or manually by human experts.

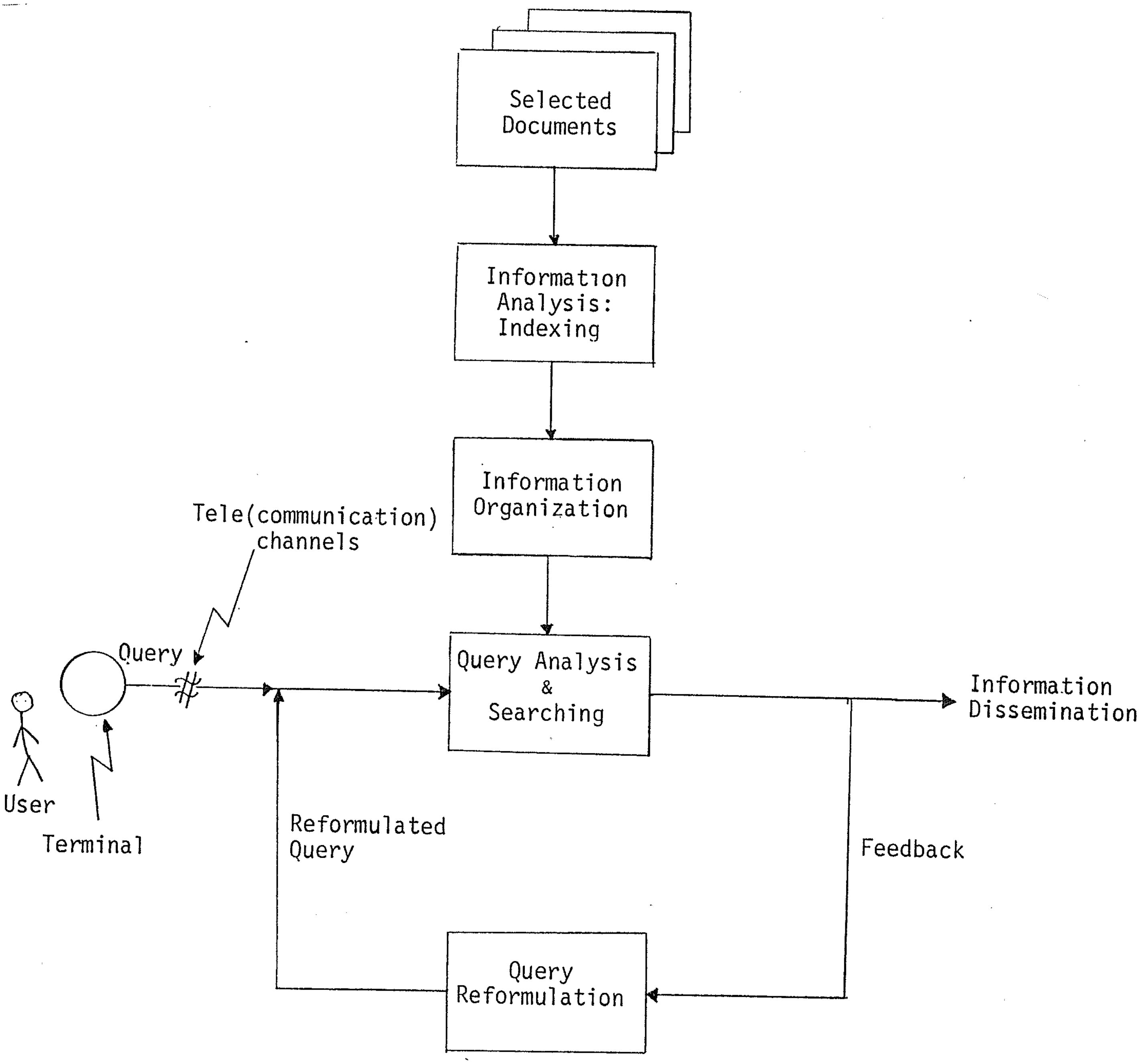

Selected Documents
Information Analysis: Indexing
Information Organization
Tele(communication) channels
Query
Query Analysis & Searching
Information Dissemination
User
Terminal
Reformulated Query
Feedback
Query Reformulation

Figure 2.2 - Functional description of an IRS.

An information organization process follows the indexing task. In commercial systems inverted file structure is used. In the experimental systems (where substantial amount of IR research is done on them) a document clustering is done. In clustering, the set of related documents are put into the same cluster.(A formal treatment of the term "clustering" will be done in the clustering chapter of the thesis). Document clusters are normally formed to facilitate the matching process that compares analyzed search requests with document identifiers and to simplify the retrieval of relevant documents. At the end of the clustering process a cluster representative or a "centroid" is formed to identify each cluster. The centroid of a cluster is a one dimensional vector, where each element of it shows the existence and/or weight of an indexing term.

The query analysis is made to initiate the necessary searching activity. The complexity of this analysis mainly depends on the form of the query. The query formulation (i.e., the way of expressing the query) can be done basically in two ways: by natural language sentences, or by Boolean expressions. Variations on these are possible |8,9,93|

In natural language query formulation a query vector, which indicates the existence and/or weight of the search terms, is generated after query analysis. This vector is compared with the document cluster centroids and the document clusters which are found similar to the user request are selected (possibly for further processing of similarity with the documents of the selected clusters).

In IR systems, which provide Boolean query formulation, the typical queries might be as follows: "A and B", "A or B", "A and not B", or "(A and B) or C". In these queries, the documents to be retrieved must include, respectively, both terms A and B, either term A or B, term A bût not B, and in the last one, documents must include the terms A and B or C. If the document indexes are organized as an inverted file, the operator "and", and "or" will be implemented by intersection and union operations of index pointers. The general procedure of searching an inverted file is summarized in Figure 2.3 |90|.

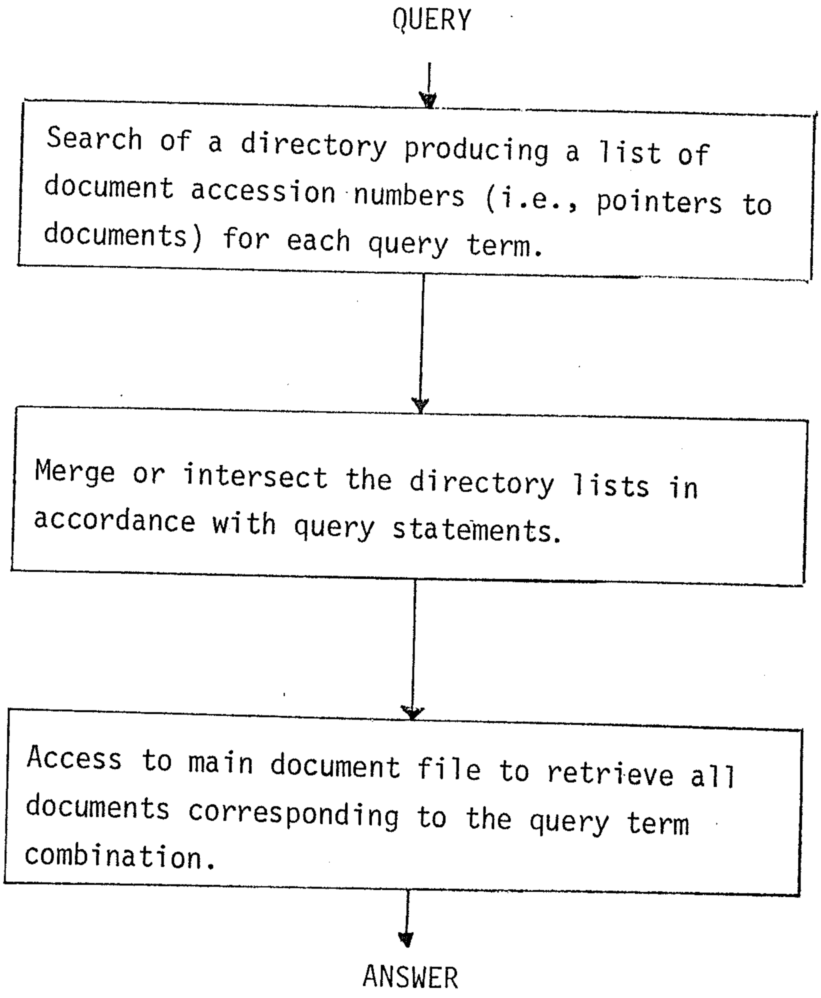

QUERY
Search of a directory producing a list of document accession numbers (i.e., pointers to documents) for each query term.
Merge or intersect the directory lists in accordance with query statements.
Access to main document file to retrieve all documents corresponding to the query term combination.
ANSWER

Figure 2.3 - General procedure for searching an inverted file.

In some of the text retrieval systems, the task of information analysis might be lightened: in such an approach, the documents are represented directly by their text. These kind of systems are called "free" or "full" text systems. Implementation of free text systems on von Neumann type machines introduces a heavy load of indexing requirement. Some possible query formulations in case of free text systems are given in the following:

(ENTITY adjacent RELATIONSHIP)
within sentence
(DATA adjacent MODEL)

The above query implies that the terms "ENTITY", "RELATIONSHIP", "DATA", and "MODEL" should occur within a sentence. The additional requirement is that terms "ENTITY" and "RELATIONSHIP" should appear adjacent to each other within a sentence. The same is valid for "DATA" and "MODEL". In the free text systems it is also possible to enforce fixed length don't care (FLDC) characters and variable length don't care (VLDC) characters to appear within the query terms. Assuming that the character? indicates one don't care character and the term * indicates a variable number of don't care characters, the following queries

(DISTRIBUTED? PROCESSING)
(DISTRIBUTED* PROCESSING)

will try to match, in the first query, the documents which contain the terms "DISTRIBUTED" and "PROCESSING" and between these two terms, one character with any value can appear; in the second query, between the same two terms, any number of characters (i.e., VLDC) can appear.

Another approach in query formulation is to assign weights to the terms. Instead of retrieving all of the documents, the documents, which exceed a predefined threshold value might be retrieved. Considering the query:

GENERALIZED(2) DATABASE(5)
FILE(2) ORGANIZATION(5)

In this query the numbers which appear in parentheses indicate the weights assigned to the terms which appear to the left of the number. If the retrieval threshold is 9, the output generated will be as follows:

a) documents containing all the four terms (sum of weights=14);

b) documents containing the terms "generalized", "database", and "file (sum of weights=9);

c) and etc.

In the above query, the documents which contain the single terms will not appear in the answer set, since their individual weights do not exceed the threshold value, which is 9. It is also possible to express a weighted threshold query in a semantically equivalent Boolean query form |3|.

The text retrieval systems also facilitate the use of truncated terms. Consider the following query:

COMPUT* and PROG*

In this query, the terms are truncated and * indicates that they might match with any character at the right hand side. Therefore, these terms might match with the following words: "computer", "computation", "computability", and "programming", "programmed", etc. The typical text retrieval operations, which are especially used in free text systems, are listed in Appendix-A |46|. As it is pointed out previously, in free text systems search strings can have "don't care" characters with fixed length and variable length. The search terms can be combined using Boolean operations, and threshold functions. Ranges can be specified, indicating that the terms must all appear in the same context (sentence) or within a given number of words. The order of the search terms (such as B after A) can also be specified. All of these make a very powerful query language. The proximity capability (alias context sensitivity) provides more meaningful use of "common" words. For example, if one searches for the documents which contain the phrase "National Science Foundation", only the documents containing the three consecutive words "National", "Science" and "Foundation" would be retrieved. The proximity is much stronger than the "and" operation. This is because all the documents which contain all of these three words do not satisfy the query. The free text retrieval operations increase precision and decrease ambiguity by permitting context sensitive queries as opposed to individual words (The text retrieval capability added to the RAP database machine in this work permits the context sensitive free text operations, see Chapter 5 and 6).

In a text retrieval environment the output of a single search effort may not lead to a satisfactory result. In such a case, a sequence of search operations might be carried out to approach the desired documents little by litle. This porcess of query reformulation to access the desired documents is called as "browsing" |92|.

During browsing, the original query is made more similar to the documents which are found relevant and more dissimilar to the document which are found nonrelevant by the user. The query reformulation is done either automatically or by the users. Generally, the former approach is used in experimental systems and the latter approach is used in commercial systems. The strategies of formulating better queries for information retrieval can be seen in |4|.

2.2.3. Performance Evaluation of Information Retrieval Systems

For the evaluation of IR systems six main measurable quantities are set |55,92,106|;

a) The recall of the system, i.e., the ability of the system to find the relevant documents for the users.

b) The precision of the system, i.e., the ability of the system not to retrieve the documents which are not relevant for the users.

c) The user effort, this is both physical and mental activity of the user in obtaining answers to his query.

d) The response time, which is the average time between the submission of user query and getting back the answers.

e) The form of presentation of the output.

f) The coverage, i.e., the extent with which the collection of the system includes the relevant material.

The above points are usually relevant to the evaluation of the commercial systems. An experimental system usually concentrates its attention to the first two topics, i.e., recall and precision.

They are the means of measuring the effectiveness of an IR system. Recall is the ratio of the number of relevant documents retrieved to the total number of relevant documents in the collection, and precision is the ratio of the number of relevant documents retrieved to the total number of documents retrieved. In terms of Figure 2.4, the definitions of recall and precision are as follows:

$$\text{Recall} = \frac{c}{b + c}$$

$$\text{Precision} = \frac{c}{c + d}$$

relevant		nonrelevant	
b	c	d	a
not retrieved	retrieved		not retrieved

Figure 2.4 - Partitioning of a document collection after the processing of a query

Recall and precision tend to vary inversely in searching. An effort to increase the recall will degrade precission. A typical plot of recall versus precision is show in Figure 2.5.

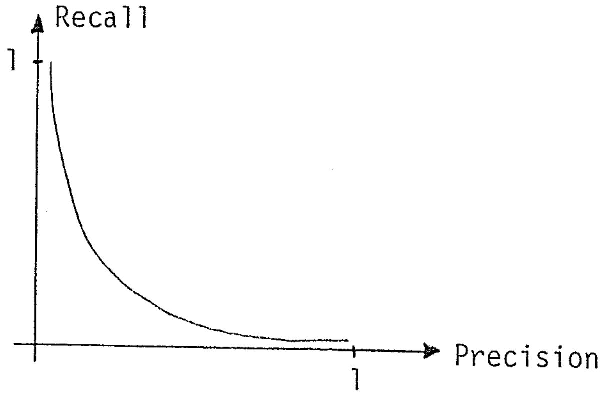


Figure 2.5 - Plot of recall versus precision

Other effectiveness measures for IR systems, such as the Swets model, the Robertson model, the Cooper model, the Smart's measures, and Van Rijsbergen's effectiveness measure can be seen in |106|.

2.2.4. Hardware Solutions For Text Retrieval

Text retrieval related hardware varies from partial to reasonably complete solutions to the problem. In the partial category, one can mention the proposals on constructing fast index processors to speed inverted list intersection and merge |43,45|.

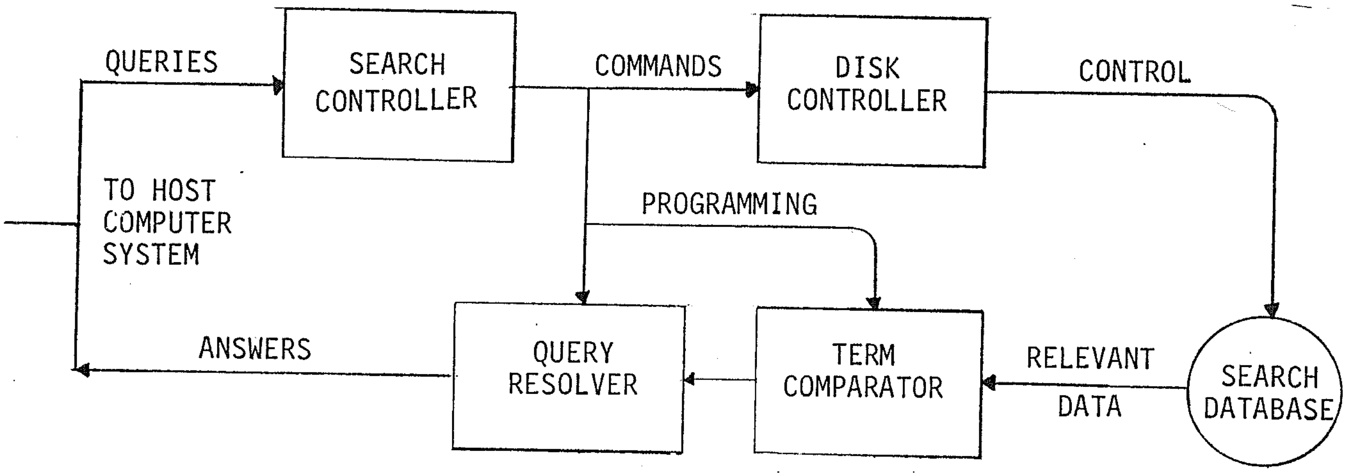


Figure 2.6 - Text retrieval computer

The hardware organizations specializing in text retrieval operations are called "text retrieval computers". In general, the text retrieval computers have the basic architecture presented in Figure 2.6 and this architecture is especially suitable for free text systems |46,47,49|.

In this organization, the text retrieval query issued by the user is taken by the "search controller" which controls the overall operation of the system. It sends the programming information to the units called "query resolver" and "term comparator". Search controller also sends some commands to a disk controller; these commands will fetch relevant data from the search database into the term comparator unit. The task of term comparator unit is very heavy, since it must perform a fast matching operation with the terms and phrases specified in the query on a very high volume of data. Further qualification tests (such as to see if the match occurs in the proper context, i.e., to see if the proximity of the match is proper with respect to a previous match, etc.) on data are done in the query resolver. The fully qualified data items are returned to the host computer as the answers. The answer set can be either pointers to documents or actual text of documents matching the query.

In the complete system architecture, the two major system components are the term comparator (or term matcher) and query resolver subsystems. Because the query resolver does not require much novel hardware and the system bottleneck rest in the term comparator, various proposals have concentrated on the term comparator. The following are the techniques used by the various proposals made to date:

a) Parallel comparators,

b) Associative memories,

c) Cellular arrays,

d) Finite state automata.

The items (a) and (b) can be considered together. In the parallel comparators approach, the term comparator sits in series with the data stream which is read off a mass memory such as disk. The data in passing, is buffered in a window and simultaneously made available to a number of parallel comparators |98|. The associative memory technique is a more advanced version of the former. Search terms are stored in an associative memory which acts as a term matcher synchronous with the data. The advantage of this is the ability to store a large number of search terms each of which corresponds to a comparator in the former approach. In reality, one uses a fast random access memory that emulates an associative memory. The majority of commercial systems of text retrieval hardware use this associative memory approach |6|.Two points to realize in this approach are: the difficulty of implementing context sensitive full text operations, especially fixed and variable don't cares and serial nature of the system.

In the cellular array approach, the basic building block is a cell element which can match single character of the searched term. The system is serial with data and cells operate by communicating with their immediate neighbours so that when a match occurs a number of consecutive cells produce match signals |19,61|. This approach requires too much dynamism and a large number of cells. The dynamic interconnection problem and data streaming for a large number of cells are non-trivial and costly issues.

The finite state automata (FSA) has some similarities with the cellular approach. Since ach state of the FSA is a term comporator. According to the input stream the FSA traces the states and if reaches to the final (target) state this means that the search term is matched. The FSA approach is used by the Central Intelligence Agency (CIA) in its High Speed Text search system |46,83|. There have been various optimizations for FSA implementations ranging from constructing separate FSA's for the starter, sequential, and index states to efficient addressing of the state blocks. A protitioned

FSA (PFSA) has also been proposed with the aim to synchronize with the data stream at disk speed using a minimum amount of buffer memory |43|.

In addition to the above efforts there is the database machine approach initiated in the progress of this dissertation work. For this purpose, the structure of the RAP, which can be classified as a parallel associative array, has been utilized. Several associative full text match instructions are added into the RAP language. For query resolution macro programs are written using the relational DBMS and the new text retrieval commands of RAP. The RAP combines the advantages of associativity, parallelism, programmable query resolution, and operation flexibity (due to its programmability).

The latest version of RAP, which is RAP.3, database machine uses parallel cells and parallel microprocessors in each cell. Each microprocessor executes firmware of query routines implementing DBMS and IR instructions. The entire device memory which is equal to the union of cell memories is made to work like an associative memory. The difference between the associative memory approach discussed earlier and the RAP.3 (associative) database machine approach is that in the former, search terms are placed in the associative memory and the text is streamed through it, whereas in the database machine the text is stored in the device memory which is a quasi associative memory so that it can be large to accommodate bulk data. In the microprocessors of each cell, efficient string search is performed.

The advantage of the RAP.3 database machine approach is the ability to provide both a relational DBMS and the IR system in an integrated manner in the basic architecture and the elimination of the query resolver which is an important component in the other system. The details of the implementation for the RAP.3 approach will be given in Chapter 5.

2.2.5. Examples for Information Retrieval System Implementations

It is hard to give an exhaustive list of information retrieval system implementations. Here some important and well known implementations will be mentioned after some introductory information. The commercial systems are MEDLARS (MEDLINE), STAIRS, BRS, and DIALOG. As an example for the experimental systems, the SMART system will be illustrated.

2.2.5.1. Databases Used for Information Retrieval

Currently there are more than 500 (online) databases available for the use of commercial IR systems |31|. About 200 of them are highly recognized |39|. The number of documents covered by them is about 70 million with an annual update rate of about 10 million items. Disregarding the duplications, one will be faced with a collection of 40 million unique documents with an update rate of 6 million unique new documents per year |39|. The growth of databases is given in Table 2.1.

The fields covered by IRS databases are in the whole range of knowledge with an emphasis on (i) Applied Sciences, (ii) Pure Sciences, and (iii) Medicine. The number of documents for these subjects is about 34 milion, 23 million, and 11 million documents, respectively. The other subjects covered by the databases and the number of documents in them are as follows: Agriculture (5.5 million); Social Sciences (including Statistics, Political Science, Economics, Law, Public Administration, Education, Commerce; etc.) (4 million); Philosopy, Psychology (635,000), History, Geography, Biography (380,000); Linguistic, Languages (300,000); Literature (300,000); Arts, Recreation, Music, etc. (220,000); Religion, Theology (50,000) |39|. The interdisiplinary databases contribute to more than one field. An extreme example for this is some theological documents in the NASA database |39|.

If one looks at the time span covered by the databases, it will be seen that 86% of them cover the last five years. About half of them cover the decate 1970-1980. Only 8% of them cover the last 20 years.

TABLE 2.1 - No. of Documents Available in Commercial Databases

1968	1972	1976	1980
<1/4 million	3 million	24 million	65 million

If one classifies the database sizes as small, medium and large according to the number of documents in them, the percentages of the number of databases in each class, respectively, are 46, 43, and 11. In this classification the databases which contain up to 100,000 records is assumed as small sized. The sizes of medium and large databases are arbitrarily taken as 100,000 up to 1 million, and higher than 1 million, respectively |39|.

The.majority of online database vendors are from the US, they are followed by the vendors of the UK, other European countries, and Canada.

An access charge for these databases is usually mandatory. For example, an access to National Library of Medicine's (NLM) MEDLINE database is 10 to 80 dollars per connect hour. A detailed information for recognized online databases can be found in |39,60|.

2.2.5.2. The MEDLARS (MEDLINE) Information Retrieval System

In late 1950's it is recognized by the NLM that a mechanized search on ever-increasing volume of library literature is necessary. The efforts resulted with MEDLARS (Medical Literature Analysis and Retrieval System) in 1964. The MEDLARS was a batch system. In 1970, approximately 18,000 searches made, and the turnaround to users was 40-60 days |31|. In 1971 MEDLINE (MEDLARS online) became operational.

There are 19 databases maintained by the NLM. MEDLINE and other commercial IR systems are conducting about 1.5 million searches each year |31|. About 860,000 documents are kept accessible online. Each month about 20,000 new documents are added to these databases by scanning 3000 periodicals. The number of documents in the databases is 3 million. The average time delay between the receipt of an item and its appearance in the database is 80 days. Seventy percent of the documents are in English. The annual cost of building the MEDLARS database was $ 2,000,000 in 1977 |59|.

The MEDLARS (MEDLINE) system has three inverted files: the INDEX file the POSITINGS file, and the DATA (also called HEADER) file, The DATA file contains the complete information for a document. Each document is identified by a unique number called computer assigned number (CAN).

The INDEX file contains all the search terms (such as terms from controlled vocabulary, dates, author names MeSH classification codes). An identifier attached to the search term indicates location of this term in the documents (author, or text, etc.). Another attached field gives the position in the posting file where information about this term begins, and the number of postings with this term.

The POSTINGS file gives the CAN number of documents related with a term.

The Boolean algebra rules are used for searching. The user is also allowed to use results of previous steps. Such as the following queries 1 and 2:

1: COMPUTER or PROGRAMMING and LANGUAGE

2: DATABASE and QUERY

To combine the results of query 1 and 2 one may say: 1 and 2.

The user is also allowed to do string search. A preliminary search is first conducted, then a string search is performed on the documents which are selected by this preliminary search. A string search can be very expensive, that is why the system employs this feature on restricted documents. Furthermore, the string search is allowed for a time slice, at the end of each time slice the user is informed by the system with an output containing the number of documents found so far containing the search string.

2.2.5.3. The STAIRS and BRS Information Retrieval Systems

The STAIRS (Storage and Information Retrieval System) is a program product of IBM. The users of the STAIRS uses the commercially available databases, since no database is supplied with the STAIRS system by IBM. STAIRS runs on the user's own computer |92|.

The STAIRS system consists of two programs:

a. Utility for database creation and maintenance;

b. An online retrieval system called AQUARIUS (a query and retrieval interactive utility system).

The system has the ability of doing both DBMS and IRS operation. For these SELECT and SEARCH modes are used, respectively. SELECT and SEARCH modes cannot be mixed. However, the output of one may be used by the other.

In the STAIRS system the inverted file approach is used. An entry is introduced into the directory file for each unique word. From this directory file there is a pointer to an inverted file containing information for each occurrence of the word; such as the document containing the word, within the document the paragraph, the sentence, and the word positions for the word. Synonym information for the words can also be introduced into the directory file.

The STAIRS retrieval system uses the free text of the documents or document abstract for searching. The search mode is entered by the SEARCH command. The query formulation is done by Boolean algebra. The followings are legitimate search statements:

```
PROGRAM
PROGRAM OR QUERY
PROGRAM$ AND QUERY$3
```

The $ sign indicates truncation, $3 indicates truncation with up to three unspecified characters. Some of the other operators are ADJ (indicating that two terms must be adjacent to each other), SAME (specifying that two terms must appear in the same paragraph), WITH (indicating that two terms must appear in the same sentence).

The select mode of the STAIRS system is used to perform DBMS operations on the formatted fields of data containing the values of record attributes. Another IBM product, IMS, is usually associated with SELECT mode.

The criticism on STAIRS system contains the following:it requires a large IBM system, a database software for DBMS operations, and a very large disk storage for the inversion of the free text.

Because of the above criticism on STAIRS a streamlining operation on it was performed and BRS (Bibliographic Retrieval Services) has resulted. It is a product of BRS Inc.

BRS does not have DBMS features, it is a document retrieval system. A search statement like the following is legal

```
1: PROGRAMMING ADJ LANG$
```

This query (query number 1) will find all the documents which contain the words PROGRAMMING and LANG (like LANGUAGES, LANGUAGE). This search statement can be modified like as follows:

2: 1. TI

In this case the search will be conducted only on the titles of the documents.

The BRS is now operating on about forty databases. The BRS users do not need to have their own IBM systems.

2.2.5.4. The DIALOG Information Retrieval System

The DIALOG is a product of Lockheed Information Systems. In 1980, 122 databases were available through the DIALOG system |92|. Like any other commercial system it is based on inverted files.

The system creates sets of documents by using the SELECT commands. These sets can be manipulated by the COMBINE command (The pocket quide of the DIALOG commands is also available in the appendices of |55|). For example SELECT COMPUTER and SELECT PROGRAMMING commands in DIALOG will create two sets of documents, set number 1 and 2 related with these commands. One may use COMBINE 1 AND 2 to have the effect of SELECT COMPUTER AND PROGRAMMING.

The DIALOG system allows truncation. For example COMPUT? will match with any of the following terms: COMPUTER, COMPUTING. The truncation character may be embedded inside a term. For example, WOM?N would be used to indicate both WOMAN and WOMEN.

The user can unite a select command like as follows: SELECT ENTITY (3 W) RELATIONSHIP. This command tries to find the documents which contain the keywords ENTITY and RELATIONSHIP, where the second term can be at most 3 words away from the first one.

The DIALOG system also uses field identifiers: author (AU), classification code (CC), corporate source (CS), document type (DT), journal name (JN); language (LA), publication year (PY), and update (UD).

2.2.5.5. The SMART Experimental Information Retrieval System

The SMART system was first developed between the years 1961 and 1964 at the Harward University |85|. It is the most recognized experimental system. The research related with the SMART system is under the supervision of Dr Salton and now he is at the Cornell University.

Unlike commercial systems which are restricted to inverted files, the new methods and ideas are usually used in the experimental systems. Here the features of the SMART system will be given very briefly |92|: (1) the text analysis and indexing is done by fully automatic methods; (2) the related documents are clustered; (3) the documents to be selected are first chosen by a similarity analysis between the query and the cluster representatives, then the query and the documents of the selected clusters are compared and the retrieved documents are listed in a ranked from according to their similarity with the query; (4) a feedback procedure is employed to improve the original user query. A simplified SMART system flowchart is given in Figure 2.7 (adapted from |92|).

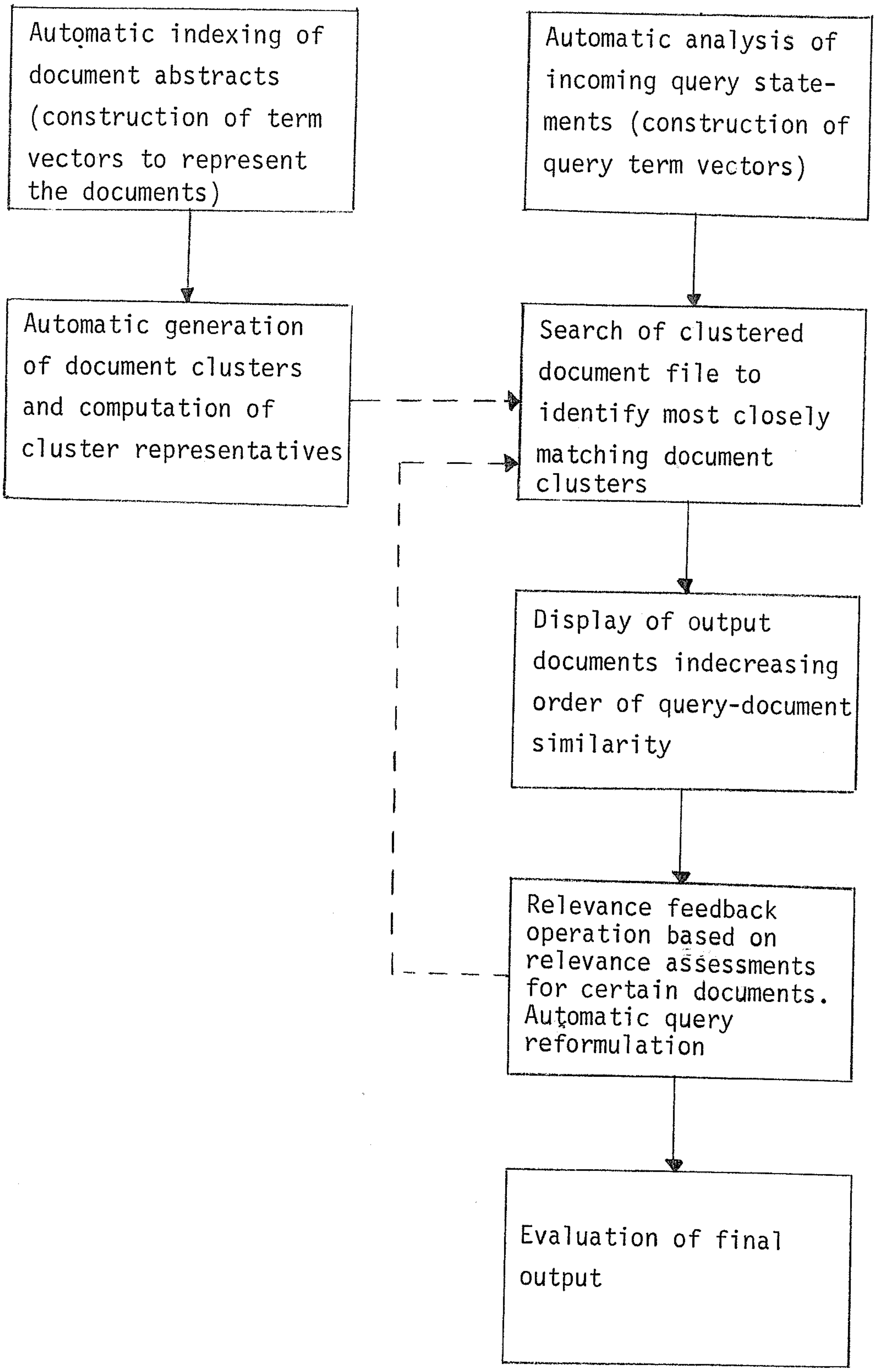

Automatic indexing of document abstracts (construction of term vectors to represent the documents)
Automatic analysis of incoming query statements (construction of query term vectors)
Automatic generation of document clusters and computation of cluster representatives
Search of clustered document file to identify most closely matching document clusters
Display of output documents indecreasing order of query-document similarity
Relevance feedback operation based on relevance assessments for certain documents. Automatic query reformulation
Evaluation of final output

Figure 2.7 - Simplified SMART system flowchart (Adapted from reference 92)

3. CLUSTERING ALGORITHMS

In the context of information retrieval, the term cluster indicates a homogeneous group of objects (documents) such that the objects within a group are more strongly associated with each other than those in different groups. The generation of the clusters is called clustering.

The clustering has very large application areas. A nonexhaustive list will contain the following: life sciences (biology, botany, zoology, etc.); medical sciences (psychiatry, pathology, clinical diagnostic); the behavioral and social sciences; the earth sciences; some engineering sciences (pattern recognition, artificial intelligence, etc.); the information policy, and decision sciences (information retrieval, political sciences, economics, etc.) |2|. Possible uses of the generated clusters are the following (1) finding a true typology, (2) model fitting, (3) prediction based on groups, (4) hypothesis testing, (5) data exploration (6) hypothesis generation and (7) data reduction |36|.

At first sight one may think that an expert of a field will be able to judge the clustering of the observations at hand. Then it might be practical to enumarate all possibilities and simply choose the one which looks the best. However, the problem is not that simple. The number of ways clustering n observations into m nonempty subsets is a Stirling number of second type and given as follows |34, p.162|

$$S_n^{(m)} = \frac{1}{m!} \sum_{k=1}^{k=m} (-1)^{m-k} \binom{m}{k} k^n$$

If n >> m, the last term is more significiant. For n=19 and m=3, the number of possible subsets is

$$S_{19}^{(3)} = 1.98 \times 10^8$$

This is an unbelievably large number and it shows that all cluster possibilities cannot be described with an expression having length bounded by a polynomial function of the input length.

It is very expectable that any given set of data may result with many different number of meaningful clusters, each emphasizing a different feature of the data. It is also possible that the set may not contain any cluster, i.e. the objects are unique by themselves. Similarly the data set may contain only one cluster. In cluster analysis these two possibilities are usually overlooked |2|.

For clustering or clustering analysis different terms are used. Numerical toxonomy is used among life scientists. Social scientist usually prefer "typology". In pattern recognition and cybernetics the terms "learning without teacher" and "unsupervised learing" are usually related to cluster analysis |2,35|. Some of the other words used for this purpose are taximetrics, taxonorics, morphometrics, botryology, nosology, nosographly, and systematics |42|.

The remainder of this chapter is reviewed in the following: Firstly a classification for the clustering algorithms is given. This is followed by the evaluation of clustering algorithms and the process of generating cluster representatives is briefly reviewed. The use of clustering in information retrieval is emphasized once more and it is made clearer by a brief explanation. The remainder of the chapter deals with the illustration of some new clustering concepts which are published and got considerable recognition in the IR community |12,13,14|. These concepts are used in two new clustering algorithms. The concepts introduced also facilitate the cluster maintenance in dynamic collection environments. The complexity of the algorithms is also illustrated. Lastly, the superior characteristics of the algorithms and the algorithm concepts are emphasized and presented.

3.1. CLASSIFICATION OF CLUSTERING ALGORITHMS

In clustering, firstly the documents (or objects in general) are represented by vectors with a length of n, where n indicates the number of terms (attributes) to represent a document. An ith document d_i will correspond to the vector $d_i=(t_{i1},t_{i2},...,t_{in})$, where each term t_{ij} indicates the weight or existence of the jth term in the ith document. For example, if all documents are described by five terms and if a given document is described by the 1'st, 3'rd and the 4'th terms, then the corresponding binary document description vector will be (1,0,1,1,0). A 1(0) at a term position indicates the existence (nonexistence) of the corresponding term for the document. In the weighted approach, the higher (lesser) weight for a term indicates the higher (lesser) importance of the term in the document. By this approach a collection of m documents will be mapped into a matrix, say D, of $m \times n$ size.

The clustering process can be abstracted by an ordered tuple (D, C, A) |38|. Where,

D : indicates the document (or object) collection to be clustered;

C : clusters to be formed, which is a set of sets. Each member of C, C_i, is a nonempty set which contains "associated" documents;

A : indicates the method to be used for clustering the objects. That is, the concept of "association" is materialized according to A.

If the cardinality of a cluster is $|C_i| = 1$, then the cluster is called a singleton. The intersection of two different clusters, $C_i \cap C_j$ where $i \neq j$, can be null. if so the clustering scheme (algorithm) is called a partitioning type. If $C_i \cap C_j \neq \phi$ for at least one i,j ($i \neq j$), then clustering algorithm is an overlapping type.

The clustering algorithms can be classified according to many aspects |2,20,86|. One may divide the clustering algorithms into two: hierarchical algorithms, and iterative algorithms. In hierarchical clustering, as its name implies, there is a hierarchy of clusters. This means that a cluster may contain lower level cluster(s). In this approach, at the lowest level of hierarchy there are documents. At one level higher, there are document clusters. At the upper levels, there are cluster of clusters, or super clusters. In such a case, the above clustering model should be modified such that, C is the nonempty sets of documents at the first level of hierarchy, them at the higher levels there are nonempty sets of clusters.

The hierarchical algorithms utilize similarities between documents. For this purpose the D matrix is mapped into a m x m symmetric document similarity matrix by a chosen mapping process. The (i,j)th element of this matrix resperesents the degree of association between the ith and jth documents in terms of their indexing terms (attributes). A comprehensive list of similarity (association) measures can be found in |2, Chapter 4-5|. The most commonly used similarity measures of the literature are given in the following :

(i) Dice's coefficient : $2 \dfrac{|X \cap Y|}{|X| + |Y|}$

(ii) Jaccard's coefficient : $\dfrac{|X \cap Y|}{|X \cup Y|}$

(iii) Cosine coefficient : $\dfrac{|X \cap Y|}{|X|^{1/2} * |Y|^{1/2}}$

(iv) Overlap coefficient : $\dfrac{|X \cap Y|}{\min(|X|, |Y|)}$

Where, in the above formulas, X and Y indicate two distinct document vectors. |•| indicates the counting operator. In these formulas it is assumed that the document vectors are binary. However, it is easy to generalize them for weighted document representations |2|.

In hierarchical clustering, concepts such as "single link", "average link", and "maximal complete graph" are used |79,86,89|. In single link, each document is expected to be linked to at least one of the other members of its cluster. In the average link, the number of links of a single element to the other members of the cluster is expected to be at least a minimum number. In the maximal complete subgraph, all members of a cluster are linked to all the other members of the cluster. In this clustering algorithm, the similarity matrix is used as follows: A similarity threshold of T_1 is specified and for any similarity matrix entry (say for d_i and d_j) greater than T_1 the documents i and j are assumed to be linked |38|. This test can be repeated for other threshold values less than T_1 until a small enough T_k is reached where all the documents will be assumed to be connected to each other. The iteration process from T_2 to T_k would show the hierarchical connections among the documents. The end product of this porcess, hierarchical clustering tree, is called a dendrogram |106|.

In the iterative algorithms, the main idea is to use the descriptions of documents. For this purpose there are several alternative approaches |2,42,86,89|. One typical method starts with the assignment of documents into existing initial clusters. A document to be clustered is compared with the available clusters and if it is found to be similar to a cluster, then it is assigned to that cluster. The centroid of the cluster is modified accordingly. After the assignment of all document to the clusters a stopping criterion is tested. If it is satisfied, then clustering process stops. Otherwise the same assignment process is performed once more. This clustering process can be performed for a fixed number of clusters |13|. The number of clusters can also vary according to the general behavior of the clustering process |85, Chap.11|. Notice that one may apply the same algorithm to the representatives of clusters, centroids, to form higher level clusters. The selection of the initial clusters (initiators or seeds) is important, since they affect the entire clustering process |89|.

There are also some clustering algorithms which are one pass. They may be considered a special case of iterative algorithms. A typical one pass clustering algorithm considers the documents one at a time. The first item is considered as a cluster, and the next one is compared with it to see if the latter is sufficiently similar to the previous one to join its cluster. If not, a new cluster is generated. The assignment of the other objects is performed in the same manner |89|.

In any type of the clustering algorithms, the overlapping of clusters may or may not be allowed. It should be noticed that overlapping will increase the storage space needed to represent a cluster, since a document's description vector will contribute to more than one cluster's centroid. As an advantage, however, recall of the text retrieval system will be increased.

In the above discussion it is assumed that the document descriptions are made by using the term vectors. The description of the documents can also be made in other ways, such as citations. In such an approach a document will be described by the documents which are referenced by it |92|. For example, there are some efforts for this approach in ISI (Institute for Scientific Information) in the USA |44, p.290|. The exploitation of the user queries for document clustering is an another approach for clustering |81,85 Chap.13|.

3.1.1. Evaluation of the Clustering Algorithms

For the evaluation of the clustering algorithms usually the following factors are considered |89,106|:

a. The clusters produced are unlikely to change when further documents are added;

b. The algorithm is immune to the effects of small errors made in the description of the documents (property (a) and (b) show the stability of an algorithm;

c. The algorithm is independent of the initial ordering of the documents, i.e., whatever the sequence of the documents the same clusters will be generated.

d. The clusters produced are "well-formed", i.e., from a collection either one classification or at least one of a small set of compatible classifications will be produced.

An additional criterion in selecting an algorithm is the efficiency of both the computer time and memory space.

The hierarchical clustering algorithms are usually considered as theoretically more attractive, since they usually satisfy the properties of the good clustering algorithms. For example, they are not affected by the small errors made in the description of documents; the clustering pattern generated is independent of the initial ordering of the documents; they are also "well formed" since the same thresholds are used to define the links among the documents. However, they are not very effective in terms of computation. For example, the computation of the similarity matrix is of order m^2. Where, this does not include the actual clustering process. However, there are some attempts to resolve these difficulties |25,40|.

The iterative methods are usually less satisfying, since they usually do not satisfy the requirements of the good clustering algorithms. For example, they usually do not produce "well formed" clusters; they may depend on the initial ordering of the documents and the goodness of the choice of the documents which will attract the other documents to form the clusters. The research for the evaluation of clustering algorithms is not ended yet and no one clustering algorithm is chosen as the best one and a unique panacea for all problems,

3.1.2. Centroid Generation for the Clusters

The use of clusters is materialized by a cluster representative, generally known as centroid, or profile. The generation of the cluster centroids is a critical task, since (1) it should allow the construction of the hierarchical cluster if they are used for hierarchical clustering; (2) it should match with the query processing strategy. If one considers the hierarchical clustering, for example, as one goes up the hierarchy the number of nonzero entries in a centroid will be less and less and the common terms would be emphasized. In such a case if a top down query processing is employed (i.e., if query vectors are first compared with the topmost cluster's centroid and so on) the query vector and centroid comparisons would not lead to the selection of the most appropriate clusters. In other words, the query vectors will have difficulty in identifying the lower level clusters (For the use of centroids in query processing see the next subsection). In the remainder of this section various centroid generation policies will be introduced.

A cluster-i, C_i, is composed from n_{c_i} documents; $C_i=\{d_1,d_2,\ldots,dn_{c_i}\}$ The centroid for C_i is represented by a vector of size n, G_i (G is the first letter of the word "gravitation"). Some of the most commonly used centroid generation policies are the following |86,92,104,106|:

(a) A typical member of the cluster can be chosen as the centroid. If some documents are used as initiators (or seeds) for the construction of the clusters then they can be used for this purpose. If links are used, then a "maximally linked" document can be chosen as the cluster centroid |105|.

(b) A centroid entry g_{ij} $(j=1,\ldots,n)$ is assigned one if any of the documents of the cluster members contain a 1 at the jth term position.

(c) A centroid entry g_{ij} indicates the document frequency of the corresponding term in the member documents. If a weighted approach is used for the description of documents, then an entry for a term indicates the total weight of the terms in the member documents.

|d| A normalization factor can be introduced in case of (c). Such as, the new g_{ij} (let us show it by g'_{ij}), $g'_{ij}=g_{ij}/n_{c_i}$ or $g'_{ij}=g_{ij}/\|G_i\|$ where $\|G_i\|=(g_{i1}^2+g_{i2}^2+\ldots+g_{in}^2)^{1/2}$. Similarly, a centroid entry can be assigned 1 if $g_{ij}>\log_2 n_{c_i}$, or if the document frequency of a term is greater than a threshold |106, p.102|.

(e) Another approach to prevent considerable variation of the term weights by using the document frequencies is the "rank values". In this case, the terms are sorted according to their frequency in decending order and a centroid entry is taken as the difference between a base value and the rank of the corresponidng term in the sorted list, i.e., $g_{ij} = b - \sum_{d_k \varepsilon c_i} d_{kj}$, where b(base) is given as a parameter.

In centroid generation one should pay attention to its usage, i.e., the centroids should be handy for query processing strategy under use. In other words, they must depend on an estimate of the search requirements of the possible users. For example, short centroid definition will increase the precision of the IR system, while decreasing the recall. The inverse, i.e., longer centroid vectors will result with completely reverse situation (higher recall, lesser precision). In short, the centroid generation is a critical task, and it should not overestimate or underestimate the importance of a term within a cluster or within the entire collection.

3.2. THE USE OF CLUSTERING IN INFORMATION RETRIEVAL SYSTEMS

The use of clustering and clustered documents in information retrieval is introduced in Chapter 2 and it will be emphasized once more. The importance of clustering for document retrieval comes from the "clustering hypothesis" of Van Rijsbergen. This hypothesis states that "closely associated documents tend to be relevant to the same request" |106|. This fact is shown empirically on different experimental retrieval collections |103|.

If one clusters the documents in a reasonable way, then it is obvious that the documents of the same cluster will be much more relevant to each other rather than the documents of other clusters. In query processing, the query vectors will be compared with the cluster centroids (representatives). The clusters of the centroids which show highest similarity will be chosen for further analysis with the query vector (although this is the typical use, cluster based searching shows variations). As it is stated in the clustering hypothesis, closely associated documents (i.e., the documents in the same cluster) will be relevant to the same request. Therefore, it is reasonable to use clustering in information retrieval.

If a hierarchical clustering algorithm is used then query processing can be done either top-down or bottom-up. In the former, the query vector is compared with the centroid of the topmost centroids and the tree traversal is done according to a cluster selection function |104|. In the bottom-up approach, the cluster selection process is done in the reverse direction. Firstly, one or more lowest level clusters are chosen using one or more documents which are known as relevant to user request. This is followed by a cluster tree node selection in the upper direction |26|. In both of the methods, the documents related to the selected node(s) is considered for retrieval. In tree traversing one or more branches can be considered. These approaches are called the narrow and broad search strategies, respectively. The query vector and centroid comparison can be done at once and at the lowest level of clustering |86|. This process is also rewarding, since the number of clusters is much less than the number of documents.

3.3. TWO PARTITIONING TYPE CLUSTERING ALGORITHMS

In this section two new partitioning type clustering algorithms will be introduced |12,12|. In the development of these algorithms the new concepts like coupling coefficient, cluster seed power, distribution of documents among the clusters, and related features are introduced. The presentation of the algorithms will be accompanied by an illustrative example.

For a partitioning type algorithm, different approaches can be used. A generally accepted strategy is to choose a number of seed points (documents) and assign the other documents to these points to form the clusters.

In the clustering algorithms to be presented in this dissertation, a number of documents are selected as the seed points and the rest of the documents are assigned to these seeds according to two different strategies. The selection of these seed points will also be introduced. To contrast the seed selection method used in this dissertation with those of the earlier studies, a list of existing methods used to date is given in the following |2| (where each item in the list corresponds to a different method):

(a) Select the first n_c documents of a collection as clusted seeds, where n_c is the number of clusters to be generated.

(b) Select the seeds randomly from the collection.

(c) Randomly generate the initial cluster seeds.

(d) Divide the collection into partitions, then compute the group centroids as the seed points.

(e) For each term, the number of documents assigned to it (term generality) is calculated. Then for each document, the sum of the term generality of its terms is calculated. The documents with the largest sum are chosen as the initial seed points |85, Chap.12|.

(f) Use an inverted file structure which provides a list of documents per term contained in the documents. In this file, select the documents related with each term as the initial members of the cluster related with the term. The centroids for these clusters can be used as the initial seed points |86,107|.

In the algorithms to be presented, a number of documents are chosen as the cluster seeds and the others are assigned to the clusters initiated by these seeds and a document cannot be assigned to more than one seed. Therefore, both algorithms are of partitioning type. However, it is very easy to change them to allow overlapping among clusters. A model for seed oriented partitioning is given in the following.

3.3.1. A Model for Seed Oriented Partitioning

A partitioned type clustering can be defined by an ordered pair (D,P); where,

D : indicates the documents to be clustered, which is $(d_1, d_2, \ldots, d_m)$.

P : is a particular nonoverlapping partitioning (clustering pattern) of these documents, such that $P = (C_1, C_2, \ldots, C_q)$. C_i $(1 \leq i \leq q)$ indicates the clusters of the partition P. $C_i \cap C_j = \phi$ for $i \neq j$. Each C_i is a collection of ℓ_i documents, $C_i = (d_{i1}, d_{i2}, \ldots, d_{i\ell_i})$, such that $\ell_i \geq 1$ and $\sum_{i=1}^{q} \ell_i = m$.

In a seed oriented partitioning algorithm, there exists a nonempty subset $D_o \subseteq D$, called the seed set, and an equivalence (symmetric, transitive, and reflexive) relation R_M (member of the same cluster relation). This relation has the following propersties:

a. No two distinct seeds are the member of the same cluster, i.e., $d_1 \neq d_2$ and $d_1 R_M d_2 \rightarrow (d_1 \varepsilon D-D_0) | (d_2 \varepsilon D-D_0)$. Where, - and | indicate the set difference and or operators, respectively.

b. For each of the documents which is a member of the ordinary document set, $D - D_0$, $\exists$ a seed document which is the member of the same cluster. Hence, $\forall\, d' \varepsilon D-D_0 \rightarrow \exists\, d \varepsilon D_0$, $d\ R_M\ d'$.

Corollary : For any ordinary document d', there exists exactly one seed document, d_1, satisfying $d' R_M d_1$.

It is stated in (b) that d' has at least one seed d_1 satisfying $d' R_M d_1$. Let us have two seeds d_1 and d_2 ($d_1 \varepsilon D_0$, $d_2 \varepsilon D_0$). To test the above case, we will have $d' R_M d_1$ and $d' R_M d_2$. Since R_M is an equivalence relation, $d' R_M d_1$, $d' R_M d_2 \rightarrow d_1 R_M d_2$. However, this contradicts (a) |13|.

3.3.2. Concepts of the Algorithms

The algorithms introduced in this dissertation use a binary document-by-term matrix, D. Each entry of D matrix, d_{ij}, indicates the existence ($d_{ij} = 1$) or nonexistence ($d_{ij} = 0$) of term-j within document-i. The concepts are also valid for a weighted D matrix with very minor alterations. This will be emphasized later.

There are two basic properties of the D matrix

a. $\sum_{j=1}^{n} d_{ij} \geq 1$, $i=1,\ldots,m$ (each document has at least one term),

b. $\sum_{i=1}^{m} d_{ij} \geq 1$, $j=1,\ldots,n$ (each term is assigned to at least one document).

Where as .it is introduced previously, m and n are, respectively, the number of documents and the number of terms used for the description of the documents.

The concepts of the algorithms will be accompanied with an illustration. The D matrix to be used for this purpose is taken from |86, p.351|. The collection contains 7 documents and each document is described by 8 terms.

$$D = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

3.3.2.1. Cover Coefficient Matrix: C

The matrices S and S', having respectively, the elements S_{ij} and S'_{ij} can be defined as follows:

$$S_{ij} = d_{ij} / \left(\sum_{k=1}^{n} d_{ik} \right) \quad \text{for} \quad i=1,\ldots,m, \quad j=1,\ldots,n$$

$$S'_{ij} = d_{ij} / \left(\sum_{k=1}^{m} d_{kj} \right) \quad \text{for} \quad i=1,\ldots,m, \quad j=1,\ldots,n$$

where,

S_{ij} = the significance of term-j (t_j) for document-i (d_i);

S'_{ij} = the significance of document-i (d_i) for term-j (t_j).

The entry $S_{ij}(S'_{ij})$ is obtained by dividing the value of d_{ij} by row-i (column-j) sum in the D matrix. In case of the S matrix, as the value of row-i sum decreases (increases), the significance of the corresponding term-j increases (decreases) for document-i. Similarly, for the S' matrix, as the value of the column-j sum decreases (increases), the significance of the corresponding document-i increases (decreases) for term-j.

From the definition of matrices S and S', the row sum of S and the column sum of S' matrices are equal to 1. Additionally, the nonzero row (column) entries of S(S') matrix are identical to each other.

In order to have the information kept by these S and S' matrices in the one single matrix, one can enumarate all the allowable cross multiplications among matrices S, S^T and S', S'^T. Note that S^T and S'^T indicate the transpose operation over matrix S and S', respectively. The matrices generated by the product SS'^T, $S'S^T$ are called C and C^T, respectively. Similarly, the products S'^TS, S^TS' generate the matrices C', C'^T. The size of C is m x m, and that of C' is n x n.

Each entry of $C=SS'^T$, c_{ij} (i=1,...,m, j=1,...,m), is expressed as follows:

$$c_{ij} = \sum_{k=1}^{n} S_{ik}S'^T_{kj} = \sum_{k=1}^{n} \text{(significance of } t_k \text{ in } d_i)*\text{(significance of } d_j \text{ for } t_k)$$

Similarly each entry of $C' = S'^TS$, c'_{ij} (i=1,...,n, j=1,...,n) is expressed in the following way:

$$c'_{ij} = \sum_{k=1}^{m} S'^{T}_{ik} S_{kj} = \sum_{k=1}^{m} (\text{significance of } d_i \text{ for } t_k) * (\text{significance of } t_k \text{ in } d_i)$$

Each entry of C indicates a covering coefficient among the documents

$$c_{ij} = \begin{cases} \text{to what extent document-i is covered by document-j,} \\ \text{for } i \neq j. \\ \\ \text{uniqueness of document-i for } i=j \\ \text{(extent with which } d_i \text{ is covered by itself);} \end{cases}$$

Similarly each entry of C' indicates a covering coefficient among the terms used for the description of the documents. From this point on, the matrix C will be referred to as the cover coefficient matrix (of documents).

The discussion to be given in the following section is valid both C and C' matrices. Since our consideration is the clustering of the documents, the discussion will be made in terms of the C matrix.

3.3.2.2. Properties of the Cover Coefficient Matrix

The coverage information held by C is a weight information. As stated in the previous section, its entries (c_{ij}, $i=1,\dots,m$, $j=1,\dots,m$) indicate the extent of coverage of document-i by document-j. The following properties hold for the C matrix.

a) $0 \leq c_{ij} \leq 1$, and $c_{ii} > 0$;

b) For all i, $\sum_{j=1}^{m} c_{ij}=1$ which implies that $\sum_{i=1}^{m} \sum_{j=1}^{m} c_{ij}=m_i$

c) For all i and j, $c_{ii} \geq c_{ij}$. However, if document-i does not have a common term with the rest of the documents, then $c_{ij}=0$ for $i \neq j$ and $c_{ii}=1$.

d) If $c_{ij}=0$ then $c_{ji}=0$ and similarly, if $c_{ij} > 0$ then $c_{ji} > 0$ but in general $c_{ij} \neq c_{ji}$

The properties may be described as follows: Property (a) states that a document-i may or may not be covered by document-j. However, it is always covered by itself. Property (b) states that for any document-i, the row sum will be equal to 1. The sum of all c_{ij}'s will give the total number of documents. The property (c) states that a document is mostly covered by itself, but there might be some other documents which cover the document as much as itself. Lastly, the property (d) states that, coverage among documents is mutual. However, the extent of coverage between two documents might not be identical when coverage is greater than 0. If a document, d_i is unique, i.e., if the terms used in d_i are not used by any other document, then $c_{ii}=1$. If d_i contains some common terms with the other documents, then the value of c_{ii} will be less than 1.

If we consider the C matrix as a relation R_c, called the coverage relation, it is a reflexive relation as shown by:

$$d_i \; R_c \; d_i \qquad \text{for all } i=1,\ldots,m$$

and also it is a symmetric relation as shown by:

$$d_i \; R_c \; d_j \rightarrow d_j \; R_c \; d_i \quad \text{for } i=1,\ldots,m,\ j=1,\ldots,m$$

The proofs of the properties (a) through (d) will follow after the following definitions of the variables to be used.

$\alpha_i = 1 / (\sum_{j=1}^{n} d_{ij})$, inverse of the sum of row-i;

$\beta_j = 1 / (\sum_{i=1}^{m} d_{ij})$, inverse of the sum of column-j;

$c_{ij} = \sum_{k=1}^{n} \alpha_i d_{ik} \beta_k d_{jk}$, general for any element of C matrix.

Proof of Property-a:

The left part of the inequality is obvious according to the general c_{ij} formula and the right part of the inequality is implied by the definition of C elements which is given in terms of significance. If a document has a set of terms and if these terms are not shared by any other document, then the diagonal entry of the C matrix corresponding to this document will be 1. $c_{ii} > 0$, this is obvious from the properties of the D matrix and the general formula for the C matrix.

Proof of Property-b:

$$\sum_{j=1}^{m} c_{ij} = c_{i1} + c_{i2} + \dots . 1\ c_{im} = 1$$

$$c_{i1} = \sum_{k=1}^{n} \alpha_i \beta_k d_{ik} d_{1k}$$

$$c_{i2} = \sum_{k=1}^{n} \alpha_i \beta_k d_{ik} d_{2k}$$

$$\vdots$$

$$+ \quad \underline{c_{im} = \sum_{k=1}^{n} \alpha_i \beta_k d_{ik} d_{mk}}$$

$$= \alpha_i \left[\sum_{k=1}^{n} \beta_k d_{ik} d_{1k} + \sum_{k=1}^{n} \beta_k d_{ik} d_{2k} + \ldots + \sum_{k=1}^{n} \beta_k d_{ik} d_{mk} \right]$$

$$\begin{aligned} = \alpha_i \Big[& (\beta_1 d_{i1} d_{11} + \beta_2 d_{i2} d_{12} + \ldots + \beta_n d_{in} d_{1n}) + \\ & (\beta_1 d_{i1} d_{21} + \beta_2 d_{i2} d_{22} + \ldots + \beta_n d_{in} d_{2n}) + \\ & \quad \ldots\ldots\ldots \\ & (\beta_1 d_{i1} d_{m1} + \beta_2 d_{i2} d_{m2} + \ldots + \beta_n d_{in} d_{mn}) \Big] \end{aligned}$$

$$\begin{aligned} = \alpha_i \Big[& \beta_1 d_{i1} (d_{11} + d_{21} + \ldots + d_{m1}) + \\ & \beta_2 d_{i2} (d_{12} + d_{22} + \ldots + d_{m2}) + \\ & \quad \ldots\ldots\ldots \\ & \beta_n d_{in} (d_{1n} + d_{2n} + \ldots + d_{mn}) \Big] \end{aligned}$$

where α_i and β_i are :

$$\alpha_i = 1 \; / \; (d_{i1} + d_{i2} + \ldots + d_{in})$$

$$\beta_i = 1 \; / \; (d_{1i} + d_{2i} + \ldots + d_{mi})$$

Therefore, each β term cancels the long multiplication term. After this simplification, we are left with :

$$\sum_{j=1}^{n} c_{ij} = \alpha_i \left[d_{i1} + d_{i2} + \ldots + d_{in} \right] = 1.$$

Proof of Propert-c:

$$c_{ii} = \sum_{k=1}^{n} \alpha_i d_{ik} \beta_k d_{ik} = \sum_{k=1}^{n} \alpha_i \beta_k d_{ik}$$

In the general formula of c_{ij}, in addition to the $\alpha_i \beta_k d_{ik}$ term, we have the additional d_{jk} term. Therefore, c_{ij} will be equal to c_{ii} if, and only if, $d_{jk} = 1$ for all $k = 1,\ldots,n$. However, from the definition of the D matrix this equality will not always hold, therefore $c_{ii} \geq c_{ij}$. The case with $c_{ii} = 1$ is obvious.

Proof of property-d:

The general formulas for the terms c_{ij} and c_{ji} are as follows :

$$c_{ij} = \sum_{k=1}^{n} \alpha_i d_{ik} \beta_k d_{jk}$$

$$c_{ji} = \sum_{k=1}^{n} \alpha_j d_{jk} \beta_k d_{ik}$$

From the definition of the D matrix we know that α_i and β_i are always greater than zero for $i=1,\ldots,m$ and $j=1,\ldots,n$, respectively. That is, in order to have a nonzero value for c_{ij} and c_{ji} we must have the product $d_{ik} d_{jk}$ (or equivalently $d_{jk} d_{ik}$) greater than zero. Since both terms sum over the same product, if one becomes zero, the other will also be zero. The second part of the proof, i.e., if $c_{ij} > 0$ then $c_{ji} > 0$ is obvious from the general formula of the c terms.

The cover coefficient matrix, C, corresponding to the D matrix shown earlier is given in the following (the row sums might not be exactly 1 because of rounding). The entire C matrix need not to be constructed for the execution of the algorithms. However, it is fully given for illustration purposes.

$$C = \begin{bmatrix} 0.29 & 0.29 & 0.07 & 0.29 & 0.07 & 0.00 & 0.00 \\ 0.22 & 0.47 & 0.05 & 0.22 & 0.05 & 0.00 & 0.00 \\ 0.07 & 0.07 & 0.51 & 0.07 & 0.18 & 0.00 & 0.11 \\ 0.22 & 0.22 & 0.05 & 0.34 & 0.05 & 0.13 & 0.00 \\ 0.10 & 0.10 & 0.27 & 0.10 & 0.27 & 0.00 & 0.17 \\ 0.00 & 0.00 & 0.00 & 0.50 & 0.00 & 0.50 & 0.00 \\ 0.00 & 0.00 & 0.17 & 0.00 & 0.17 & 0.00 & 0.67 \end{bmatrix}$$

where the values of and for the D matrix are as follows :

$\alpha_1 = 1/3$, $\alpha_2 = 1/4$, $\alpha_3 = 1/3$, $\alpha_4 = 1/4$

$\alpha_5 = 1/2$, $\alpha_6 = 1$, $\alpha_7 = 1/2$.

$\beta_1 = 1/5$, $\beta_2 = 1/3$, $\beta_3 = 1/2$ $\beta_4 = 1$,

$\beta_5 = 1/3$, $\beta_6 = 1$, $\beta_7 = 1$, $\beta_8 = 1/3$.

3.3.2.3. Decoupling (Coupling) Coefficient : δ_i (Ψ_i)

It should be noted that, if a document d_i is unique, i.e., if the terms used by d_i are not used by any other document, then $c_{ii} = 1$. If d_i contains common terms with the other documents, then $c_{ii} < 1$. By the aid of this fact, the following new concepts are introduced.

$$\delta_i = c_{ii}$$

$$\Psi_i = \sum_{j=1}^{m} c_{ij} = 1 - \delta_i, \qquad \text{where } j \neq i$$

$$\delta = \text{trace }(C)/m = \sum_{i=1}^{m} \delta_i/m$$

$$\Psi = (\sum_{i=1}^{m} \sum_{j=1}^{m} c_{ij}) / m = 1 - \delta \qquad \text{where } i \neq j$$

where δ_i indicates the decoupling (uniqueness) coefficient of document d_i. If d_i has less common terms with the other documents, then δ_i will be higher. The reverse situation, i.e., more common terms with the other documents will cause a decrease in δ_i. Ψ_i is the coupling coefficient of document d_i with the other documents. It is obvious that $\delta_i+\Psi_i=1$, and $0<\delta_i\leq 1$, $0\leq\Psi_i<1$. δ and Ψ, respectively indicate the overall decoupling and coupling coefficient of the document collection. Again $\delta+\Psi=1$, and $0<\delta\leq 1$, $0\leq\Psi<1$.

For a document collection, if decoupling of documents is high, it would mean that the documents are not very much related to each other. In other words, they deal with unrelated subjects. As an extreme case, if $\delta=1$, it would indicate that there is no term common among the documents. If coupling of the collection, Ψ, is very high that would imply that the collection deals with the same subject areas having many common terms.

These concepts can be observed from the example C matrix. For example, the decoupling coefficients for document-1 and document-2 (i.e., δ_1 and δ_2) are 0.29 and 0.47 respectively. Similarly the coupling coefficients of these documents (i.e., Ψ_1 and Ψ_2) are (1-0.29=) 0.71 and (1-0.47=) 0.53. The overall decoupling coefficient of the documents is $\delta = \sum_{i=1}^{7} \delta_i/7 = 0.436$. The overall coupling coefficient of the documents becomes $\Psi=0.564$.

3.3.2.4. Number of Clusters within a collection : n_c

The concept of coupling/decoupling of documents can be used to group the document collection into clusters. It is hypothesized that the number of clusters, n_c, within a document collection, would be the following:

$$n_c = (\text{decoupling coefficient of the collection})*(\text{number of documents})$$

$$n_c = \delta * m = \left(\sum_{i=1}^{m} \delta_i / m\right) * m = \sum_{i=1}^{m} \delta_i = \text{trace }(C)$$

If one thinks about the formula for n_c, it makes sense. Since it is also the summation of the uniqueness coefficients for the documents. For example, if we have a collection with all unique documents then the decoupling coefficients for all them will be 1. Then as a logical result, we will have $n_c = m$ number of clusters each holding one document. In other words, each document by itself is a cluster (singleton). A decrease in the decoupling coefficient of the documents, which is an indication for more relatedness among the documents, will also decrease the number of clusters.

If n_c is fractional, then n_c can be taken as $n_c = \lceil n_c \rceil$. In the next section, it will be proved that the number of clusters implied by C and C' matrices are equal to each other. This means that $n_c \leq \min(m,n)$. However, $n_c = \min(m,n)$ does not imply that m=n.

It is obvious from the above n_c expression that, if all documents are distinct, then there will be m different clusters with one document per cluster. Using the definition of n_c, the average number of documents within a cluster, d_c, would be:

$$d_c = m/n_c = m/(\delta * m) = 1/\delta.$$

For our example,

$$n_c = \delta * m = 0.436 * 7 = 3.05 \text{ and}$$

$$d_c = 1/\delta = 1/0.436 = 2.3$$

The relationship between Ψ, δ, and d_c is depicted in Figure 3.1. In Figure 3.1.a, the straight line shows the relationship between Ψ and $\delta(\Psi+\delta=1)$ and the curved line shows the relationship between δ and $d_c(d_c=1/\delta)$. In Figure 3.1.b the relationship between d_c and δ is shown in the logarithmic scale-according to this relationship, for example when $\delta=0.1$ (log.0.1=-1) d_c is equal to 10 (or log10=1), etc.

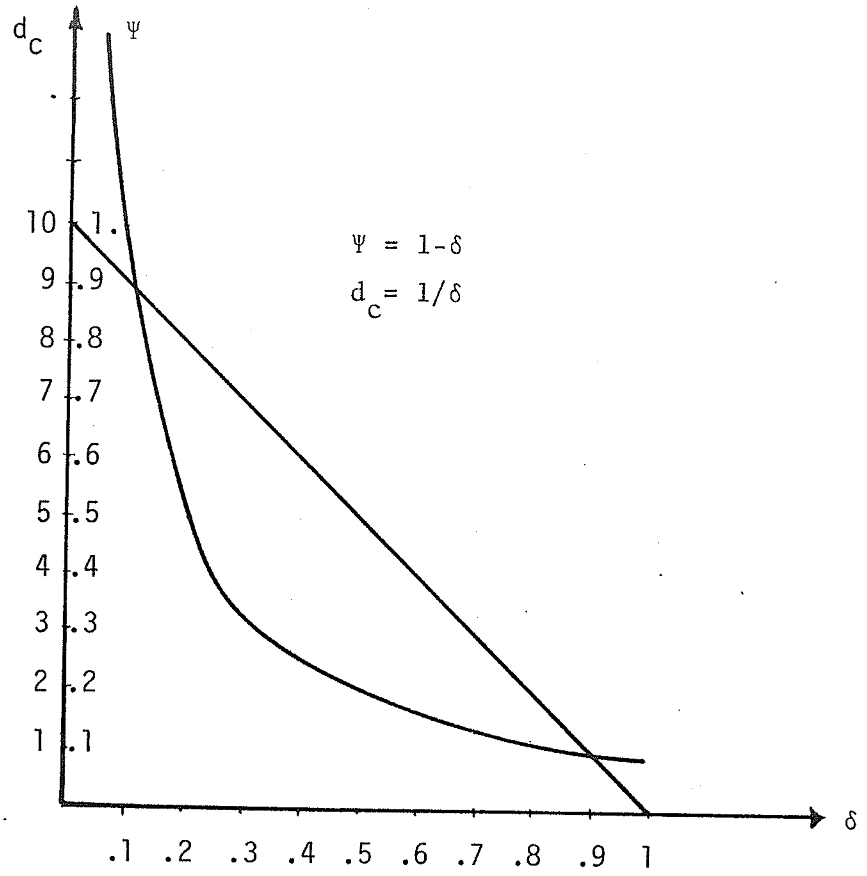
3.3.2.5. The Relationship Between C and C' Matrices

The matrix C' constructed for the clustering of terms will produce n_c' number of clusters, which is:

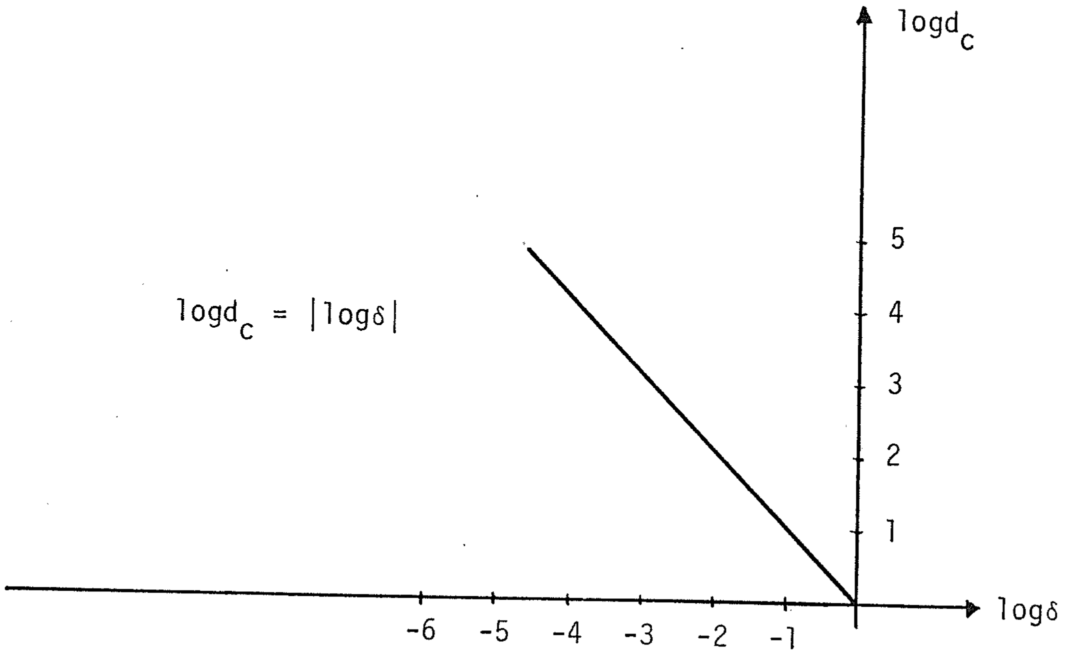
$$n_c' = (\text{decoupling coefficient of the terms})*(\text{number of terms})$$

$$n_c' = \delta' * n = \left(\sum_{i=1}^{n} \delta_i'/n\right) * n = \sum_{i=1}^{n} \delta_i' = \text{trace } (C')$$

It can be shown that $n_c=n_c'$. That is, the number of clusters, implied by the document by term matrix, for documents and terms are equal to each other. Theoretically, this implies that $n_c(=n_c')\leq\min(m,n)$.



Şekil 3.1.a - The relationship between δ and Ψ, d_c. $\Psi = 1 - \delta$, $d_c = 1/\delta$ $(0 \leq \delta \leq 1)$



Şekil 3.1.b - The relationship between δ and d_c in logarithmic scale, $\log d_c = |\log \delta|$ $(0 < \delta \leq 1)$

Proof of $n_c=n'_c$:

The general formulas for n_c and n'_c are given as follows:

$$n_c = \sum_{i=1}^{m} c_{ii} = \sum_{i=1}^{m} \alpha_i(d_{i1}\beta_1 + d_{i2}\beta_2 + \ldots + d_{in}\beta_n)$$

$$n'_c = \sum_{i=1}^{n} c'_{ii} = \sum_{i=1}^{n} \beta_i(d_{1i}\alpha_1 + d_{2i}\alpha_2 + \ldots + d_{mi}\alpha_m)$$

From the general formulas of n_c and n'_c, it is easy to see that there is one to one correspondence among the individual terms. Actually, both formulas are identical with the summation $\sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij}\alpha_i\beta_j$. This proves that $n_c=n'_c$.

The equality between n_c and n'_c does not imply that the clusters implied by documents and terms have one to one correspondence. However, it is worth to be investigated further (See section 8.1 for further discussion).

The equality $n_c=n'_c$, i.e., $\delta*m=\delta'*n$ also implies the following:

$\delta = (n/m) * \delta'$,

$\delta' = (m/n) * \delta$.

3.3.2.6. Cluster Seed Power (p_i) and Cluster Seed Selection

The next step for the construction of the algorithms is the selection of some of the documents as cluster seeds. The other documents will be concentrated around these documents (seeds) to form the clusters. The selection of the cluster seeds will depend on a concept which will be referred to as "cluster seed power". This concept will be introduced after some considerations on cluster seed selection process.

A cluster seed should have a large number of documents around itself. If each cluster seed is separated from the other seeds as much as possible, this will decrease overlap among clusters. Furthermore, by this way clusters will be more unique and unrelated with each other.

The cluster seeds might be chosen as follows. Determine the column sums of the C matrix excluding the diagonal entry c_{ii}. The documents which give the maximum sum can be selected as the cluster seeds. There may be false cluster seeds that are to general to be of use. Such documents, as tutorials, might come up as a cluster seed because they cover a lot of subjects. To get rid of this pitfall one may use the product of the column sum computed corresponding to the ith document and the decoupling coefficient of it, δ_i. Algebraically this is equal to $(\sum_{j=1}^{m} c_{ij}) * \delta_i$, where $i \neq j$. In this product, the column sum and δ_i will contribute to the generality and separation of the clusters,respectively. Although this is a good metric, the calculation of the C terms will require a time which is in the order of $O(m^2n)$, and thus, is very costly.

Some ideas for the selection of cluster seeds are given below :

a) It may appear that the first n_c documents with the highest coupling coefficient value, Ψ_i, should be selected. Although at first sight this seems appropriate, there are some disadvantages of this approach. For example, a document which is described with a very few, but very common terms will, have a high coupling coefficient value.

b) One might select the first n_c documents with the highest decoupling coefficient and which use as many terms as possible. This metric can be calculated as the product of α_i^{-1} and δ_i. Where, α_i^{-1} indicates the number of terms used for the description of the document d_i (α is used in the definition of the C matrix and in the proofs related with its properties). However, this would not be a good metric. Since documents which use a lot of terms (such as tutorials and surveys) will show similarity to the majority of the documents. Therefore, they cannot be good cluster seeds. To provide distinction among clusters, the seeds should be separated from each other as much as possible.

c) After the previous points, it seems that the metric $\delta_i \Psi_i \alpha_i^{-1}$ will be appropriate for selecting document cluster seeds. In this metric, δ_i provides the separation of the cluster seeds (external isolation); Ψ_i provides the generality of the cluster seed, which provides the internal cohesion among the cluster members. This is again a required property of a cluster seed. The term α_i^{-1} provides normalization for the product $\delta_i \Psi_i$. The product $\delta_i \Psi_i$ might be high for a document d_i, but d_i might not be a good cluster seed, since it should also cover a large number of terms in order to provide connection to different documents.

The product $\delta_i \Psi_i \alpha_i^{-1}$ is called the cluster seed power, p_i, of document d_i. The next step for identifying the cluster seeds is sorting of the cluster seed powers in descending order. Afterwards, the first n_c documents corresponding to the first n_c cluster seed power of this sorted list will be chosen as the cluster seeds.

It has been observed in the experiments that the cluster seeds chosen by this method and the expensive method described at the beginning of this section always have 90% or more common terms |13,14|. Therefore, the product $\delta_i \Psi_i \alpha_i^{-1}$ is a good metric for determining the cluster seed power.

Figure 3.2 depicts the relationship between the number of terms assigned to a document and the average decoupling (uniqueness) coefficient, coupling coefficient, and cluster seed power of the documents with that many number of terms for an ideal situation. In that figure a normalized cluster seed power is assumed, where the normalization is made by dividing the corresponding seed power with the observed maximum seed power.

The seed power of each document in the example D matrix, in descending order, is determined as follows (the results are rounded and truncated):

$$p_2 = \delta_2 \Psi_2 \alpha_2^{-1} = 0.47 * (1 - 0.47) * 4 = 0.996$$

$$p_4 = \delta_4 \Psi_4 \alpha_4^{-1} = 0.34 * (1 - 0.34) * 4 = 0.900$$

$$p_3 = \delta_3 \Psi_3 \alpha_3^{-1} = 0.51 * (1 - 0.51) * 3 = 0.750$$

$$p_1 = \delta_1 \Psi_1 \alpha_1^{-1} = 0.29 * (1 - 0.29) * 3 = 0.616$$

$$p_7 = \delta_7 \Psi_7 \alpha_7^{-1} = 0.67 * (1 - 0.67) * 3 = 0.444$$

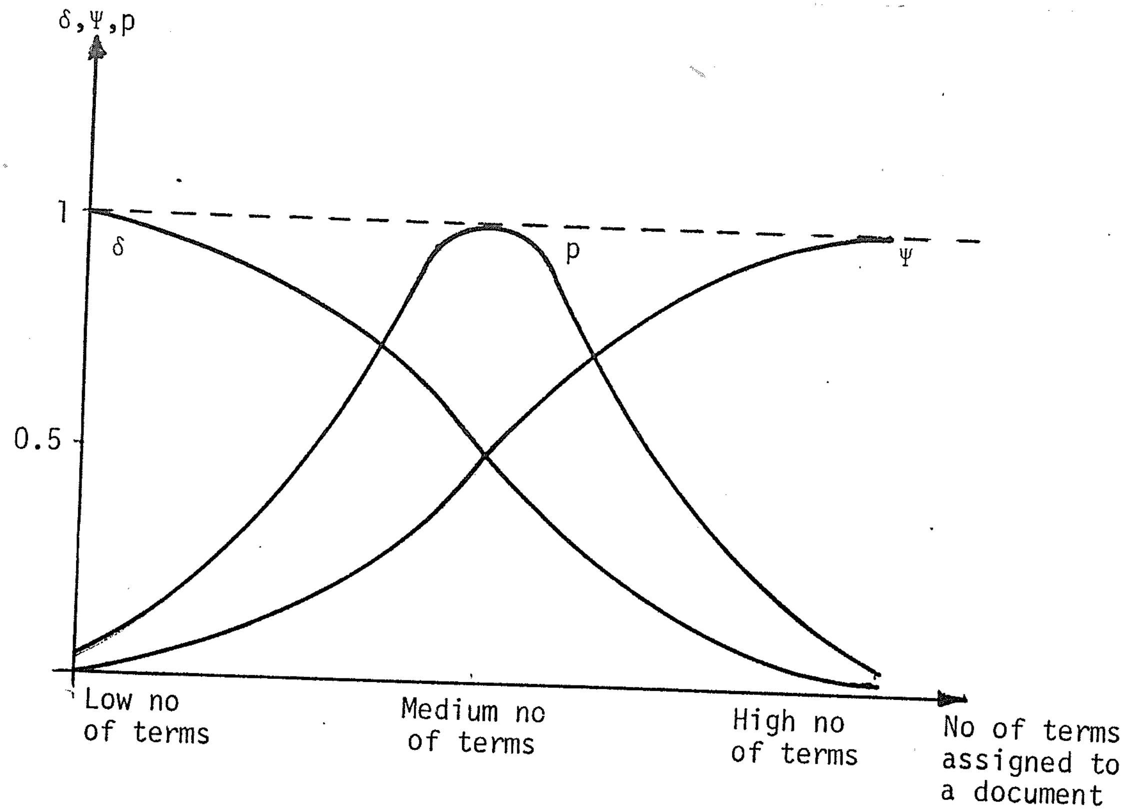


Figure 3.2 - The relationship to be observed between number of terms assigned to a document and document's δ, Ψ, and p for an ideal situation.

$$p_5 = \delta_5 \Psi_5 \alpha_5^{-1} = 0.27 * (1 - 0.27) * 2 = 0.391$$

$$p_6 = \delta_6 \Psi_6 \alpha_6^{-1} = 0.50 * (1 - 0.50) * 1 = 0.250$$

If one calculates the cluster seed powers according to the costly method and lists them in the descending order the following list is obtained :

$$p_4 = (\sum_{j=1}^{7} c_{j4}) * \delta_4 = 1.18 * 0.34 = 0.401 \qquad j \neq 4$$

$$p_2 = (\sum_{j=1}^{7} c_{j2}) * \delta_2 = 0.68 * 0.47 = 0.320 \qquad j \neq 2$$

$$p_3 = (\sum_{j=1}^{7} c_{j3}) * \delta_3 = 0.61 * 0.51 = 0.313 \qquad j \neq 3$$

$$p_7 = (\sum_{j=1}^{7} c_{j7}) * \delta_7 = 0.28 * 0.67 = 0.188 \qquad j \neq 7$$

$$p_1 = (\sum_{j=1}^{7} c_{j1}) * \delta_1 = 0.61 * 0.29 = 0.177 \qquad j \neq 1$$

$$p_5 = (\sum_{j=1}^{7} c_{j5}) * \delta_5 = 0.52 * 0.27 = 0.140 \qquad j \neq 5$$

$$p_6 = (\sum_{j=1}^{7} c_{j6}) * \delta_6 = 0.13 * 0.50 = 0.065 \qquad j \neq 6$$

The theoretically implied number of clusters is three, the first three documents with the highest cluster seed power are selected as the cluster seeds. These are documents 2, 4, and 3. Notice that both cluster seed selection methods provide the same documents as the cluster seeds. However, as it is discussed above, the one which uses the formula $\delta_i \Psi_i \alpha_i^{-1}$ will be used for determining the cluster seed power. This both because of its efficiency and effectiveness.

3.3.2.7. Modifications in the Clustering Concepts for a Weighted D matrix

The properties of the C and C' matrices are given for a binary D matrix. However, they are also valid for a general (weighted) D matrix description (i.e., $0 \leq d_{ij} \leq \infty$ for i=1,...,m, j=1,...,n) with a very minor alteration. All the properties of the C matrix will be valid for the weighted case, the only alteration is in property-c. This property states that $c_{ii} \geq c_{ij}$, however, in the weighted form c_{ii} can be less than the off-diagonal entries of the ith row. This can easily be seen from the general formulas for c_{ii} and c_{ij} where

$$c_{ii} = \sum_{k=1}^{n} \alpha_i \beta_k d_{ik}^2$$

$$c_{ij} = \sum_{k=1}^{n} \alpha_i \beta_k d_{ik} d_{jk}$$

Consider a case such that $d_{jk} > 0$ when $d_{ik} > 0$ and assume that $d_{jk} \geq d_{ik}$ for k=1,...,n and d_{jk} is greater than d_{ik} for at least one given k value. Such a condition will make $c_{ij} > c_{ii}$.

In the case of the weighted D matrix, the definition of the cluster seed power needs some modifications. In the binary D matrix, the seed power for document-i is given as $p_i = \delta_i * \Psi_i * (\sum_{j=1}^{n} d_{ij})$. In the weighted D matrix, the contributions due to the terms i.e. $\sum_{j=1}^{n} d_{ij}$ need to be normalized in a way. Since there might be some over estimation on the weights of some terms in document-i. The following can be used for seed power calculation in the weighted D matrix.

$$p_i = \delta_i * \Psi_i * (\sum_{j=1}^{n} d_{ij} * \delta_j' * \Psi_j')$$

Again in this case each term will contribute to the seed power of document-i. In this contribution the weight of each term of the document is normalized by the product of decoupling and coupling coefficients of the term. By such an approach a term with high weight, but with skewed frequency (i.e., very frequent or very rare terms) will not contribute that much to the summation, i.e., the seed power of the document-i.

3.3.3. The Algorithms

The algorithms to be described here use the same method for cluster seed selection. However, the assignment of the documents to the cluster seeds is different. The first algorithm uses the new "cover coefficient" concept and it is a single-pass algorithm, while the second one uses the conventional notion of "similarity" and it is a multi-pass algorithm. The second algorithm is introduced to show the validity of the new cover coefficient concept, and for this purpose, a set of experiments were performed the results of which will be the topic of the subsequent chapter.

As stated above, both algorithms use the same method for the selection of the cluster seeds. It is obvious that not all of the documents selected as a cluster seed can be used directly as such. Some of the cluster seeds may be false cluster seeds, since there may be nearly identical documents described by the same number of nearly identical terms. The elimination of the false seeds can be done by using the algorithm defined in the next subsection.

3.3.3.1. An Algorithm to Eliminate the False Cluster Seeds

For the elimination of false cluster seeds the following algorithm can be used.

```
a) Take the first $n_c$ documents which give the maximum cluster
   seed power;
   $N_c = n_c$; /* $N_c$ = current number of cluster seeds. */
```

```
b) repat;
      If $N_c < N_c$ then
      do;
         Consider the next $(n_c - N_c)$ documents
         with the maximum cluster seed power
         as the new cluster seeds.
      end;
         Determine the number of equivalence
         classes within this cluster seed
         collection, and set $N_c$ to this number.
   until $N_c = n_c$;
```

The equivalence classes are found by using the relation "equal level", Re. Two seeds, i and j, are related with each other according to the relation Re, if $c_{ii} = c_{jj}$, $c_{ii} = c_{ij}$, $c_{jj} = c_{ji}$. It is obvious that, the relation Re holds the requirements of an equivalence relation. Since it is reflexive, symmetric, and transitive. This relation is illustrated in the following :

a. Re is reflexive, i Re i is trivial.

b. Re is symmetric, since i Re j, and j Re i. i Re j implies that $c_{ii} = c_{jj}$, $c_{ii} = c_{ij}$, $c_{jj} = c_{ji}$; where j Re i implies the same equalities by changing the order of the operands at both sides of the equality relation (=). Hence, Re is symmetric.

c. Re is transitive, since i Re j and j Re k implies that i Re k. i Re j implies $c_{ii} = c_{jj}$, $c_{ii} = c_{ij}$, $c_{jj} = c_{ji}$; j Re k implies $c_{jj} = c_{kk}$, $c_{jj} = c_{jk}$, $c_{kk} = c_{kj}$. Where these equalities also implies that $c_{ii} = c_{kk}$, $c_{ii} = c_{ik}$, $c_{kk} = c_{ki}$. Therefore, Re is transitive.

After showing that Re is an equivalence relation, then it is trivial to show that Re partitions the cluster seeds into equivalence classes |5, p.87-88|. It is obvious that within an equivalence class there might be two or more cluster seeds. Only one of the seeds of an equivalence class is taken as a cluster seed, and the rest of them are considered as false, since all of the others are compatible (or equivalent) with the one that is chosen as the cluster seed (Here it is assumed that the clusters generated by the equivalent seeds will be nearly equivalent to each other).

The above algorithm might be applied as follows. To get rid of the false cluster seeds, it is necessary to compare each cluster seed with the next possible cluster seed. This assumes that the cluster seed powers are sorted in descending order, the first seed is compared with the second seed and so on. It will be enough to compare the cluster seed under consideration with the next (lower) cluster seed, since they will be in consecutive order due to their close similarity. If the seeds (documents) i and j are the members of an equivalence class, then the entries c_{ii}, c_{ij}, c_{jj}, and c_{ji} will be almost identical, i.e., for example $c_{ii} - c_{jj} \leq t$. Where t is a small number chosen as a threshold.

3.3.3.2. The Single-pass Algorithm

In this algorithm, a document is joined to a cluster, if it is maximally covered by the corresponding cluster seed (document). We know that the matrix entry c_{ij} shows how strongly document-i is covered by document-j. Therefore, if j and k are two different cluster seeds and if $c_{ij} > c_{ik}$, then this means that document-i will join the cluster initiated by document-j rather than that of document-k. The single pass algorithm is as follows |13,14|:

```
a) Determine the cluster seeds, which are the documents
   determining the first $n_c$ highest cluster seed powers
   (this step also eliminates the false cluster seeds);
   $i=1$;

b) repeat;
      If document-i is not a cluster seed
      then
        do;
           Find the cluster seed which maximally covers
           document-i. If there is more than one cluster
           seed with this property, then the document
           might be assigned to all of the seeds, if
           overlapping is allowed, otherwise it will
           be assigned to only one according to a predetermined
           decision (such as assigning the document
           to the cluster which has the maximum seed power
           among the candidates);
        end;
        $i = i+1$;
   until   $i > m$;

c) The documents which are not covered by any of the seeds
   might form a separate cluster by themselves (i.e.,
   $\{d_i \varepsilon D - D_o \mid d_j \varepsilon D_o$ and $c_{ij} = 0\}$). Alternatively, for each
   unclustered document, one may determine the maximal
   cover of it and assign the document to the covering
   document's cluster. If an unclustered document has more
   than one maximal cover, then it will be assigned to one
   of them according to a decision as in step |b|. Step |c|
   might be applied iteratively until the size of the set
   of unclustered documents reaches stability.

d) Stop
```

The phrase "maximal cover", used in the algorithm, needs some explanation. The maximal cover of an unclustered document (say d_i) is any of the clustered documents (say d_j), where d_j covers d_i with a cover coefficient value, such that, no other clustered document can cover d_i with this strength.

The reason that step (c) may be repeated is because we may extend the size of the clusters by using the maximal cover concept. As a result, some of the documents which were not covered previously may now be covered by the recently clustered documents (which were clustered by using the cover coefficient concept).

It is hypothesized that, the estimated number of documents in cluster-i, initiated by d_i, is given by n_i as:

$$n_i = (p_i \; / \sum_{k=1}^{n_c} p_k) * m, \qquad i = 1,\ldots,n_c$$

This means that, the number of documents related with a cluster depends on the magnitude of the cluster seed power of the corresponding seed document.

3.3:3.3. The Multi-pass Algorithm

The multi-pass partitioning algorithm to be presented in this section uses the same method shown earlier to find the cluster seeds of the collection. This time, however, the asssignment of documents to cluster seeds will depend on a similarity criterion, rather than cover coefficient. Similarity among documents can be expressed in many different ways. The most commonly used measures are given in Section 3.1.

In this algorithm, the clusters will be optimized by doing the assignment process repetitiously; accordingly, a document is assigned to the cluster which has the maximum "document : cluster seed" similarity value. After the assignment of all documents to the cluster seeds, the centroid of each cluster is computed according to the documents assigned to it. During the first iteration, the cluster seed's description vector, which is simply a document description vector, is taken as the cluster centroid. Each iteration of the algorithm refines both the definition of the centroids and the clusters. The algorithm of this strategy is as follows |9|.

```
a) Determine the cluster seeds as before;

b) repeat ;

      Create the centroid of each cluster according
      to the documents assigned to the cluster;
      i=1.
```

```
        repeat ;

          Find the centroid which shows the maximum
          similarity to document-i. If there is more
          than one centroid with this property, the
          document might be assigned to all of the
          corresponding clusters of the centroids if
          overlapping is allowed, otherwise it will
          be assigned to only one according to a
          predetermined decision, as in the single-
          pass algorithm
          i = i + 1 ;

        until  i > m ;

      until  no document has changed it cluster ;

    c) Treat the unclustered documents as in the single-
       pass algorithm ;

    d) stop ;
```

During the first execution of the outer repeat block, the document vectors of the cluster seeds are taken as the cluster centroids. Furthermore, since the cluster seeds are the initiators, they are taken as the natural members of each initiated cluster. Notice that in order to make the algorithm immune to the ordering of documents, the modification of the cluster centroid is made after the assignment of all the documents to the cluster centroids.

3.3.3.4. Some Refinements on the Clustering Algorithms

The definitions of both of the algorithms presented may lead to ovarlap among clusters, however, this is not allowed. Therefore, if a document has the same maximum coverage or maximum similarity with more than one cluster seed, the document is assigned to only one of these seeds. The documents which are not assigned to any cluster may form (1) a "ragbag" cluster, then in order to be consistent with the definition of the model for seed oriented partitioning, any document of the ragbag cluster might be chosen as the cluster seed; or (2) stay as a singleton.

In the implementation of the second algorithm, it was observed that the termination condition (i.e., until no document has changed its cluster;) is rather strict: This is because, after some number of iterations, a very few number of documents might change their clusters between iterations causing only slight changes in the clusters. For practical purposes, the termination condition can be modified as "until the proportion of the stationary documents reaches $\geq$ .95". The illustration of the algorithms for the example D matrix will be given after introducing the centroid generation policies.

3.3.4. Centroid Generation Policy of the Algorithms

The methodology of centroid generation for the clusters will be very important for the effectiveness of a document retrieval system. Furthermore, it may also influence the clustering process as seen in the second partitioning algorithm. In this section, generation of centroids will be illustrated.

3.3.4.1. Centroid Generation Policy for the Single-pass Algorithm

A centroid for the clustered documents is represented by G_i, where

$$G_i = (g_{i1}, g_{i2}, \ldots, g_{in}), \qquad i = 1,\ldots,n_c$$

Each g_{ij} indicates the existence (nonexistence) of term-j in the centroid-i.

The values of the centroid entries will be based on the following variables :

$f_j^i = \sum_{d_k \varepsilon C_i} d_{kj}$ indicates the frequency of term-j within the document vectors of cluster-i;

$f_{javg} = (\sum_{i=1}^{m} d_{ij}) / n_{c_j}$, where the dividend indicates the document frequency of term-j (i.e., its generality), the divisor indicates the number of clusters containing the documents whose definition vectors contain term-j, i.e., $n_{c_j} = |\{C_e | d_{ij} \neq 0 \text{ and } d_i \varepsilon C_e\}|$, where $|\cdot|$ indicates the cardinality of a set. Therefore, it is the average number of occurrences of term-j within the clusters containing it.

δ_j' is the diagonal entry of the term by term matrix $C' = S^T S'$ for term-j; where, the diagonal entries of C' indicate the uniqueness of term-j.

$\delta' = \text{trace}(C') / n$, overall decoupling of the terms.

A centroid entry g_{ij} is assigned a value of 1 if $f^i_j \delta'_j \geq f_{javg}\delta'$ (this will be referred to as "the state of existence rule of a term within a centroid"), otherwise g_{ij} will be 0. In |12|, f_{javg} is taken as $(\sum_{i=1}^{m} d_{ij})/n_c$, which is the average number of occurrences of a term within a cluster. Since the value of n_c is considerably high, the condition $f^i_j \delta'_j \geq f_{javg}\delta'$ was satisfied readily. Consequently, this resulted in centroids with too many terms (nonzero entries). Therefore, the state of existence rule of a term within a centroid has taken the final form presented above (i.e., f_{javg} is computed using n_{c_j} rather than n_c).

According to the centroid definiton given above, the terms which appear often enough within a cluster and which have noticeable uniqueness will appear in the centroid description.

Proposition : The terms which have a uniqueness value (i.e., the diagonal entries of the C' matrix, $\delta'_i = c'_{ii}$) greater than or equal to the average uniqueness value of the terms ($\delta' = \sum_{i=1}^{n} \delta'/n$) will appear at least once in the centroid vectors.

Proof : Consider the condition $f^i_j \delta'_j \geq f_{javg}\delta'$ that assigns a value 1 to a centroid entry. Take the sum of both sides for all the clusters. Then we are left with $\sum_{i=1}^{m} d_{ij} * \delta'_j \geq \sum_{i=1}^{m} d_{ij} * \delta'$, which leads to $\delta'_j \geq \delta'$. If the left side of the general inequality formula is always less than the right side for all clusters then $\delta'_j \geq \delta'$ cannot hold. This means that, for the inequality $\delta'_j \geq \delta'$ to hold, at least for one of the clusters, the condition for the assignment of term-j to a cluster must hold. This concludes the proof.

Such an approach for centroid generation will emphasize the terms with higher uniqueness values. The condition of existence of a term within a centroid might be modified slightly, so that it can be more strict or relaxed. The modification involves a multiplier τ as $f^i_j \delta'_j \geq \tau f_{javg} \delta'$. After this modification, the terms which satisfy the condition $\delta'_j \geq \tau \delta'$ will appear at least once in the centroid definitions. If $\tau > 1$ ($\tau < 1$) this means that the state of existence rule of a term within a centroid is harder (easier). Accordingly, one may use $\tau > 1$ ($\tau < 1$) if precision (recall) of the IR system is more important for its users.

In this centroid generation policy some of the generated centroids might have all zeros. In such cases, the state of existence rule of a term within such centroids can be changed with the multiplier τ. However, it is expected that such cases will appear very rearly (For example in the experiments, which are illustrated in the next chapter, such a case was never observed).

3.3.4.2. Centroid Generation Policy for the Multi-pass Algorithm

In the implementation of the second algorithm, the definition of the C matrix is used for initiation of the algorithm. More specifically, the seeds are selected by using the cover coefficient concept. However, assignment of the documents to cluster seeds is done using a similarity measure. Since cluster formation is not related with the uniqueness of the terms, a conventional method, the threshold concept, has been utilized in determining the centroid terms.

According to the threshold concept to be used, if there are k documents in a cluster, in order to assign a term to the cluster centroid the term should appear in at least T (a threshold value) of the member documents. The work of Croft has shown that, under certain assumptions, T should be k/2 or slightly less |106, p.102|. In the illustration to be followed in the next subsection, and in the experiments to be described in the next chapter, T is taken as k/3.

3.3.4.3. Evaluation of Cluster Representatives

In a document retrieval environment, the generated centroids are used for query processing. The centroid should be tested experimentally by submitting different queries to the system |106, p.99|. The centroid generation policy should be in accordance with query processing strategy. Furthermore, the centroids generated can also be used for hierarchical clustering.

The following lists various ways of testing the centroid :

a. For all clusters and documents test to see whether the centroid of the cluster, in which the document is assigned, is the most similar centroid to the document. Ideally, this should be true.

b. Determine the number of mismatches among the members of a cluster and its centroid (in the ideal case this should be zero, which might happen if all the documents are clusters -singletons- by themselves). A new concept, the probability of having a mismatch at a term position, m_p, is introduced |13|. The metric m_p is the probability of having a mismatch at a term position between a centroid and a member of the corresponding cluster. Again, the ideal value for m_p is zero. Small value of m_p will imply better centroid generation.

To find the total number of mismatches (say M) between centroids and the corresponding documents, the following is used:

$$M = \sum_{i=1}^{n_c} \left(\sum_{j=1}^{n_{c_i}} \sum_{k=1}^{n} (d_{ij,k} - g_{ik})^2 \right).$$

In this formula, n_c indicates the number of clusters within the collection, n_{c_i} indicates the number of documents within cluster i, n is the number of terms, and $d_{ij,k}$ is the entry of the document vector corresponding to the kth term of the ith cluster's jth document, and g_{ij} is the value of term-j's entry in centroid-i. However, a more practical formula for M might be the following :

$$M = \sum_{i=1}^{n_c} \sum_{j=1}^{n} \text{abs}(f_j^i - n_{c_i} * g_{ij})$$

where "abs" indicates the absolute value function.

After introducing the two concepts. m_c and m_d, it will be easy to find m_p.

$m_c = M/n_c$: the average number of mismatches per cluster

$m_d = m_c/d_c$: the average number of mismatches per document

where d_c was defined as the average number of documents per cluster. After this, m_p is obtained as follows :

$$m_p = m_d / n$$

In the experiments which will be covered in the next chapter, these masures will be used to test the behavior of a given centroid generation methodology.

3.3.4.4. Document Assignment and Centroid Formation for the Single-pass Algorithm

During the assignment process, we should compare cover coefficinet values C(i,j), where i and j are, respectively, the document to be clustered and the cluster seed. For document-1: $C(1,2) = 0.29$, $C(1,4) = 0.29$, $C(1,3) = 0.07$. Document-2 has a higher seed power, therefore document-1 is assigned to its cluster, although $C(1,2)=C(1,4)$. If one proceeds in the same manner, the clusters will be formed as follows: (2,1), (4,6), and (3,5,7). Therefore the number of documents associated with the cluster seeds 2, 3, and 4 are 2, 2, and 3, respectively. The implied number of documents (due to the magnitude of the cluster seed power) associated with the same seeds, respectively, are 2.66, 2.38, and 1.98. In this example, the correlation between the implied and the actual number of documents in a cluster is not good. However, in practice, high degree of correlation is observed between these two entities |14| (see also next chapter).

Centroid generation for the first cluster is given in the following, the centroid generation for the others will be done in the same manner. If one obtains the C' matrix, the $\delta'_i(=c'_{ii})$ values for the terms come out as 0.33, 0.28, 0.63, 0.25, 0.28, 0.50, 0.33, and 0.44 for $i = 1,\ldots,8$, respectively. From this, $\delta' = \sum_{i=1}^{8} \delta_i/8$ is calculated as 0.38. f_j values are obtained from the D matrix as 2, 2, 0, 1, 2, 0, 0, 0 respectively for $j = 1,\ldots,8$. f^1_{javg} values for $j = 1,\ldots,8$, respectively, are 5/3, 3/2, 2, 1, 3/2, 1, 1, 3. A 1 will be assigned to the jth centroid position for cluster-1 if $f^1_j\delta'_j \geq f_{javg}\delta'$ for $j = 1,\ldots,8$. These conditions evaluate to: (for g_{11} through g_{18} respectively) $0.66 > 0.63$, $0.56 < 0.57$, $0 < 0.76$, $0.25 < 0.38$,

$0.56 < 0.57$, $0.0 < 0.38$, $0.0 < 0.38$, $0.0 < 1.14$. The assignment conditions of a term to a centroid is satisfied only for term-1. Therefore, the generated centroid becomes

$$G_1 = (1, 0, 0, 0, 0, 0, 0, 0).$$

The centroids for the seeds 3 and 4, respectively, are the following:

$$G_2 = (0, 0, 1, 0, 0, 0, 0, 0)$$

$$G_3 = (1, 0, 0, 0, 0, 1, 0, 1)$$

The number of mismatches among the member documents and the corresponding centroids are 5, 3, and 4, respectively, for the centroids 1, 2, and 3. Then M becomes $5 + 3 + 4 = 12$. m_c and m_d, accordingly, become (12/3=)4 and (4/2.3=)1.74, respectively. This results in an m_p value of (1.74/8=)0.22.

The similarity coefficients among the documents and the cluster centroids are highest if the centroid belongs to the document's cluster. In this example, the centroid formation behaves rather poorly, since the cluster centroids contain very small number of terms with respect to document vectors. This is because, the average number of occurrences of a term within clusters is very high, which hardens the assignment of a term to a centroid. However, this should not be the case in a real life environment.

3.3.4.5. Document Assignment and Centroid Formation for the Multi-pass Algorithm

This algorithms starts with the same cluster seeds as in the previous example. The document description vectors of the seeds are taken as the centroids. In the first iteration of the algorithm the documents are assigned to the seeds as follows (where in the following, Sm(d,s) indicates the similarity between the document d and the cluster seed s according to the Dice's coefficient):

d = 1 : Sm(1,1) = 6/7, Sm(1,2) = 6/7, Sm(1,3) = 2/6. d_1 will join cluster-1 (notice that $p_1 > p_2$).

d = 2 : is a seed, so it is already assigned.

d = 3 : is a seed, so it is already assigned.

d = 4 : is a seed, so it is already assigned.

d = 5 : Sm(5,1) = 2/6, Sm(5,2) = 2/6, Sm(5,3) = 4/5. d_5 will join cluster-3.

d = 6 : Sm(6,1) = 0, Sm(6,2) = 2/5, Sm(6,3) = 0. d_6 will join cluster-2.

d = 7 : Sm(7,1) = 0, Sm(7;2) = 0, Sm(7,3) = 2/5. d_7 will join cluster-3.

The resulting clusters are: C_1(2,1), C_2=(4,6), and C_3=(3,5,7). The centroids corresponding to these clusters, respectively, are G_1=(1, 1, 0, 1, 1, 0, 0, 0), G_2=(1, 1, 1, 0, 1, 0, 0, 0), and G_3= (1, 0, 0, 0, 0, 1, 1, 1).

In the second and subsequent iterations, all the documents will be compared with all the clusters.

d = 1 : Sm(1,1) = 6/7, Sm(1,2) = 6/7, Sm(1,3) = 2/7. d_1 will stay in the same cluster.

d = 2 : Sm(2,1) = 1, Sm(2,2) = 6/8, Sm(2,3) = 2/8. d_2 will stay in the same cluster.

d = 3 : Sm(3,1) = 2/7, Sm(3,2) = 2/7, Sm(3,3) = 6/7. d_3 will stay in the same cluster.

d = 4 : Sm(4,1) = 6/8, Sm(4,2) = 1, Sm(4,3) = 2/8. d_4 will stay in the same cluster

d = 5 : Sm(5,1) = 2/6, Sm(5,2) = 2/6, Sm(5,3) = 4/6. d_5 will stay in the same cluster.

d = 6 : Sm(6,1) = 0, Sm(6,2) = 2/5, Sm(6,3) = 0. d_6 will stay in the same cluster.

d = 7 : Sm(7,1) = 0, Sm(7,2) = 0, Sm(7,3) = 4/6. d_7 will stay in same cluster.

At the end of the second iteration, since no document changes its cluster, the algorithm terminates. The number of mismatches between the cluster members and the centroids are 1, 3, and 5, for the clusters 1, 2, and 3. The values of M, m_c, m_d, and m_p becomes (1+3+5=)9, (9/3=)3, (3/2.3=)1.30, and (1.30/8=)0.16, respectively.

3.3.5. Complexity Analysis of the Algorithms

In this section complexity analysis of the algorithms will be presented. The complexity of some of the other algorithms can be seen in Table 3.1. The explanation for these algorithms can also be found in |86|.

One of the most credential feature of a clustering algorithm is its order independence. It is observable from Table 3.1 that, the complexity of the algorithms which provides order independence is rather high, i.e., either exponential $(0(k^m))$ or quadratic $(0(m^2))$. After the following complexity analysis of the two clustering algorithms, it will be seen that they are favorably comparable to the complexity of the other clustering algorithms proposed to date.

TABLE 3.1 - Analysis of Some of the Clustering Algorithms

Clustering Method	Clustering Time	Order Independence
Clique finding:		
All cliques	$0(k^m)^*$	Yes
Limited cliques	$0(m)^2$	Yes
Single-link method	$0(m^2)$	Yes
Density test method:		
Worst case	$0(m^2)$	No
Average case	$0(m^2/\log m)$	No
Interchange process:		
Worst case	$0(m^2)$	No
Average case	$0(m/\log m)$	No
One pass method:		
Worst case	$0(m^2)$	No
Average case	$0(m/\log m)$	No

* *k is a constant.*

3.3.5.1. Complexity Analysis of the Single-pass Algorithm

The complexity of the algorithm can be determined from the selection of cluster seeds and the clustering process. In the first step, the cluster seed power of each document is calculated and placed into a sorted order. The complexity of this step would be $O(m*n + m*\log m) \cong O(m*n)$. Where,

$m*n$: corresponds to the complexity of cluster seed power calculations;

$m*\log m$: indicates the complexity of sorting m cluster seeds.

The complexity of the document assignment (clustering) process will be $O(n_c*m*n)$. The significance of each term is as follows:

n_c : indicates that all cluster seeds are considered in finding the maximum coverage.

m : indicates that all documents are assigned to a cluster seed except cluster seeds ($m-n_c$ is approximated as m, since $m \gg n_c$).

n : indicates that in computing the coverage coefficient of each cluster seed, n terms will be considered.

Accordingly, the overall complexity of the algorithm will be given by:

$$O(m*n) + O(m*n_c*n) \cong O(m*n_c*n).$$

The effect of step-c of the algorithm is assumed negligible. In an operational environment, it can be assumed that the number of documents is considerably greater than the number of terms, i.e., $m \gg n$. With this assumption the complexity of the algorithm would be $O(m*n_c)$.

One might criticize the assumption of $m \gg n$ in the complexity analysis on the grounds that states "the assumption $m \gg n$ is likely to be true only in the case of a large file of documents indexed with a controlled vocabulary and in the case of an uncontrolled vocabulary, m and n might be comparable". However, this is not the fact. Because, if one uses a data structure which gives the related terms of a document, then in the complexity analysis, the term n will be replaced by the average number of terms in each document (n_{avg}). It is obvious that $m >> n_{avg}$. Therefore, the assumption of the complexity analysis hold for all cases. Then the complexity of the algorithm reduces to $O(m*n_c*n_{avg}) \cong O(m*n_c)$.

Within a collection of m documents, on the average, logm documents can be assumed within a cluster |86, p.359|. Therefore, m/logm clusters will be obtained during clustering, i.e., step-b of the algorithm. With this assumption, the complexity of the algorithm will be $O(m^2/\log m)$.

Assuming one document per cluster (i.e., if each document is a cluster seed by itself) the behavior of the algorithm would be as follows. Since each document is a cluster seed by itself, no calculation will be needed to assign it to a cluster seed. This would result in an order of complexity of O(m), for checking whether a document is a seed or not. Therefore, the overall complexity of the algorithm is $O(m*n_{avg}) + O(m) \cong O(m)$, where $O(m*n_{avg})$ comes from the first step of the algorithm (Remember that $m >> n_{avg}$).

It deems necessary to emphasize that this is an inverse behavior with respect to the usual clustering algorithms. This is because as the size of the clusters decreases (i.e., as the number of clusters increases) the computational speed of the algorithm increases.

In the case of having a document collection with very few clusters, the behavior of the algorithm improves. The complexity of the algorithm becomes $Q(m{*}n_{avg}) + O(m{*}n_c{*}n_{avg})$, which reduces to $O(m{*}n_{avg}) + O(m{*}n_{avg}) \cong O(m)$. Accordingly, it can be claimed that the worst case behavior of the algorithm would not be worse than that of the average case.

The assumption made in the analysis of the algorithm (i.e., on the average logm documents in the clusters) implies that in the algorithms $1/\delta \cong \log m$. This is consistent with the example given for 7 documents. A persistent consistency was observed, which validates the assumption, in the experiments to be described in the next chapter.

3.3.5.2. Complexity Analysis of the Multi-pass Algorithm

Similar to the complexity analysis of the single-pass algorithm, the complexity of the multi-pass algorithm can be determined from the selection of cluster seeds and the clustering process. The complexity of the seed selection process is found in the previous section and is $O(m{*}n_{avg})$. The clustering process of this algorithm has two significant aspects: the assignment of the documents to the seeds and the centroid generation process. The complexity of the document assignment to the seeds is again $O(n_c{*}m{*}n_{avg})$, where

n_c : indicates that all cluster seeds will be considered during document assignment.

m : indicates that all documents will be considered during this assignment process. In the first iteration, $(m-n_c)$ documents will be considered, since cluster seeds are assumed as already clustered. Like in the first analysis, since $m \gg n_c$ then $(m-n_c)$ can be taken as m.

n_{avg} : indicates that all the terms of a document is considered during similarity calculations.

After each document assignment process, the centroid formation activity follows. The complexity of this step is $O(m/n_c * n_{avg} * n_c) = O(m * n_{avg})$, $w(m * n_{avg})$, where:

m/n_c : indicates the average number of documents within a cluster

n_{avg} : indicates that during centroid formation all terms are considered, therefore $(m/n_c * n_{avg})$ corresponds to the complexity of centroid generation for a cluster.

n_c : indicates that, centroid formation will be performed for all clusters.

The overall complexity of the algorithm for k iterations would be:

$$O(m*n_{avg}) + O((k-1)*n_c*m*n_{avg}) + O((k-1)*m*n_{avg}) + O(n_c*m*n_{avg})$$
$$=O(k*m*n_{avg}) + O(k*n_c*m*n_{avg}) \cong O(k*n_c*m*n_{avg})$$

In the last, kth, iteration, since there is no change in the cluster configuration, no centroid formation takes place.

If one assumes, on the average, logm documents in each cluster, the value of n_c would be m/logm. Furthermore, $m >> n_{avg}$. This leads to the overall complexity of $O(k*m^2/\log m)$.

If a very few clusters is assumed, the complexity will reduce to $O(k*m)$ from the general complexity of $O(k*m^2/\log m)$. If we assume a large number of clusters, the complexity of the algorithm would be $O(k*m^2)$, where in this case k will be very low, leading to immediate stability of clustering. Because, if n_c is large (which is a result of high term specifity-notice that if the average number of documents in which a term is assigned is low, then this would mean a high term specifity |106, p.25|, then the centroids will be sharply different from each other and very similar to its cluster's members. This leads to the immediate stability of the document assignment process, which is what is observed in the experiments to be presented in the next chapter |14|.

3.3.6. The Use of the Algorithm Concepts for Cluster Maintenance

Efficient cluster maintenance capability is the most important feature for a clustering scheme. The maintenance capability means that updates (e.g. adding documents to the database) should not be too difficult. Many algorithms claim this feature, but in reality they do not provide it |106, p.58|.

One of the well known maintenance approach is "cluster splitting" |89,92|. In this approach, the new comers (documents) are treated like a query and a search is performed according to a query processing strategy. Then the new document are assigned to the most relevant clusters. In this approach, if a cluster becomes too fat then it is split into more than one cluster.

Another maintenance strategy is proposed by Crouch |27|. In this approach, firstly category vectors are generated for the document collection at hand. Where, a category vector is a set of related terms. After this process, the documents are assigned to these category vectors to form the document clusters. This means that each category vector corresponds to a cluster (The cluster centroids are also generated). For the new documents to be clustered the categorization process is restared where it stopped previously. During categorization, some new category classes may be generated and some old category classes may change. During cluster readjustment the new comers are assigned to the category vectors and at the same time the members of the clusters corresponding to the modified category vectors are also reassigned. During this new reassignment, all category vectors are considered.

If one considers the above approaches the weaknesses will be seen very easily. In the cluster splitting process no interaction can exist between a split cluster and unsplit cluster. For example, a document from a split cluster cannot join to an untouched cluster. Although it is not mentioned by him the Crouch's approach is order dependent. Since the generation of category vectors is order dependent.

In the following, an algorithm for cluster maintanence for dynamic (extending) document collections will be introduced. The meaning of the symbols to be used in the algorithm are as follows:

D_o : documents of the old collection (The size of D_o is m_o, $m_o = |D_o|$).

D_N : new (additional) documents to be clustered.

D_E : extended collection, $D_E = D_o \cup D_N$ ($m_e = |D_E|$, obviously $m_e > m_o$).

D_F : documents of the clusters initiated by the cluster seed documents which are not seed anymore (i.e., falsified). If S_E and S_o are the set of cluster seeds corresponding to the document collection D_E and D_o, respectively, then

. $S_F = S_o - (S_o \cap S_E)$, where S_F corresponds to the falsified seeds.

D_c : The set of documents to be clustered after extension, $D_c = D_N \cup D_F$ $(m_c = |D_c|)$.

The algorithm for cluster maintenance is as follows :

a. *Compute the seed power of the documents in the extended document collection,* D_E.

b. *Find the cluster seeds of* D_E*, i.e., the set* S_E*. Determine the previous cluster seeds which are not seed anymore, i.e., the set* S_F.

c. *Determine the set of documents to be clustered,* D_C.

d. *The cluster seeds which cover the documents of* D_C *maximally are found and the documents to be clustered are assigned to the maximally covering seed's cluster.*

It is obvious that the complexity of the above algorithm is much more less than the full utilization of the single-pass algorithm and it can be found by considering each step (In the following n_{avg} indicates the average number of terms in a document). The complexity of the above algorithm can be found by considering the complexity of each algorithm step :

Step-a : The complexity of cluster seed power calculation is $O(m_e*n_{avg})$, the determination of the cluster seeds has a complexity of $m_e*\log m_e$.

Step-b : This step needs n_c number of comparisons. Where $n_c = \max(n_{co}, n_{ce})$. n_{co} and n_{ce} are the number of clusters in the collections D_o and D_E, respectively (Normally, Step-c : Minor.

Step-d : The complexity of this step stems from the calculation of the required C matrix entries. If we assume n_{ce} number of clusters in the extended collection, then the complexity of this step becomes $O(n_{ce}*m_c*n_{avg})$.

Then the overall complexity of the algorithm becomes:

$$O(m_e*n_{avg} + m_e*\log m_e) + O(n_c) + O(n_{ce}*m_c*n_{avg}).$$

This can be approximated by the dominating component which is $O(n_{ce}*m_c*n_{avg})$, if we show that

$$n_{ce}*m_c*n_{avg} > m_e*\log m_e$$

(where it is assumed that $n_{avg} > \log m_e$). If we substitude $n_{ce}=m_e/\log m_e$ in the above expression :

$$m_e/\log m_e*m_c*n_{avg} > m_e*\log m_e \rightarrow m_c*n_{avg} > (\log m_e)^2$$

Obviously the above inequality will hold. Hence, the complexity of the maintenance algorithm would be $O(n_{ce}*m_c*n_{avg})=O(m_e/\log m_e*m_c*n_{avg})$. This is much cheaper than the reinitiation of the single-pass algorithm for the extended document collection D_E, which has a complexity of $O(m_e^2/\log m_e)$. Since we may expact that $m_e^2/\log m_e>>m_e/\log m_e*m_c*n_{avg}$, or $m_e>>m_c*n_{avg}$.

It should be remembered that this cluster adjustment cannot be done indefinetely. After some extension, the cluster falsification can be very dense. In such a case it would be better to perform reclustering operation. Furthermore, automatic indexing could be done periodically to reflect the effects of the new directions in the pertanent science field. For example, the language of the "Annual Review of Information Science and Technology" was studied. It is observed that the vocabulary is changing at a rate about 4% per year, with old terms leaving the vocabulary at about the same rate as new ones enter it |41|. This fact also necessitates the initiation of the reclustering process periodically.

3.3.7. Characteristics of the Algorithms

In the implementation of the algorithms, a new concept called "cover coefficient" is introduced for the selection of cluster seeds. From the description of the algorithms, it is easy to observe the following facts.

a) It is possible to estimate the number of clusters within a collection. This feature is an outcome of the cover coefficient concept and is very novel for a clustering algorithm. This might be used as a check point for the description of the D matrix, i.e., if the number of clusters (n_c) implied by the D matrix is enormously different than the expectation, then one could change the indexing strategy accordingly. Furthermore, the cost of determining the n_c is very low, i.e., $O(m*n_{avg})$.

b) The algorithms are stable, small errors in the description of documents lead to small changes in clustering. Since, small changes in the document collection (D matrix) will lead to small changes in the C matrix.

c) The clustering pattern generated by the algorithms are well defined. Since the algorithms produce a single classification. This statement is absolutely true for the single-pass algorithm. In the case of the multi-pass algorithm, the threshold concept is used to stop the iterations. However, the threshold is chosen in such a way that (i.e., a high value like 95%) the clustering pattern achieved is one of the compatible classifations with respect to the absolute stability of the documents in the clusters (i.e., a threshold of 100%).

d) Many clustering algorithms will produce clusters from any set of data, no matter how dispersed are the entities. However, the cover coefficient concept, which is the starting base of the algorithms, is sensitive to this fact.

e) The algorithms distribute the documents uniformly among the clusters, in other words they do not cause a few "flat" clusters and a lot of singletons, a fact which is the general complaint of the information retrieval community. The observations for this will be depicted in the next chapter |14|.

f) The algorithms are independent of the order of the documents. This is clear from the following facts. The values of the C matrix are independent of the order of documents. This is because, the cover coefficient value between two documents will be the same regardless of the order of documents in the D matrix. In the case of the single-pass algorithm, a document to be clustered is assigned to the seed point which covers it maximally, and again, regardless of the order of the documents, the seed will always be the same seed. In the case of the

.multi-pass algorithm, the centroids are constructed after assigning documents to the seeds and after the first iteration, centroids will be considered as seeds-this is order independent.

g) The single-pass algorithm has a very unique feature, which is the prediction of the number of members of a cluster. This prediction is made according to the magnitude of the cluster seed power of a cluster. In the experiments to be given in the next chapter, a high degree of correlation was observed between the predicted number of documents and the number of documents assigned experimentally |14|. This is again a very unique feature for a clustering algorithm.

h) It is possible to implement the document assignment process of both of the algorithms in such a way that the assignment to different seeds can be overlapped in real time.

i) The cover coefficient concept (δ) implies some basic relationships. These relationships state strong association between the average number of terms per document (depth of indexing), the average number of documents per (term generality) and, respectively, the average number of terms per term cluster, the average number of documents per document cluster. This association also determines the number of clusters (n_c) within a collection. These basic relationships are observed for the first time in this thesis research. They will be discussed in the next chapter.

The implementation of the algorithms requires considerably less memory space. The most of the other algorithms require considerably more memory space and therefore, increase the programming effort. For example, consider an experimental collection with 450 documents, and 4726 terms used for their description |100|. As can be noticed, m is taken as quite small (450) compared to what it should be in real life. The memory requirements of the algorithms and another algorithm which uses the (dis)similarity of documents in a graph theoretical manner can be comparatively determined as follows: if we skip the representation of the D matrix which might be the same in both algorithms, the (dis)similarity matrix, for the other algorithm, will require $|m*(m-1)/2$ elements$|$ $(450*449)/2=101,025$ memory locations. For the algorithms presented, this would take 450+4726+450=5,625 memory locations. These are the storages required for α, β, δ_i. The substantial difference between the two storage requirements also indicates the power of the presented algorithms.

The memory requirements of a graph theoretical algorithm can be resolved by tricky programming. For example the entries of the (dis) similarity matrix can be calculated in segments |106, p.60|. Another way of doing this is the exploitation of the inverted file structures provided that the algorithm at hand should accept the (dis)similarity at any order and fetch of the same matrix entry more than once |25|. By such an algorithm the memory requirements is lowered from 67,425,078 to 3,969,046 (5.9% of the entire matrix) for a document collection of 11,613 entries |25|. In the clustering algorithms presented in the thesis the number of elements to handle the diagonal entries of the C-matrix is 11,613. For the document assignment process, if we assume n_c=1000 (which is a very large number, probably larger than the case of the real life) for the execution of the single-pass algorithm and one pass of the multi-pass algorithm about 1,161,300 numbers will be calculated. These figures are considerably less than the figure given for the graph theoretical algorithm. In the single-pass algorithm, the entries of the C matrix, which is calculated during document assignment, are used immediately, i.e., they are not saved. The same is also valid for the multi-pass algorithm in the calculation of similarity values.

The complexity of the single pass algorithm can be further decreased if a file structure, which will indicate the disjoint documents and cluster seeds, is used (The disjoint documents do not have any common term). In such an approach, the complexity of clustering step will drop from $0(n_c*m*n_{avg})$ to $(n_c*(m-\ell)*n_{avg})$, where $\ell = \sum_{i=1}^{n_c}$ (No of documents that are disjoint with cluster seed-i). (Although $m>>n_{avg}$, the n_{avg} term is not dropped. However, it will vanish automatically in the following calculations). The complexity of calculating ℓ is $0(n_c*(m-n_c)*n_{avg})$. Where,

n_c : indicates that all cluster seeds are considered.

n_{avg}: each seed is represented by n_{avg} number of terms.

$m-n_c$: all documents except cluster seeds are considered to find the disjoint documents.

Therefore, in order to have some reduction in the complexity of the algorithm, the following inequality should be satisfied:

$$n_c*(m-\ell)*n_{avg} + n_c*(m-n_c)*n_{avg} < m*n_c*n_{avg}$$

$$(m-\ell+m-n_c)*n_c*n_{avg} < m*n_c*n_{avg}$$

$$2m-\ell-n_c < m$$

$$m < \ell+n_c.$$

This inequality is very easy to satisfy. For example, if there are three seeds which are disjoint with one-half of the documents then the inequality is readily satisfied. Therefore, the determination of the disjoint documents will be very useful in a real life enviroment.

4. SIMILARITY AND STABILITY ANALYSIS OF THE ALGORITHMS

In this chapter a similarity and stability analysis for the clustering algorithms, which are presented in the previous chapter, will be given. This chapter also includes other aspects of the algorithms, such as the number and the size of the clusters generated, the similarity between the members of a cluster and the corresponding centroid, the close association between indexing policy and the distribution of documents among the clusters, and other relevant details.

Two methods will be used for the similarity and stability measurements. These methods are due to Rand |2,82| and Goodman and Kruskal |2,37|. To test the similarity and stability aspects of the algorithms with respect to these methods, a database from 167 ACM TODS (Association for Computing Machinery, Transactions on Database Systems) has been constructed for the experiments. The features of the document collection as well as the other findings of the experiments will also be given.

Let us first give an informal definition of similarity and stability of clustering algorithms. A clustering algorithm is considered stable if small changes in the input leads to small changes in the clustering pattern generated. Two clustering algorithms are assumed similar if the clustering pattern generated by one resembles the clustering pattern generated by the other. A metric used to measure similarity might be also used for stability. The similarity concept is used to compare the results of more than one clustering algorithm for the same input. If the clustering patterns generated by an algorithm for different input are similar, then this similarity might be a clue for the stability of the algorithm. Notice that, by different input, what is implied here is different descriptions of the same collection |14|.

In terms of document clustering, a clustering algorithm is called "stable under extension of range" when introduction of new objects does not drastically alter it. Similarly, it is called" stable under change or increase of (indexing) terms" when addition of further relevant terms does not substantially alter it. The definition of stability within these two contexts is given by Jardine and Sibson|50|.

In view of various sources of errors, such as in data conversion, and changes in document collection (both in terms of documents and terms), stability testing of clustering becomes necessary. As an example, one can list those due to input data as follows |79|:

a. In converting a document representation into a standard computer form, various sorts of clerical errors may be introduced (Such as misspelling).

b. A term which is not important or does not appear in a document may be mistakenly assigned (erroneous indexing).

c. Similar to (b), a term which is important for a document may be mistakenly unassigned.

d. Addition and deletion of documents, respectively, due to collection growth and document retirement, might change the importance of terms. Because of such dynamism, some terms can lose their importance, and some new terms might be introduced with the new documents.

e. The terms which are chosen to represent a document collection may not be unique. Two different indexing strategies may produce similar but different set of indexing terms.

The above sources of errors and/or perturbations clarify the importance of stability analysis for document clustering. However, the task of stability analysis is not straightforward. For example, the effect of different indexing policies on the same document collection can be seen by obtaining the document-document similarity matrices for two policies and, then, determining the similarity between these two matrices by using a measure (e.g. Kendal's coeifficient of agreement |80|). A similar approach cannot be used to compare two different clustering patterns. Since in this case, there is no one to one correspondence among the clusters of the two distinct clustering patterns. The next subsection will deal with the metrics for this measurement.

4.1. MEASURES OF SIMILARITY AND STABILITY

One might have different clustering patterns for the same collection due to different clustering algorithms. The same may happen by using the same algorithm with different D matrices corresponding to the same collection (The difference can be due to different indexing policies). The similarity coefficient calculated for two different clustering patterns might indicate the similarity of two algorithms if they are using the same D matrix for clustering. The same similarity measure can be used to show the stability of an algorithm if the differences in clustering patterns are due to the D matrix rather than the algorithms. If an algorithm is immune to the discrepancies in the definition of D matrices, then the similarity coefficient will have a high value.

There are various stability measurement algorithms in the literature. Among those, an algorithm for graph theoretical clustering methods can be seen in |78|. An overview of the stability analysis methods for clustering algorithms can be found in |2,79,80|. The complexity analysis for stability algorithms can be found in |29|.

In this thesis, two different methods will be used for stability analysis. The first one is due to Rand |82|. It stems from the similarity concept. The second stability measure is due to Goodman and Kruskal |37| which stems from the concept called "predictivity power". In this concept, if there are two clustering patterns, one may use the information held in one of them to predict the clustering pattern of the other. This concept was introduced for the cross classifications of tables. However, later, also used for partitioned clustering |2|. In the following section, both metrics will be introduced very briefly.

4.1.1. The Rand's Coefficient

The Rand's coefficient can be used for a partitioning type clustering algorithm. It can also be modified for overlapping clustering algorithms |80|. Another metric, called D metric in the literature is identical with the Rand's coefficient |79|.

According to this metric, clusters are defined just as much by those points (in IR documents) which they do not contain as by those points which they do contain. This means that if a distinct document pair is assigned either to the same cluster or to different clusters in two different partitionings of the same input data, then, the Rand's coefficient assumes a similarity between these two clustering patterns. This means that, a pair of documents are considered to be similarly placed either if the pair is in the same cluster in both clustering patterns or they are in different clusters in both clustering patterns.

The Rand's similarity measure can be expressed in tabular form |2|. In this representation, pairwise combinations of documents in each partition are classified into two classes:

Class-0 : the documents are in different clusters in the partition;
Class-1 : the documents are in the same cluster in the partition;

The pairwise comparisons between two partitions -say P_A and P_B- are given in the following chart.

P_A / P_B	Class-1	Class-0
Class-1	a_{11}	a_{10}
Class-0	a_{01}	a_{00}

The meaning of the entries of the above chart are as follows:

a_{11} : number of document pairs which appear in the same cluster, in P_A and P_B.

a_{10} : number of document pairs which appear in the same cluster in P_A, but in different clusters in P_B.

a_{01} : number of document pairs which appear in different clusters in P_A, but in the same cluster in P_B (reverse of a).

a_{00} : number of document pairs which appear in different clusters in both P_A and P_B.

The Rand's coefficient in terms of these variables is expressed as follows :

$$c(P_A, P_B) = (a_{11} + a_{00}) \,/\, (a_{11} + a_{10} + a_{01} + a_{00})$$

A computationally more efficient way of calculating this coefficient and a detailed illustration of its utilization for comparing the clustering algorithms can be found in |21,82|. The value of c ranges from 0 to 1. c is 0 when two partitions are not similar (i.e., when one consists of a single cluster and the other consists from singletons) and c = 1 when partitions are identical.

4.1.2. The Goodman-Kruskal Coefficient

The Goodman-Kruskal metric is similar to the chi-square based association measure in statistics and uses a cross classification of partitions. If there are p and q clusters in the partitions P_A and P_B, respectively, the cross classification of these partitions can be done as shown in Table 4.1.

TABLE 4.1 - Cross Classification of Two Partitions P_A and P_B.

P_A/P_B	1	2 . . . q	TOTALS
1	n_{11}	$n_{12} \cdots .n_{1q}$	$n_{1.}$
2	n_{21}	$n_{22} \cdot \cdot \cdot n_{2q}$	$n_{2.}$
.	.		.
.	.		.
p	n_{p1}	$n_{p2} \cdot \cdot \cdot n_{pq}$	$n_{p.}$
TOTALS	$n_{.1}$	$n_{.2} \cdot \cdot \cdot n_{.q}$	$n_{..}$

In Table 4.1, the meaning of the entries are as follows :

n_{ij} : indicates the number of documents which appear in cluster $C_i \varepsilon P_A$ and cluster $C_j \varepsilon P_B$ jointly;

$n_{i.} = \sum_{j=1}^{q} n_{ij}$: indicates the number of documents in C_i $(C_i \varepsilon P_A)$

$n_{.j} = \sum_{i=1}^{p} n_{ij}$: indicates the number of documents in C_j $(C_j \varepsilon P_B)$;

$n_{..} = \sum_{i=1}^{p} \sum_{j=1}^{q} n_{ij}$: indicates the total number of documents in the collection

where, $1 \leq i \leq p$ and $1 \leq j \leq q$

After introducing the following, Goodman and Kruskal proposed three metrics λ_A, λ_B, and λ.

$n_{m_j j}$: maximum entry in the jth column;

$n_{m.}$: maximum row sum;

n_{im_i} : maximum entry in the ith row;

$n_{.m}$: maximum column sum.

In terms of the above variables and the entries of Table 4.1 λ_A, λ_B, and λ are defined as follows:

$$\lambda_A = \frac{\sum_{j=1}^{q} n_{m_j j} - n_{m.}}{n_{..} - n_{m.}}$$

$$\lambda_B = \frac{\sum_{i=1}^{p} n_{im_i} - n_{.m}}{n_{..} - n_{.m}}$$

$$\lambda = \frac{\sum_{i=1}^{p} n_{im_i} + \sum_{j=1}^{q} n_{m_j j} - n_{.m} - n_{m.}}{2n_{..} - n_{.m} - n_{m.}}$$

Within the context of this study λ_A, λ_B, and λ means the following:

λ_A : indicates the relative decrease in the probability of error in predicting the unknown cluster of a document in P_A by knowing the cluster of the document in P_B.

λ_B : indicates the relative decrease in the probability of error in predicting the unknown cluster of a document in P_B by knowing the cluster of the document in P_A.

λ : indicates the relative decrease in error probability if the prediction of P_A from P_B is taken as being equally important as the prediction of P_B from P_A. In other words λ is the relative decrease in error probability due to the use of predictor partitions when the directions of prediction are equally important.

Some properties of λ are : |37, p.743|:

a) λ is determinate except when the entire population lies in a single cell of the table. Otherwise, the value of λ is between 0 and 1 inclusive and λ_A and λ_B inclusive (i.e., $0 \leq \lambda \leq 1$, and $\min(\lambda_A, \lambda_B) \leq \lambda \leq \max(\lambda_A, \lambda_B)$.

b) λ is 1 iff the entire population lies in isolated cells, i.e., cells which are the only nonnull cells in both the rows and columns.

c) λ is 0 in the case of statistical independence, but converse need not hold.

d) λ is unchanged by permutation of rows and columns.

4.1.3. An Example for Cluster Similarity Calculation

In this section an example of cluster similarity calculation by using the Rand's coefficient and the Goodman-Kruskal predictivity power will be given. The definition of the document collection, i.e., the D matrix, is taken from |86, p.351|. This D matrix is also used in the previous chapter. Using the single-pass and the multi-pass algorithms, which are illustrated in the previous chapter, one obtains the following clusters: $C_{A1} = (1,2)$, $C_{A2} = (4,6)$, $C_{A3} = (3,5,7)$. The clustering of these documents with respect to another algorithm |88, p.351| produces the following clusters : $C_{B1} = (1,2,4)$, $C_{B2} = (3)$, $C_{B3} = (7)$, $C_{B4} = (5,6)$, where, $C_{Ai} \varepsilon P_A$ $(i = 1,2,3)$ and $C_{Bi} \varepsilon P_B$ $(i = 1,\ldots,4)$. P_A and P_B corresponds to two partitions.

The.similarity of the two partitions according to the Rand's coefficient can be found as follows. The document pair <1,2> is assigned to the same clusters; the document pairs <1,3>, <1,5>, <1,6>, <1,7>, <2,3>, <2,5>, <2,6>, <2,7>, <3,4>, <3,6>, <4,5>, <4,7>, <6,7> are assigned to seperate clusters in both partitions. However, the document pairs <1,4>, <2,4>, <3,5>, <3,7>, <4,6>,<5,6>, <5,7> are assigned to the same cluster in one partition but to separate clusters in the other partition. Using this, it is easy to determine 14 similarities out of 21, which gives the Rand's similarity coefficient as 14/21 = 0.67.

The Goodman-Kruskal predictivity power coefficients for the above partitions can be determined as follows. The cross classifications of the partitions are given in Table 4.2.

TABLE 4.2 - Cross Classification of the Partitions P_A and P_B

P_A/P_B	c_1	c_2	c_3	c_4	$n_{A.}$
c_1	2	0	0	0	2
c_2	1	0	0	1	2
c_3	0	1	1	1	3
n.B	3	1	1	2	n.. = 7

Using Table 4.2 we have :

$n_{1m_1} = 2$ $\qquad$ $n_{m_1 1} = 2$

$n_{2m_2} = 1$ $\qquad$ $n_{m_2 2} = 1$

$n_{3m_3} = 1$ $\qquad$ $n_{m_3 3} = 1$

$$n_{m_4 4} = 1$$

$$n_{.m} = 3 \qquad n_{m.} = 3$$

$$\lambda_A = \frac{(2+1+1+1) - 3}{7 - 3} = \frac{2}{4} = 0.5$$

$$\lambda_B = \frac{(2+1+1) - 3}{7 - 3} = \frac{1}{4} = 0.25$$

$$\lambda = \frac{(2+1+1) + (2+1+1+1) - 3 - 3}{2 * 7 - 3 - 3} = \frac{3}{8} = 0.375$$

In this example $(\lambda_A + \lambda_B) / 2 = \lambda$. However, this is just a coincidence.

4.2. EXPERIMENTAL PROCEDURE FOR THE ANALYSIS

The main purpose of the experiments conducted during this thesis research was to test the similarity and the stability of the two clustering algorithms. This would also explore the behavior of the algorithms for the changes in the document collection and the utility of the new cover coefficient concept. The two main questions that were investigated are the following:

a. How sensitive are the algorithms to perturbations of data (documents)?

b. How sensitive are the algorithms to additional documents?

The following was the test procedure |14|.

a. Generate the cluster for the base D matrix.

b. Perturb the D matrix according to the sensitivity to be measured, i.e, either change the number of terms used for the description of the D matrix, or change the number of documents (Notice that the change in the number of documents may also change the number of terms used for the description of the D matrix although the condition for being a term has been kept fixed).

c. Obtain the clusters corresponding to the perturbation D matrix.

d. Obtain the similarity of the algorithms by comparing the partitions obtained by using the same D matrix for the two algorithms. Obtain the stability of the algorithms by comparing the results of the base and perturbation matrices.

e. Repeat steps (b) through (d) for each experiment to be carried out.

In the experiments, the base D matrix represents the so-called "ideal conditions". Accordingly, the perturbation matrix represents the deviations from the ideal. The experiments that dealt with the changes in the number of terms are referred to as the T-experiments and those dealt with the changes in the number of documents are referred to as the D-experiments. In the T-experiments the number of documents in the collection was kept fixed. However, in the D-experiments the number of documents were changed and as it is mentioned before, the change in the documents also affected the number of terms.

In the rest of this chapter, first the properties of the document collection which is used in the experiments are illustrated. This is followed by the experiments themselves and their interpretation. In the remainder of this chapter the single-pass and the multi-pass algorithm will be referred to as the Algorithm-1 and the Algorithm-2, respectively.

4.2.1. Document Collection and the Indexing Approach

A document collection was constructed for the purpose of the experiments. The papers of the Association for Computing Machinery Transactions on Database Systems (ACM-TODS) available since 1976 were entered into the corresponding database. The first seven volumes covering the period between 1976 and 1982 inclusive provided 167 papers. For automatic indexing purposes, the database was built with the titles, keywords given by the authors, and the abstracts of the papers.

Each document of the database consists of a collection of words entered from the selected parts of the papers as it is mentioned above. A word is taken as any sequence of characters, whose total string length is three or more and which begins with a letter and followed by letters, numbers or apostrophe. The maximum length of a word is set to eight, which is the average length of an English word |76|. Words having more than eight characters are truncated at the right. The words which appear in the stop list (a list which contains the frequently used words of the English language and the database literature) are not considered for indexing purposes (The stop list words are provided in Appendix-B). The general characteristics of the document collection are summarized in Table 4.3.

In the experiments, it is assumed that a word has the weights of 2, 1, and 3 if it appears in a title, an abstract or keyword list, respectively |14,53,54|. This means that a word should appear 3 times

TABLE. 4.3 - General Statistical Information About the Database

Total number of documents	:	167
Total size of the document database	:	190 K byte (approx)
Number of words in the stop list	:	134
Number of words in the collection	:	18936
Number of distinct words in the collection	:	2389
Percent of the stop words in the collection	:	28
Number of words/doc (min,max,avg)	:	28,260,113
Number of distinct words/doc (min,max,avg)	:	18,125,52
Number of sentences/doc (min,max,avg)	:	1,18,6

within an abstract to have the same weight with a word that appears in a keyword list.

For a word in the collection, to qualify it as a term some requirements must be met. These requirements are two fold. Firstly, a word should appear within a range of frequencies within the documents. Secondly, it should satisfy a similar range requirements for its weight. This means that to satisfy the frequency (weight) requirements, the number of occurrences (the total weight, i.e., the sum of all weights according to the place of the term in the document) of a word should be greater than or equal to a minimum and less than or equal to a maximum value. It is obvious that by changing the range requirements one may obtain different terms for the description of the same collection.

The D matrices of the experiments were generated from the document database. To construct a D matrix, the terms of the documents were identified. The simplest indexing procedure, which describes a document by the terms used in it as a binary state variables was adapted |86, p.102|.

4.2.2. Generation of D Matrices

In the generation of the base D matrix, some conditions which would enable the generation of the perturbation D matrices were created. For example, to test the sensitivity of the algorithms to perturbations of data we need to change the number of terms or the number of documents used for the description of the D matrix.

In the T-experiments, in the generation of D matrix for the experiment T4, the conditions chosen for the terms, led to a medium number of terms for the description of the base D matrix. These conditions for T4 are chosen as follows: the total weight of a word for being a term should be within the range of 5 to 80, and similarly, the number of occurrences should be within the range of 3 to 30. These conditions, i.e., the weight and the number of occurrences (frequency) should be satisfied at the same time. For example, assume that a word appears in four different documents and always in the abstracts, then this word satisfied the requirement for the number of occurrences. However, it does not satisfy the weight condition for being a term. This is because, the total weight of the term is 4 and does not satisfy the weight requirement (remember that if a word appears in an abstract, then it will gain a weight of 1 for each occurrence of it). Therefore, this word will not appear in the description vectors of the documents.

By changing the conditions for being a term one would have different D matrices for the same collection. More rigid conditions will lead to less number of terms and looser conditions will lead to higher number of terms. The experiments (T1, T2, T3) and (T5, T6, T7) correspond to the experiments which rigid and loose conditions for the terms, respectively. Table 4.4 shows the conditions for a word to be a term in the T-experiments. These different conditions will lead to the generation of different D matrices for the same document collection.

TABLE 4.4 - The Term Generation Conditions in the T - experiments

Exp No	Min Weight	Max Weight	Min No of Occ.	Max No of Occ.
T1	8	60	5	15
T2	7	60	4	20
T3	6	75	4	25
T4	5	80	3	30
T5	4	85	2	40
T6	3	90	2	50
T7	2	95	1	60

TABLE 4.5 - No of Documents in the D-experiments

Exp No	No of Doc
D1	78
D2	93
D3	108
D4	123
D5	138
D6	153
D7	167

To test the sensitivity of the algorithms for additional documents, a medium number of documents (123 documents) were used for the generation of the base D matrix. For the perturbation D matrices, the number of documents used for the description of the D matrix was gradually decreased or increased. To make this change controlled by only the variations in the number of documents, the term conditions were kept fixed (In the D experiments, minimum and maximum number of occurrences are 2 and 35, minimum and maximum weights are 4 and 100, respectively for a word to be a term). In the D-experiments, the collection with the higher number of documents will include the documents of the smaller collection and in addition some more documents. D4 corresponds to the base D matrix. The experiments, (D1, D2, D3) and (D5, D6, D7) correspond to the experiments with less and more number of documents with respect to the experiment D4-see Table 4.5.

4.2.3. Behavior of the Algorithms in the Experiments

Table 4.6 contains the information pertaining to the definition of the D matrix and the clusters generated in the experiments. In the T-experiments (T1 thorugh T7), it was observed that the conditions set for the D matrix introduce a smooth change in the number of terms. As the number of terms for the description of the documents increases, the average number of terms per document (depth of indexing = x_d) also increases. However, the average number of documents per term (i.e., the average number of documents the term is assigned, or term generality = t_g) decreases.

The relationship between x_d and t_g are as follows |58|:

TABLE 4.6 - Information Pertaining to the Definition of D Matrices and the Clusters Generated.

EXP NO	NODOC (m)	NOTERM (n)	AVG NO TERM/DOC (x_d)	AVG NO DOC/TERM (t_g)	NO OF CLUSTERS ($n_c = n'_c$)	DOC.DECOUP COEFF. (δ)	AVG NO DOC/CLUST (d_c)	TERM DECOUP COEFF (δ')	AVG NO TERM/CLUST (d'_c)	logm
T1	167	243	12.94	8.89	19	0.113	8.79	0.077	12.79	7.38
T2	167	341	18.02	8.83	19	0.114	8.79	0.055	17.95	7.38
T3	167	406	21.40	8.80	19	0.113	8.79	0.047	21.37	7.38
T4	167	539	24.94	7.77	22	0.130	7.59	0.040	24.50	7.38
T5	167	747	30.26	6.76	25	0.148	6.68	0.033	29.88	7.38
T6	167	899	33.93	6.30	26	0.158	6.42	0.030	34.58	7.38
T7	167	1391	39.02	4.68	35	0.212	4.77	0.026	39.74	7.38
D1	78	411	29.73	5.64	14	0.177	5.57	0.034	29.36	6.29
D2	93	457	29.56	6.02	15	0.166	6.20	0.033	30.47	6.54
D3	108	532	30.82	6.26	17	0.160	6.35	0.032	31.29	6.75
D4	123	600	31.33	6.42	19	0.155	6.47	0.032	31.58	6.94
D5	138	656	31.71	6.67	21	0.149	6.57	0.032	31.24	7.11
D6	153	712	31.56	6.78	22	0.147	6.96	0.031	32.36	7.26
D7	167	754	31.07	6.88	24	0.145	6.96	0.032	31.42	7.38

$$t = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} \qquad \text{(total number of term assignments)}$$

$$x_d = \frac{t}{m} \qquad \text{(depth of indexing)}$$

$$t_g = \frac{t}{n} \qquad \text{(term generality)}$$

$$t = x_d m = t_g n$$

After the above definition we can introduce term specifity (t_s) as the inverse of term generality ($t_s = 1/t_g$). So, if all terms are assigned only one document, then $t = n$ will lead to $t_s = t_g = 1$. By these definitions of t_g and t_s, their theoretical range is given as follows : $1 \leq t_g \leq m$, $1 \geq t_s \geq 1/m$. Similarly, the possible values of x_d is as follows $1 \leq x_d \leq n$.

After making the above definitions, the decrease in t_g comes from the fact that the rate of increase in n (Δn) is higher than the rate of increase in t (Δt), i.e., $\Delta n > \Delta t$. This means that, the terms introduced by loosening the conditions for being a term increases t_s.

For the D-experiments (D1 through D7), with the increase in the number of documents in the collection, the number of terms used for the description of the D-matrix also increases. However, there is no significant change in the term generality (t_g) and the depth of indexing (x_d). This comes from the fact that the rate of increase in m (number of documents) and n (number of terms used for indexing) are identical.

The average decoupling of the documents (δ) smoothly increases as the number of terms increases in the T-experiments. This is obvious, since in the T-experiments, the newly introduced terms -i.e., additional terms- will make the documents more and more unique. In other words, new terms decreases (increases) term generality (specifity). Lower (higher) $t_g(t_s)$ means less and less similar documents. In short, it is observed that δ is inversely (directly) related to $t_g(t_s)$. However, in the case of D-experiments, as we introduce more documents to the collection, the coupling among the documents will increase. This is because, as we introduce more documents, we also introduce new terms which are common among the documents. Since the words which do not satisfy the conditions for being a term before satisfy these conditions with higher number of documents. This decreases t_s.

A similar discussion can be given for the decoupling of the terms (δ'). For the case of δ', it is observed that, δ' is inversely related to the depth of indexing (x_d), which is obvious. Since, lower x_d will lead to more unrelatedness among the terms.

The number of clusters in the collection, n_c, increases as the number of terms increases in the T-experiments. This comes from the increase in δ. The number of term clusters, n'_c, (where $n'_c = n_c$) increases also. However, in this case the increase can be attributed to the increase in the number of terms ($n'_c = n*\delta'$). Notice that, as we go from T1 to T7 δ' decreases, but n increases with a higher rate. In case of the D-experiments, the increase in n_c is mainly because of newly added documents rather than the changes in δ. In fact, in this case δ decreases, but n_c increases as we go from D1 to D7. The increase of n'_c for term clusters should also be attributed to the increase in number of terms. Since throughout the D-experiments the δ' values are nearly constant.

The average number of documents (terms) per cluster, $d_c(d_c')$, decreases (increases) as one goes from T1 to T7. This is because of the increase (decrease) in (δ'). (Notice that $d_c = 1/\delta$ and $d_c' = 1/\delta'$) In the case of the D-experiments, d_c does not vary too much (changes from 5.57 to 6.96). Because as n_c increases, the number of documents in the collection also increases. d_c' is nearly constant throughout the D-experiments because of the (nearly) constant value of δ'.

In the analysis of the algorithms it is assumed that within a document cluster there would be logm number of documents. This assumption leads to the following equalities in the algorithms presented in this thesis: $\log m = 1/\delta = d_c$. If one looks at the values given in Table 4.6 it will be observed that this assumption is nearly always validated. Especially in the case of mild conditions (the base matrices used in T4 and D4) the assumption shows very little deviation from the experimentally observed values (The theoretically implied d_c's -logm values- are 7.38, 6.94 according to the assumption and experimentally observed values are 7.59 and 6.47, respectively, for the experiments T4 and D4).

If we look at Table 4.7, we can observe the behavior of both algorithms in the T-experiments. The meanings of the abbreviations in Table 4.7/A and Table 4.7/B are as follows: EXP NO/ALG TYPE: experiment no/algorithms type; NO OF ITER: number of clustering iterations which is not applicable (NA) for the Algorithm-1; EXEC TIME: execution time; CLUST SIZE COEF OF VAR: coefficient of variation of cluster size; AVG NO TERMS USED BY CLST MBRS: average number of distinct terms used by cluster members; AVG NO TERM USED IN CENT: average number of terms in centroids, i.e., the number of 1's in the binary centroid vector; m_p: when comparing a cluster member and the corresponding centroid vector the probability of having a mismatch at a term position; NO OF DSTNC CENT TERMS: number of distinct centroid terms, i.e., the number of terms which appear at least once in a centroid vector; % OF DOCS AT MST SIM CLST: % of documents assigned to the cluster which correspond to the most similar centroid; COR. BTW THEO & EXP #MBRS : correlation between the theoretical and experimental number of documents (members) within a cluster; NO OF SNG/MCS: number of singletons/ maximum cluster size. These abbreviations are also valid for Table 4.8/A and Table 4.8/B.

Table 4.7/A - Behavior of the Algorithms in the T-experiments.

EXP NO/ ALG TYP	NO OF ITER	EXEC TIME	CLUST SIZE COEF OF VAR	AVG NO TERMS USED BY CLST MBRS	AVG NO TERM USED IN CENT
T1 / 1	NA	0.38	0.34	75.95	25.89
T1 / 2	4	1.05	0.25	75.68	8.47
T2 / 1	NA	1.08	0.52	104.53	33.63
T2 / 2	3	1.29	0.43	102.32	10.16
T3 / 1	NA	1.32	0.43	124.53	46.11
T3 / 2	3	1.52	0.23	126.32	13.05
T4 / 1	NA	2.31	0.44	133.77	49.95
T4 / 2	3	3.20	0.32	134.09	14.00
T5 / 1	NA	4.28	0.34	153.20	55.56
T5 / 2	3	6.10	0.43	146.08	24.12
T6 / 1	NA	6.01	0.48	162.96	59.77
T6 / 2	3	8.24	0.57	156.31	29.31
T7 / 1	NA	12.51	0.54	148.63	55.03
T7 / 2	2	11.56	0.56	144.80	48.43

Table 4.7/B - Behavior of the Algorithms in the T-experiments.

EXP NO/ ALG TYP	m_p	NO OF DSTNC CENT TERMS	% OF DOCS AT MST SIM CLST	COR BTW THEO & EXP # MBRS	NO OF SNG/MCS
T1 / 1	0.11	233	93	0.22	0 / 15
T1 / 2	0.06	128	99	-0.10	0 / 12
T2 / 1	0.12	320	95	0.45	0 / 19
T2 / 2	0.05	145	97	-0.10	0 / 15
T3 / 1	0.13	376	96	0.64	1 / 17
T3 / 2	0.06	184	97	0.22	0 / 11
T4 / 1	0.11	459	96	0.70	0 / 16
T4 / 2	0.05	221	95	0.10	0 / 14
T5 / 1	0.08	548	99	0.41	0 / 11
T5 / 2	0.04	358	98	0.34	0 / 12
T6 / 1	0.08	635	98	0.25	1 / 12
T6 / 2	0.04	427	98	0.01	2 / 15
T7 / 1	0.05	806	98	0.60	2 / 11
T7 / 2	0.03	872	95	-0.11	2 / 11

In the T-experiments, it is observed that, the execution speed of the Algorithm-1 is noticeably faster than that of the Algorithm-2, except for T7. The execution speeds are given in terms of (minute, second). These experiments are performed by using the computing facilities of the Arizona State University. The system was IBM 3081. At the Middle East Technical University, experiment T4 is repeated on Burroughs 6930 by using the original form of the program. The execution time was observed as more than 30 minutes. By some optimization efforts the execution time is lowered to 5 minutes (Actually the

TABLE 4.8/A - Behavior of the Algorithms in the D-experiments.

EXP NO/ ALG TYP	NO OF ITER	EXEC TIME	CLST SIZE COEF OF VAR	AVG NO TERMS USED BY CLST MBRS	AVG NO TERM USED IN CENT
D1 / 1	NA	0.40	0.56	112.71	46.43
D1 / 2	2	0.53	0.59	111.36	39.07
D2 / 1	NA	0.57	0.62	122.07	49.07
D2 / 2	2	1.15	0.62	121.07	37.93
D3 / 1	NA	1.29	0.58	133.76	52.47
D3 / 2	2	1.20	0.58	129.41	27.53
D4 / 1	NA	2.06	0.58	142.32	55.16
D4 / 2	3	2.57	0.45	142.11	32.11
D5 / 1	NA	2.51	0.57	147.33	54.76
D5 / 2	3	3.54	0.57	144.90	36.29
D6 / 1	NA	3.35	0.46	159.27	57.64
D6 / 2	3	4.54	0.55	151.64	28.33
D7 / 1	NA	4.23	0.37	161.04	58.46
D7 / 2	3	5.55	0.42	152.54	25.21

new form of the program contains some extra code which does have some contribution to this 5 minutes). Therefore, if one runs the same program on the IBM 3081 system probably execution time speed will be lowered at least by a factor of 6 (30/5).

After a similar optimization effort, the execution time of the Algorithm-2 is lowered from 36.03 (min.sec) to 13.14 (min.sec) for the experiment T4, i.e., the execution time is lowered by a factor of 2.72. This means that, the optimization effort for the Algorithm-2

TABLE. 4.8/B - Behavior of the Algorithms in the D-experiments

EXP NO/ ALG TYP	m_p	NO OF DSTNC CENT TERMS	% OF DOCS AT MST SIM CLST	COR BTW THEO & EXP # MBRS	NO OF SNG/MCS
D1 / 1	0.14	313	97	0.60	0 / 13
D1 / 2	0.08	291	100	0.66	1 / 12
D2 / 1	0.14	363	96	0.59	0 / 15
D2 / 2	0.07	319	100	0.29	0 / 14
D3 / 1	0.13	413	96	0.56	0 / 16
D3 / 2	0.06	276	95	0.33	0 / 12
D4 / 1	0.12	457	99	0.50	0 / 17
D4 / 2	0.06	330	99	0.14	0 / 13
D5 / 1	0.11	496	95	0.60	0 / 17
D5 / 2	0.05	398	100	0.12	0 / 17
D6 / 1	0.10	535	96	0.52	0 / 14
D6 / 2	0.05	367	97	0.39	1 / 15
D7 / 1	0.09	555	98	0.32	0 / 10
D7 / 2	0.04	365	96	0.18	1 / 13

yields an execution time reduction which is less than that of the Algorithm-1. This is because, in the Algorithm-2 the similarity calculation takes place between a document and a centroid if the cluster seeds are replaced by the generated centroids (the case when the number of clustering pass is greater than 1). (The access method for the centroid entries was already efficient in the original form of the program for the Algorithm-2). After these observations it can be said that the execution times for the algorithms will become more close to each other than that of the values given in Table 4.7/A and 4.7/B.

The new execution times should not be regarded as the best possible time for the algorithms.

The number of iterations observed in the case of Algorithm-2 ranges from 2 to 4, which is not that bad. Higher term generality (t_g) (or lower term specifity, t_s) introduces sensitivity to the similarity calculations (between document and centroid vectors) of the Algorithm-2. This can be observed from the higher number of iterations in the Algorithm-2 (see the experiment T1). The reverse, i.e., lower term generality (higher term specifity) leads to faster execution as in the experiment T7. For the similar observations for the D-experiment, one should refer to Table 4.8. In each D experiment, the Algorithm-1 runs faster than the Algorithm-2. The number of iterations in this case ranges from 2 to 3. The growth of the D matrix (see Table 4.6) increases the execution time.

As can be seen from the low values of coefficient of variation (standard variation/average value) for cluster size, the distribution of documents in clusters is quite uniform. That is, there is almost no singletons and no fat clusters. These facts can be observed from Table 4.7 and Table 4.8.

Again from the same tables, it can be observed that the cluster members use almost the same number of different terms in both experiments (this is given by the average number of terms used by the cluster members). This, somewhat, indicates the compatibility of the clusters generated by the two algorithms. The tables also provide the values of m_p. In general, the m_p values are small and this is a hint for a good clustenig algorithm and a good centroid generation policy. The m_p values corresponding to the Algorithm-1 are about twice of those given for the Algorithm-2. However, this is not so bad. Because, the average number of terms used in the centroids (the last columns of Table 4.7/A and Table 4.8/A) of the Algorithm-1 is about twice as many as those that are used for the Algorithm-2. Therefore, we may say that the centroids are compatible.

The number of distinct centroid terms (i.e., those that appear at least once in the centroid definitions) show considerable differences. On the average, 81% and 49% of the terms appear at least once in the centroid vectors in the T-experiments for the Algorithm-1 and the Algorithm-2, respectively. The same values for the Algorithm-1 and the Algorithm-2 in the D-experiments are 76% and 58%, respectively. These values can be changed by modifying the conditions for a term to appear in a centroid (see section 3.3.4.1).

After the documents are clustered, if one uses a document vector as a query, one should be able to reach the cluster which holds the document. This was successfully accomplished in both of the clustering schemes. For the Algrotihm-2, it is something natural, since it is the mean of clustering. However, the same was accomplished also for the Algorithm-1 in both T and D-experiments. This is an indication of a good clustering algorithm.

The correlation between the number of documents assigned to a cluster seed theoretically (which is proportional to the cluster seed power) and experimentally (which is determined by the execution of the algorithms) is also given in Table 4.7 and Table 4.8 for the experiments. In case of the Algorithm-1, there is a positive correlation with considerable magnitude for both experiments. However, this is not valid for the Algorithm-2 as expected. This is because, in the Algorithm-2, the cluster seeds are not the maxor criteria for the assignment of documents to clusters.

4.2.4. Similarity of the Algorithms

For the similarity considerations of the two algorithms, we should refer to Figure 4.1 and 4.2 for the T and D-experiments, respectively. In the experiments, for each D matrix corresponding to a T and D-experiment, the values for the Rand similarity coefficient (c) and the Goodman-Kruskal's prediction powers (λ, λ_1, λ_2) were calculated where the meanings of the notation used are :

λ : mutual prediction power of the algorithms;

λ_1 : predicting the partitioning generated by the Algorithm-1 by using the partitioning generated by the Algorithm-2;

λ_2 : predicting the partitioning generated by the Algorithm-2 by using the partitioning generated by the Algorithm-1.

In the case of the T-experiments (Figure 4.1), the c values remain high. As the number of terms (NOTERM) increases, the c values remain high. As the number of terms (NOTERM) increases, the c values decrease for a while then they start to increase. In the case of the D-experiments (Figure 5.2), the c values show some fluctuations, then they also start to increase. All of the c values are very high, to show considerable similarity between the two clustering algorithms.

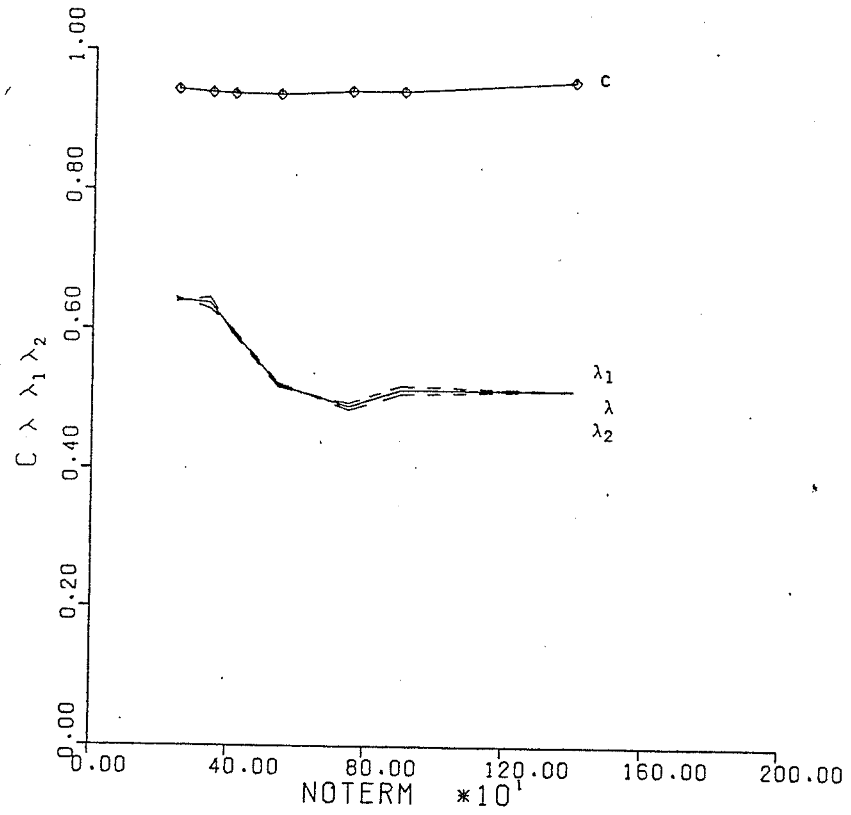


Figure 4.1 - Similarity in the T-Experiments

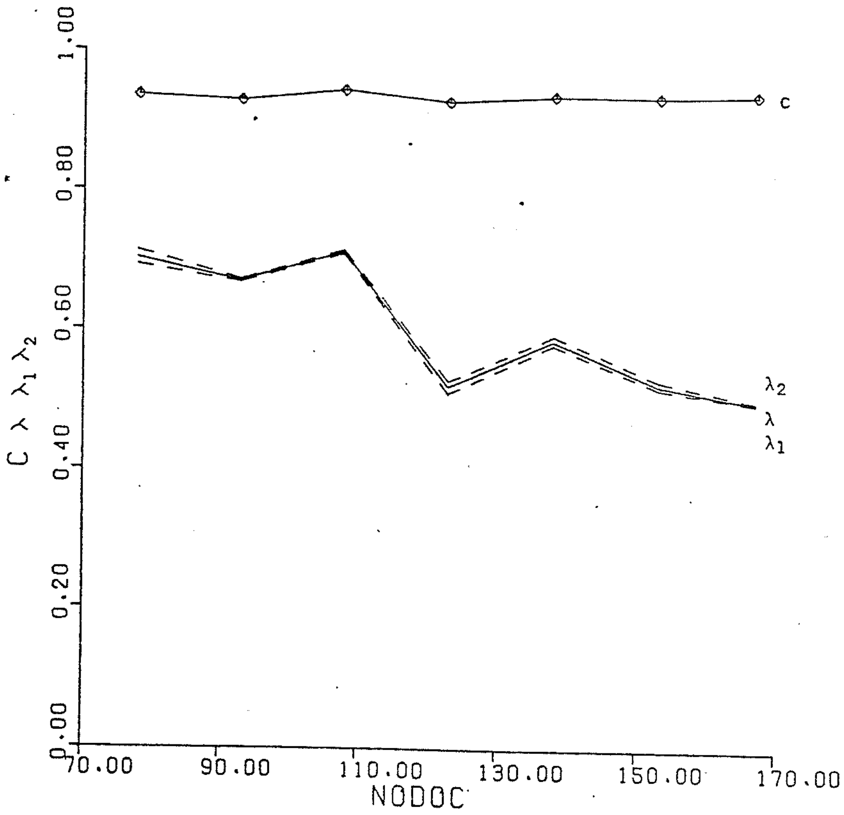


Figure 4.2 - Similarity in the D-Experiments

The mutual predictivity power, λ, of the algorithms decreases as the number of terms (number of documents) increases for the T (D)-experiments. In the case of the T-experiments, after a certain NOTERM value, it stabilizes. However, for the D-experiments this is not true, it shows considerable fluctuations. The λ_1 and λ_2 values do not show considerable difference.

The fluctuations in the c and the λ values are sometimes similar within an experiment. The λ values of the D-experiments are higher than those of the T-experiments. However, the fluctuations in the λ values reach a steady state in the case of the T-experiments. Therefore, we can state that the clustering behavior of the algorithms, with respect to each other, is more sensitive to collection growth than to different term assignment strategies. In general, the λ values are not so bad (see the example calculation given previously in in this chapter). As a result, we can conclude that considerable similarity is observed between the partitions generated by the algrotihms.

4.2.5. Stability of the Algorithms

In the stability analysis of the T(D)-experiments, the results of the $T_i(D_i)$ experiments ($i = 1,\ldots,7$) are compared with the results of the T4(D4) experiment, which correspond to the base D matrix. For the comparison, again, the Rand's similarity coefficients (c_1, c_2) and the Goodman-Kruskal's predictivity power coefficients (λ_1, λ_2) were used. It is assumed that, the higher the stability then higher would be the values of these coefficients, where the meanings of the notations used are :

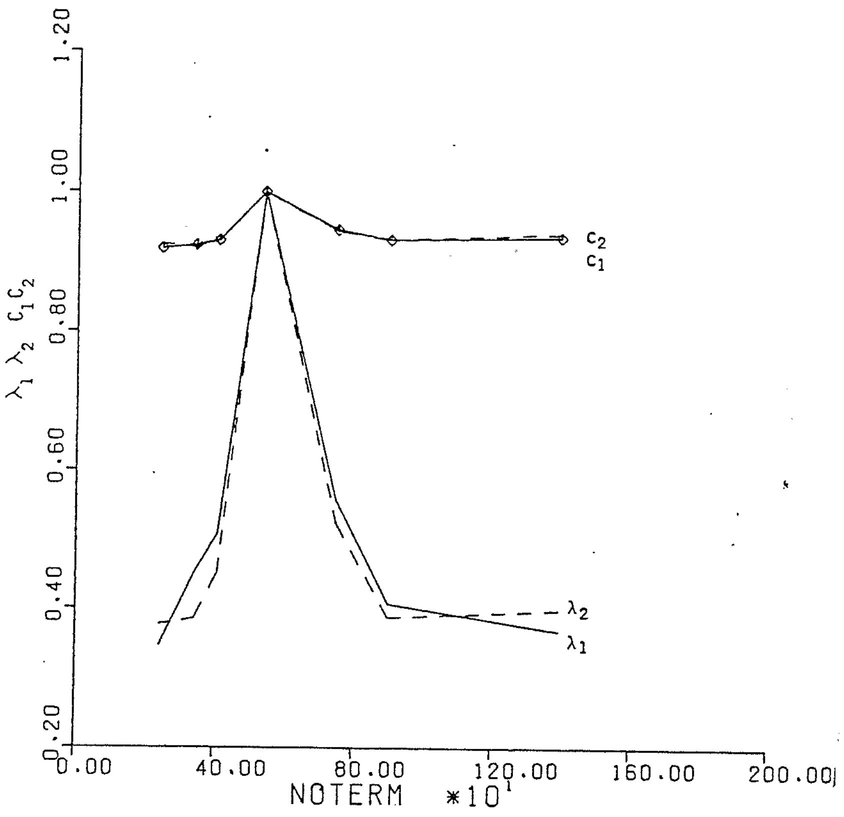


Figure 4.3 - Stability in the T-Experiments

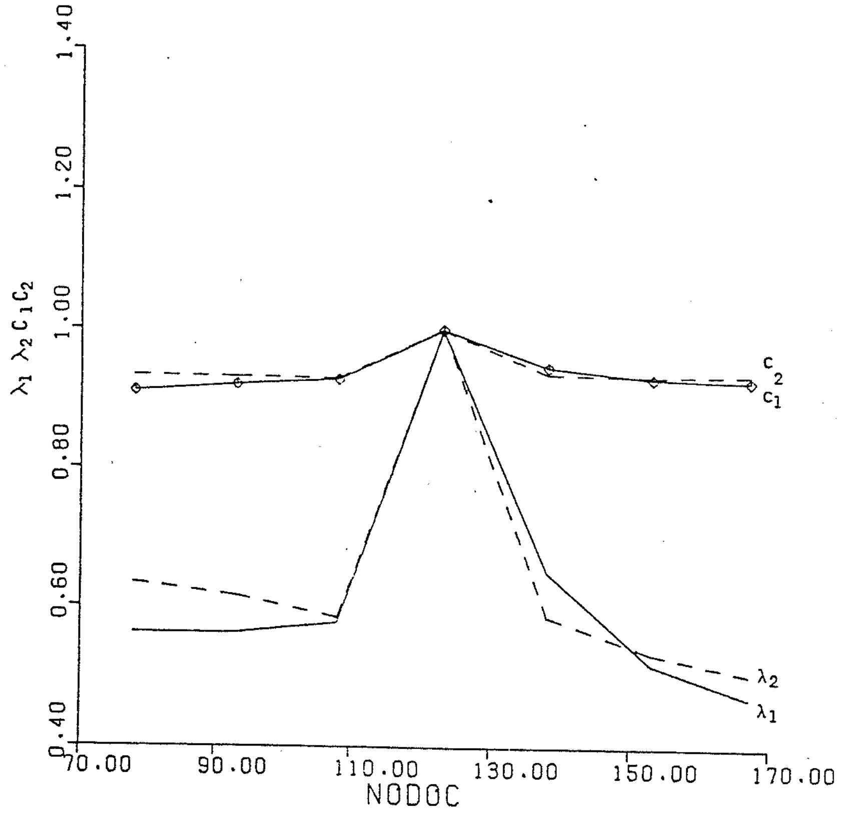


Figure 4.4 - Stability in the D-experiments

$c_1(c_2)$: The Rand's similarity coefficient for the Algorithm-1(2)-notice that, in this case the Rand's coefficient is used for stability measurement-;

$\lambda_1(\lambda_2)$: The Goodman-Kruskal's mutual predictivity coefficient between a perturbation matrix and the base matrix of the experiment for the Algorithm-1(2).

The results of the experiments are given in Figure-4.3 and Figure-4.4 respectively for the T and D experiments. In these figures bold lines are used for the Algorithm-1 and dashed lines are used for the Algorithm-2. In the T-experiments, the results of the Algorithm-1 are always compatible or better than the results of the Algorithm-2 according to the Rand's and the Goodman-Kruskal coefficient, respectively. This situation reverses for the extreme values of the number of terms.

In the D-experiments, the results of the Algorithm-1 and the Algorithm-2 are compatible. The Algorithm-1 behaves better for medium number of documents.

The stability observed in the D-experiments, in terms of the Goodman-Kruskal predictivity power, is higher with respect to the same metric observed in the T-experiments.

4.3. BASIC RELATIONSHIPS OBSERVED IN CONNECTION WITH THE EXPERIMENTS

There are some interesting empirical observations that can be made on the experiments:the number of terms used for the description of the documents (n), and in connection with this the average number of terms per document (depth of indexing, x_d), and the average number of documents per term (term generality, t_g) are important factors for determining the number of clusters in a collection (Notice that x_d and t_g are not independent of each other, a change in one will also affect the other). These empirical observations indicate basic relationships which can be formulated as follows :

$$n_c = \frac{m}{t_g} \qquad (1)$$

$$n_c = \frac{n}{x_d} \qquad (2)$$

$$d_c = 1/\delta = t_g \qquad (3)$$

$$d_c' = 1/\delta' = x_d \qquad (4)$$

where, d_c' is the average number of terms per term cluster. If we substitude value of $t_g(=t/n)$ or $x_d(=t/m)$ in the above formulas for n_c, the number of clusters within a collection can be determined in terms of m, n, and t as follows :

$$n_c = \frac{m*n}{t} \qquad (5)$$

If one substitudes t/t_g and t/x_d, respectively, for n and m in the above formula n_c can be obtained in terms of t, x_d, and t_g :

$$n_c = \frac{t}{x_d*t_g} \qquad (6)$$

It can be checked whether the equations (5) and (6) are consistent with the analytical formula given for n_c, which is the following :

$$n_c \doteq \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} \alpha_i \beta_j$$

In the above formula

$$\alpha_i = \frac{1}{\sum_{k=1}^{n} d_{ik}}, \qquad \beta_j = \frac{1}{\sum_{k=1}^{m} d_{kj}}$$

Therefore,

$$n_c = \sum_{i=1}^{m} \sum_{j=1}^{n} \left[d_{ij} * \frac{1}{\sum_{k=1}^{n} d_{ik}} * \frac{1}{\sum_{k=1}^{m} d_{kj}} \right]$$

The summations $\sum_{k=1}^{n} d_{ik}$ and $\sum_{k=1}^{m} d_{kj}$ are the depth of indexing for document-i(x_{d_i}) and the generality of term-j (t_{g_j}), respectively. By this new notation n_c can be rewritten as follows :

$$n_c = \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{d_{ij}}{x_{d_i} * t_{g_j}} \tag{7}$$

The equation for n_c due to empirical observation (equation 6) can be written as follows :

$$n_c = \frac{t}{x_d * t_g} = \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{d_{ij}}{x_d * t_g} \tag{8}$$

where x_d and t_g can also be written as $\frac{1}{m} \sum_{i=1}^{m} x_{d_i}$ and $\frac{1}{n} \sum_{j=1}^{n} t_{g_j}$, respectively. Therefore, if individual x_{d_i} and t_{g_j}, or $x_{d_i} * t_{g_j}$ are considerably close to x_d, t_g, or $x_d * t_g$, respectively; then one would expect that the above formulas for n_c(i.e., the analytical formula given by

(7) and the empirical formula given by (8)) would give considerably close values to each other. In a document retrieval environment, the above constraints will usually be observed because number of terms per document will be nearly equal to each other among the documents. In other words, the density of distribution of 1's in the binary D matrix will be uniform. That is why, the basic relationships are observed in all of the experiments associated with different D matrices. The validity of equation 6 also validates the equation 1 through 5, since equation 6 is interrelated with them.

The empirical observations, formulated by the equations 1 through 6, will be referred to as "indexing-clustering association" basic relationships. These basic relationships are at least valid for the clustering algorithms which employ the decoupling coefficient concept for the determination of number of clusters (n_c) within a collection.

4.4. SUMMARY AND CONCLUSIONS FOR THE EXPERIMENTAL ANALYSES

In this chapter, the stability and similarity analyses of the two partitioning type clustering algorithms have been presented. The findings of a set of experiments which gave the opportunity of comparing the two algorithms in various ways have also been reported. This chapter also includes a brief illustration of the metrics to be used in the analysis.

In the similarity/stability analysis, two different metrics were used. They were the Rand's similarity coefficient and the Goodman-Kruskal's predictivity power concept. A high value of similarity was observed with respect to the Rand's coefficient in both algorithms. The similarity of the algorithms in the T-experiments according to

Goodman-Kruskal measure, stabilizes after some number of terms if one increases the number of terms used for the description of the D matrix. However, the similarity of the two algorithms, according to Goodman-Kruskal's metric, during the D-experiments decreases significantly if the number of documents is increased. The stability of the Algorithm-1 in the T-experiments was slightly higher if the range of terms was medium. In the D-experiments, the stability of the Algorithm-1 is compatible with that of the Algorithm-2 for medium number of documents, this situation reverses in the extremes.

Based on the λ values, it was observed that, collection growth is more effective on the clustering patterns generated by the algorithms with respect to term assignment strategies (in different term assignment strategies the collection size is kept fixed). The sharp changes in the λ values in the D-experiments for both the similarity and stability analyses, are reasonable compared to those in the T-experiments. In the D-experiments, the rate of change in the number of documents and the number of terms are the same, which is not realistic. In an IR system, the rate of change in the number of terms will be considerably less than that of the number of documents |44, p.207|. This means that we can expect better stability in our algorithms when the collection size changes.

In the experiments basic relationships, called "indexing-clustering association" basic relationships, are observed. These relationships state that the number of terms used for the description of the documents and in connection with this, the depth of indexing (x_d), and the term generality (t_g) are the basic determinants of the number of clusters within a collection (n_c), the average number of documents in the document clusters (d_c), and the average number of terms in the term clusters (d'_c). Rigorously, these relationships are given as follows : $n_c = m/t_g$, $n_c = n/x_d$, $d_c = t_g$, $d'_c = x_d$. These relationships also imply the effect of total term assignments on the number of clusters, which is $n_c = m*n/t = t/x_d*t_g$. These are interesting empirical observations which did not appear in the literature previously.

These experiments show the validity of the cover coefficient concept and its usage in clustering. The seed selection process was observed to be effective. After observing the close similarity of the algorithms according to the Rand's coefficient and better stability of the Algorithm-1, especially in the medium range of terms and documents, one can say that the "cover coefficient" outperforms the "similarity coefficient" within the context of the algorithms presented. This is not only because of similarity and stability, but also because of lower complexity and shorter execution time of the single-pass algorithm. The single-pass algorithm will be used in the integrated fact and document retrieval system to be described in Chapter 6 of the thesis |16, 71, 75|.

5. TEXT RETRIEVAL WITH THE RAP DATABASE MACHINE

In this chapter a solution to the text retrieval problem will be proposed based upon the RAP database machine, or more specifically with RAP.3 database machine, which employs the cellular associative approach. In this chapter the RAP.3 database machine, text string and physical data structure formats for text retrieval operations in RAP.3, syntax and semantics of the new RAP.3 instructions, realization of these RAP.3 instructions, implementation of the typical text retrieval operations with RAP.3 programs incorporating the new text retrieval instructions will be presented. Performance of RAP.3 in text retrieval instructions is also presented |11,71|.

5.1. A CHRONOLOGICAL OVERVIEW OF THE RAP DATABASE MACHINES

The development of the RAP database machine has started at 1976 at the University of Toronto. RAP.1 and RAP.2 were the prototypes developed |65,66,96|. This was followed by RAP.3 prototype |62,63,69|. The RAP.3 system was subsequently enhanced for RAM memory at the Intel Corporation in Phoenix, USA |73,74|.

Originally, the RAP database machine is designed as a backend processor for a general purpose computer. In this configuration, general purpose computer (GPC) provides RAP's database contents and compiled programs and receives the results of a user's request returned by RAP.

The original design of RAP is composed of a controller, an arithmetic set function unit, and a parallel organization of cells. Each cell has a dedicated processor and one track of a rotating or circulating memory device. The set function unit is used to combine the scalar aggregate results of the cells. The controller is responsible for overall coordination. Figure 5.1 shows this organization.

This organization directly supports the relational model. Each cell is devoted to storing a relation and several cells store all the tuples of a large relation. If the relation is too large for the cell storage, then the processing is indirect since the necessary data must first be transferred from the main database store. The performance of RAP in such a configuration is analyzed and very satisfactory results have been observed |68,95|. A performance evaluation of RAP with respect to a general purpose computer also showed the superiority of RAP |7|.

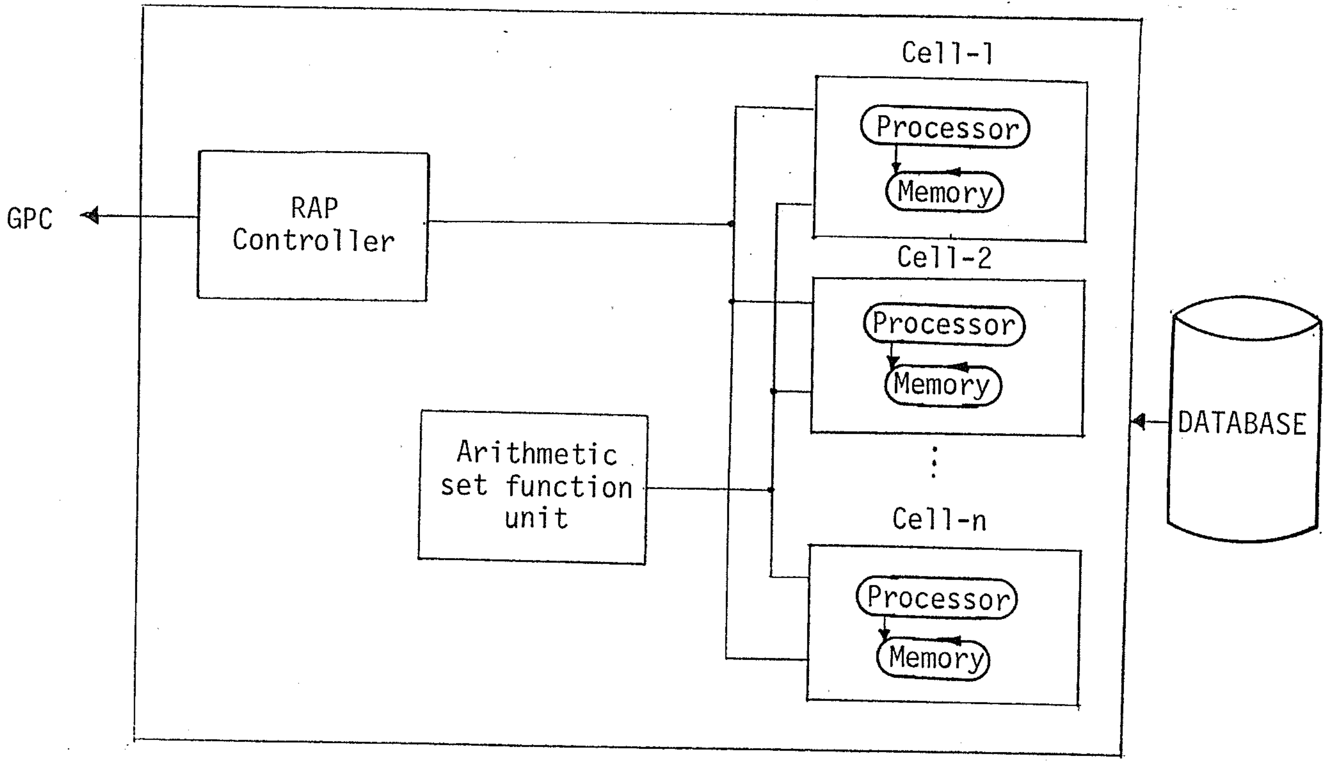


Figure 5.1 - Architecture of the RAP processor.

The read mechanism of the cell processor brings the tuples of the relation one by one to the cell buffer memory. All selection, set function computation, replacement and arithmetic update operations on tuples take place in the buffer. Then the processed tuple is written back to its place in the cell memory.

In RAP.1 and RAP.2, each cell has many comparators, each of which independently examines a particular attribute of a relation, which is specified in the qualification part of a RAP assembler instruction. Each comparator unit compares tuple value of an attribute with a specified search value.

A multi-microprocessor approach for the cell architecture restructured the RAP cell in yet another array of independently operating subcells, where each subcell comprises a microprocessor with necessary peripheral chips |62,63,74|. The tuples of a relation are loaded into the memory buffers related with each subcell and a data move strategy is incorporated to allow for parallel processing of consecutive tuples of a "RAP relation". Here, the phrase "RAP relation" is used. Since a RAP relation is a normalized relation augmented with a set of mark attributes. Accordingly, each RAP tuple instance has a set of mark (tag) bits ahead of n-tuple values. A RAP tuple can be seen in Figure 5.2. Figure 5.3 shows the structure of a RAP cell. This is the cell configuration for the latest version of RAP, which is RAP.3.

The circulating memory (CM) of each cell of RAP.3 also has a different structure than the previous RAP designs. The CM is chosen as bit parallel word serial organization. This matches the data access port size to the subcell microprocessor data bus width; furthermore, incorporates rather slow magnetic bubble memories (MBM), or high density RAM's in a parallel organization |63|.

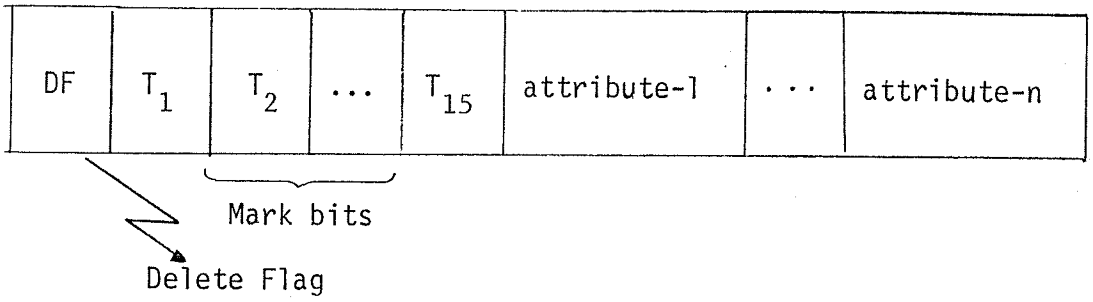


Figure 5.2 - RAP tuple structure

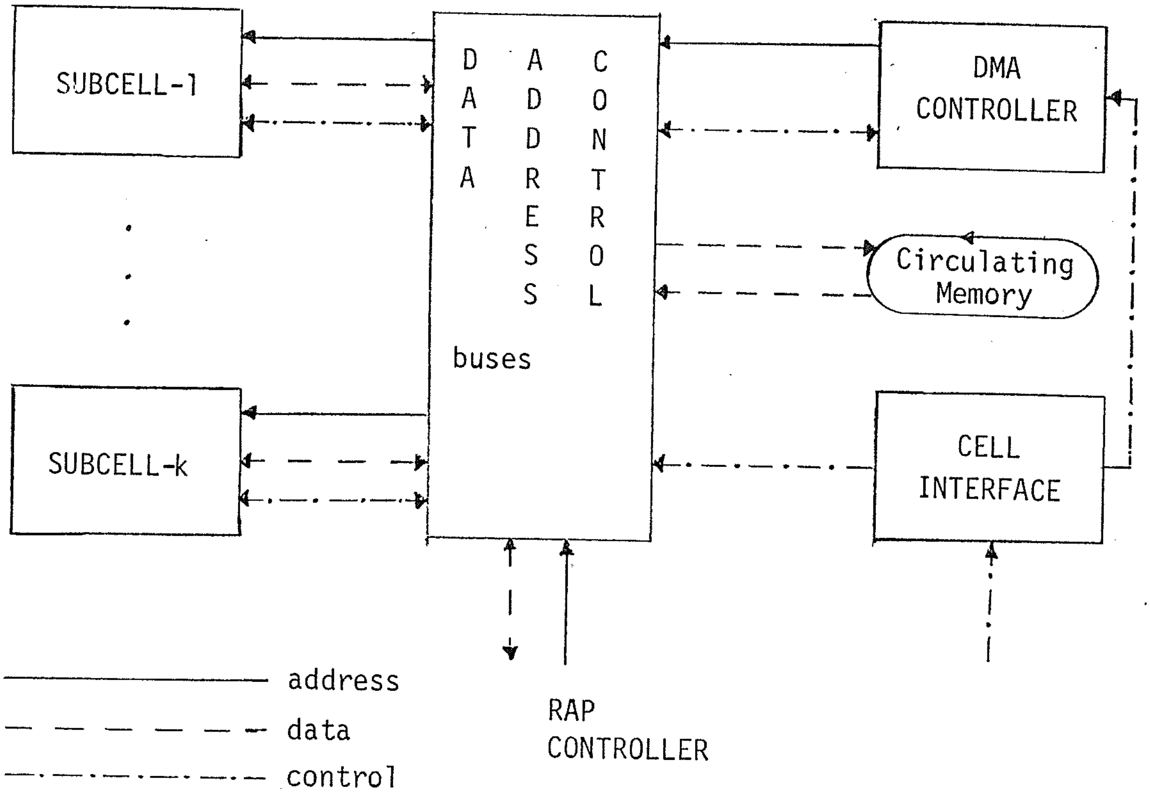


Figure 5.3 - Structure of a RAP.3 cell

In RAP.3, each cell is made of k parallel subcells each having the full functionality of a RAP cell. Each subcell has a microprocessor (INTEL/8086), enough local RAM storage to hold a tuple, RAP instruction microcode, and data parameters. The DMA hardware reads bit parallel and word serial tuple data and concurrently writes back the processed data (tuples). Each subcell is in the wait state and it is reinitiated when a tuple is loaded into its buffer. A subcell executes the "query routine", i.e., 8086 firmware corresponding to a RAP instruction, on the tuple in its RAM. The tuples are loaded (i.e., tuples are read in) and unloaded (i.e., tuples are written back onto the cell memory) in a cyclic fashion. Due to this, each subcell has a fixed amount of time to process its tuple to stay in synchronization.

The time available for a subcell to process its tuple is given by the following expression :

$$T_{PR} = \text{time allowed to process a tuple} = f(k, T_{LS}) = (k-2) * T_{LS}$$

where

$$T_{LS} = \text{time to load/store a tuple through DMA} = f(\text{Tuple Length, Memory Data Rate})$$

The timing analysis of RAP.3 cell operations is given in Appendix-C.

The RAP.3 prototype of Version-I utilized RAMs as the cell memory. However, the design was made to allow any type of memory for CM. The final design of RAP is tuned for RAM type CM |73|. This approach provided the following : the cells can process multirelations; new hardware algorithms are developed for the operation of join, projection, and transaction processing that exploit sorting; subcell processing is bypassed for the qualifications made from mark bits (this kind of qualification processing is done on the memory bus); the cell processing model is changed from I/O bound model to subcell processing bound one |73|.

In the latest RAP.3 design, the processing time for a tuple, T_p is given by the following expression :

$$T_p \geq (W * T_{WLS} + \rho * W' * T_{WLS})$$

and generally $T_p < T_{PR}$ (i.e., of the old scheme). In the new cell processing model, there is no unutilized portion of tuple I/O. The meaning of the above T_p expression is as follows :

a) The qualifications which involve tests on marked and/or unmarked bits are resolved directly on the memory bus, bypassing the subcells. This means that only one word is accessed per tuple (in time T_{WLS}).

b) Only required words of a tuple (whose number is W) are accessed from the cell memory in contrast to the input of the entire tuple. These words are needed in an instruction specification and qualification.

c) Only ρ portion of the tuples read need to be written back to the cell memory. During this, only the updated words (whose number is W') are written back. This is again a contrast to the non-selective rewrite of the previous RAP designs including RAP.3/Version-I.

5.2. RAP ASSEMBLER LANGUAGE

The RAP assembler language also known as the RAP DBMS Assembler is relationally complete, i.e., it has the whole capabilities of relational algebra |70|. The RAP DBMS Assembler has the following general syntax :

<label><opcode><specification><qualification><parameter>

The label is a symbolic instruction address, the opcode is the operation to be performed, and a specification has the following format :

<relation>(<attribute-1>, <attribute-2>,...,<attribute-s>)

A qualification is a Boolean expression (possibly null) of simple conditions Q_i where Q_i is one of the following :

a) MKED ($<t_c>$) denoting any combination of true (i.e., set) mark bits.

b) UNMKED ($<t_c>$) denoting any combination of false (i.e., reset) mark bits.

c) <attribute><comparator><operand>

where <comparator> is one of the relational operators of $=$, $\neq$, $\leq$, $>$, $\geq$, $<$ and <operand> is a numeric or literal constant, or another attribute name. <parameter> specifies the second operands in arithmetic operations, the RAP registers, or the source relation with its qualification in the binary operation of implicit join (CROSS-MARK instruction of RAP).

The operation of each instruction is iterative due to its associative nature. An instruction, directly works on the memory contents, evaluates the qualification stated on the tuple contents, and executes the opcode on the tuple contents if the qualification holds for the tuple contents. A given instruction is terminated after all tuples of the <relation> are processed in parallel.

The RAP DBMS Assembler is more powerful than the relational algebra. This comes from the fact that it does not generate intermediate relations explicitly. For example, a binary relational algebra operator (RAO) operating on relatinos R and S would require a new relation T to store the results as indicated by T←R RAO S. The same operation in RAP will be accomplished as R←R RAO S requiring no new relation. Additionally, the old contents of the R relation are not destroyed. This is made possibly by the mark bits of the RAP relations.

A brief summary of RAP instructions can be seen in Appendix-D. A detailed description of the instructions can be found in |102|. For an abstract model of the RAP language one may refer to |72,101|. A software emulator of RAP, SERAP, is available on different machines |26|. SERAP provides a good test-bed for the utilization of RAP in different applications, such as text retrieval, and office automation |75|.

5.3. TEXT RETRIEVAL OPERATIONS WITH THE RAP DATABASE MACHINE

In the previous section of this chapter, the RAP database machine has been introduced. In the remainder of the chapter, a solution to the text retrieval problem will be proposed based upon the RAP database machine which employs the cellular associative approach.

5.3.1. Text String and the Physical Data Structure Formats

For text retrieval operations with RAP, a set of related documents constitutes a relation. The query term matching operation on this relation can be either done on a limited portion of the relation or on its entirety. The limitation of the search will be done by document clustering which is to be described in the subsequent chapter of the thesis.

The integration of the text retrieval capability to RAP will give emphasis to the following points :

a) Domain type must support literals.

b) Domain length must be variable for literal types where literal domains will hold the document text.

c) Efficient string search pirimitives must be incorporated.

d) Sentence structure and adjacency must be recognized.

e) A way of processing unformated data with formatted structures must be found.

f. Resolution capability must be added to the operation supported in (c) and (d).

The requirement of (a) has already been provided by RAP. The change made in RAP allows a literal data type to be as long as a tuple itself (i.e., resulting in a unary relation). RAP also provides variable length tuple to store the unstructured document attribute. The requirement of (c) will be provided by fast pattern matching algorithms. Text representation within the RAP relation tuples will satisfy the requirements of the items (d) and (e). The (query) resolution requirement of item (f) will be provided by the RAP assembler language.

The following is the format specification for text representation in the RAP.3 relations |16,71,75|.

a) Each tuple may contain one (or more) complete sentences.

b) The first tuple of each document starts with a blank.

c) As with standard typing rules, at least one blank should follow a punctuation mark.

d) At the end of a sentence an End of Sentence (EOS) marker is placed following the period. EOS must be followed with at least one blank.

e) The last tuple of the document terminates with an End of Document (EOD) character.

The text retrieval operations with RAP introduce some constraints on the distribution of a relation's tuples on the RAP cells (in DBMS operations there are no constraints).

a) The tuples of a specified document should come one after the other in a cell memory (notice that a document may require more than one tuple).

b) If the m'th cell memory ends with the n'th tuple of a document, the memory of the m+1'th cell should start with the n+1'th tuple of the same document.

In Figure 5.4, the layout of a RAP.3 tuple for text retrieval operations is shown. According to this tuple layout, documents are mapped into the D_1 attribute of several consecutive tuples. The D_2 and D_3 attributes of the integer domain are needed for the internal use of the context sensitive text retrieval operations. D_4 is the key attribute which is the unique document identified (DOCID). D_5 through D_n are the attributes reserved for formatted data if they are needed for the convenience and efficiency of processing.

As can be realized from the text mapping scheme, an important problem of text retrieval with the formatted structure is the need to maintain text contiguity both in string searches and context resolutions. This problem has been solved, to a great extent, by the variable length

D F	Mark Bits	D_1 Long text retrieval attribute (variable length)	D_2	D_3	D_4	D_5	...	D_n

Figure 5.4 - The layout of a RAP.3 tuple for text retrieval

tuple (specifically, variable length text attribute) feature of RAP.3. By this feature, some context resolution problems such as splitting of a text word between tuples (called term overflow) and passing overflow indication to subsequent tuples to finalize the search are inherently solved. However, there still remains the problem of context resolution for the text retrieval operations of the type A ... B and <A,B>n (see Appendix-A for the meaning of the operations). This is because the specified context might not be satisfied within a tuple. The necessary information for context resolution is passed from tuple to tuple by a feature referred to as "link passing". The problem of context resolution is not different from other text retrieval systems and in fact, it is easier in RAP.3 due to the physical data as well as command structures.

In the previous version of mapping text into RAP tuples, the sentences and/or words could be split between tuples. This required overflow resolution even in simple term matching that required no context specification. The necessary query resolution was achieved by using an instruction called MATCH-OVERFLOW in the previous RAP based text retrieval implementation |11|. The new variable length tuple feature of RAP has introduced about 50% efficiency over the previous implementation by removing the requirement of overflow resolution in simple term matching.

5.3.2. The New RAP Instructions for Information Retrieval

In order to implement text retrieval operations, some new RAP instructions have been introduced |11,71|. In this section syntax and semantics of these instructions, and the realization of them will be introduced.

The following gives the syntax definition of the new RAP.3 DBMS Assembler commands needed to perform the text search, and link-pass operations.

MATCH (tc) [rel(atr$_1$ {,atr$_2$ {,atr$_3$}}):qual] {[lit]}
MATCH-WS (tc) [rel(atr$_1$ {,atr$_2$}):qual] {[lit]}
MATCH-WWC(tc) [rel(atr$_1$ {,atr$_2$},atr$_3$):qual] {[lit]} {[int]}
LINK-PASS(tc$_1$,tc$_2$) [rel({atr$_1$,} atr$_2$)]

where

- tc, tc$_1$, and tc$_2$ indicate any combination of the available mark bits (there are thirteen mark bits available to the user and tc$_1 \neq$ tc$_2$) and 14'th mark bit is used for overflow indication by the MATCH-WWC (match-within-word-count) instruction.

- rel is the relation name of the documents (cluster file).

- atr$_i$ $1 \leq i \leq 3$ denote attributes, atr$_1$, being of literal type, contains a piece of the text.

- qual is any RAP legal qualification Boolean expression.

- litc is the legal literal constant corresponding to the search pattern.

- int is a positive integer.

- { } indicate options.

The first match instruction searches for the equality of the search pattern, in the text stored in atr_1 and tc marks all the qualifying tuples within the cells storing the document. atr_2 gives the beginning search offset within the text string (i.e., atr_1). If atr_2 is not specified, then search offset is taken as 0. If atr_3 is specified, all occurrences of the search pattern within D_1 are determined and the total number of occurrences is stored in atr_3, within each tuple.

The difference of the second match instruction is that the first occurrence of the second pattern is found within the sentences (WS) of D_1. The entire match should be contained within a sentence. The beginning search position can be indicated by the contents of atr_2. If it does not appear, then it means that the beginning search offset begins at offset 0 of atr_1. The end of the search position is the first EOS character after the beginning search position. After the execution of the instruction on a tuple, atr_2 (if specified) is set as the beginning offset of the next sentence within the tuple.

The third match instruction finds the first occurrence of the search pattern bypassing at most "int" number of words of the text. The value of "int" can be specified explicitly, or implicitly by atr_3.

During execution of the third match instruction, if the search pattern is not found and the search context specified by the word count is not satisfied within a tuple, then t_{14} mark bit of that tuple is set. This specific case is called overflow condition and can occur only during the execution of the MATCH-WWC (match-within-word-count) instruction, since the context of a match is always satisfied within a tuple in the other types of the match instructions. The number of words that should be considered further for search context resolution is put into atr_3.

The LINK-PASS instruction takes one memory cycle to execute and resolves the overflow condition in those tuples marked for overflow. The resolution requires resetting the t_{14} mark bit (which is specified as tc_2 in the instruction syntax) of the overflown tuples, setting tc_1 bits of the subsequent tuples, and passing the unprocessed (i.e., remaining count values) into the predetermined attributes of the subsequent tuples, which is atr_2.

In the other use of the LINK-PASS instruction, when the literal attribute atr_1 is specified, if a tuple is tc_2 marked and if its atr_1 attribute does not contain an end of text character the following tuple(s) of this document will be tc_1 marked and their atr_2 is set to zero. This feature of the LINK-PASS instruction is used to set the search context of the text retrieval operation A...B.

A search pattern specified within a query is a semantic unit by itself which cannot be divided between two sentences. Therefore, it is assumed that the fixed length don't care (FLDC) character specified within a search pattern will not match an End of Sentence character. This semantic property of search patterns prevents the occurrence of an overflow during term matching.

The search pattern of the RAP instructions can also be specified in the RAP register REGU-1 (i.e., user register 1). In this case the literal constant of the match instruction "lit" should be omitted. A RAP register can be initialized by the RSET (register set) instruction:

RSET < reg_1, opd >

which will insert the immediate data, opd, into the register reg_1.

5.3.3. Realization of the New RAP Instructions

During the execution of the match, i.e., the instructions beginning with the keyword MATCH, the search pattern can either be specified explicitly or reside in the REGU-1 register. The RAP registers are 120 characters in length; therefore, 120 is the maximum search pattern length. However, this length is sufficient for practical purposes. For example, in English, the average word length is 8.1 characters |76|.

The formal definition of a search pattern in a RAP program is given in Figure 5.5. A question mark in a search pattern indicates a "don't care" character and n indicates the number of repetetions. An * at either side of the search pattern means that the match can occur anywhere within the text and End of Word (EOW) characters (i.e., a blank at the left hand side and a blank or a punctuation mark at the right hand side) would not be required at the corresponding sides of the text string. The following are typical search patterns :

'AA' ? 'BB'

*'A' ?? 'BB'

*'AA' ?? 'BB' 3? 'C'

The initial character of a subpattern of a search pattern (e.g., the search pattern 'AA'?'BB' contains two subpatterns 'AA' and 'BB', and the initial characters are single quotes) can appear in a subpattern; however, it should be entered twice for one occurrence of it.

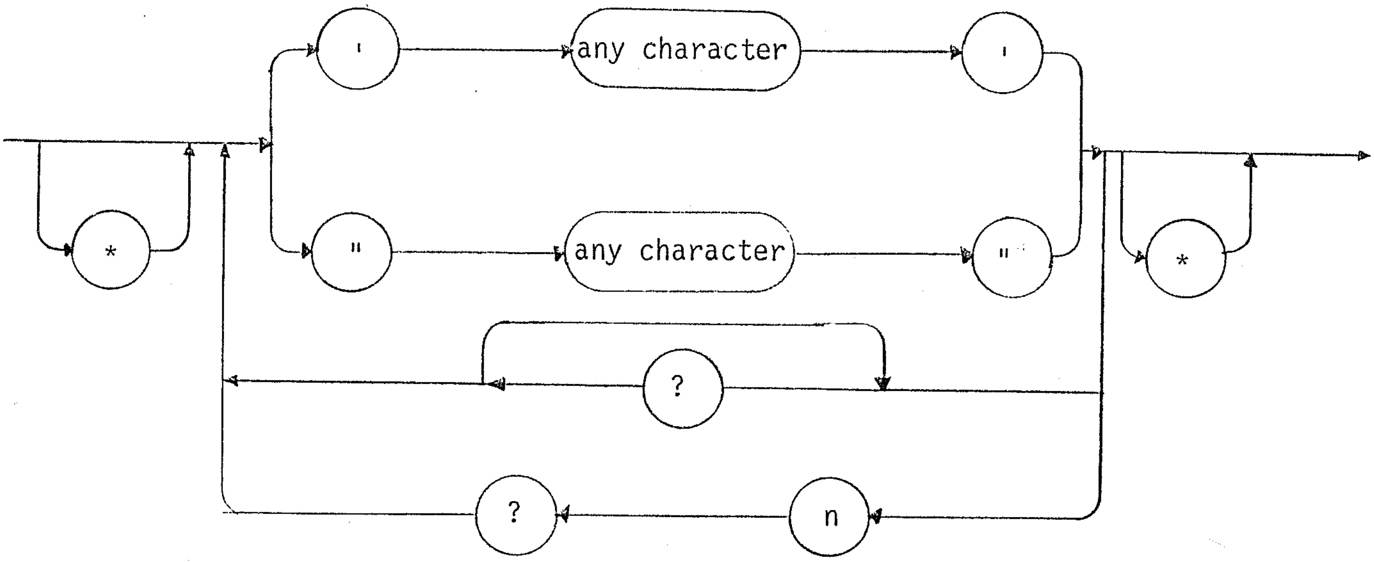


Figure 5.5 - The formal definition of a search pattern in a RAP program

The design of RAP.3 utilizes the modular multimicroprocessor approach and the machine is firmware driven. The query programs written in the RAP DBMS Assembler language are converted into a form called an "instruction-packet" (IP). In this approach, the query independent (fixed) part of the firmware is kept in the subcells and the query dependent part of the firmware is broadcasted by the RAP controller |1,62,63|. Since the current design of RAP utilizes the INTEL/8086 microprocessor, all of the firmware has been written in the machine language of this microprocessor.

The realization of the new RAP instructions requires a very little change in the hardware. This is due to the firmware driven nature of the RAP instructions. This hardware amendment is due to the LINK-PASS instruction. The LINK-PASS instruction is utilized to pass values between the adjacent tuples of the documents. It is previously pointed out that the tuples of a document come one after the other on the cell memories. Since the tuples of a document may span more than one cell, execution of the LINK-PASS instruction may need a communication path between the adjacent RAP cells. To achieve this, after the execution of a match instruction, necessary information for the LINK-PASS instruction, which is obtained from the last tuple of a cell will be sent to the cell interface processor (CIMPU).

The firmware necessary for MATCH, MATCH-WS, MATCH-WWC needs an efficient matching algorithm. Recently two fast text matching methods have been proposed. The first one is due to Knuth, Morris, and Pratt |52|; the second one is due to Boyer and Moore |7|. These two algorithms resemble each other. In the first algorithm, the number of character comparisons needed to match a text of length m with a pattern of length n is in the order of (m+n), while in the second algorithm, it is independent of the pattern length and the number of character comparisons is in the order of (m) or less for a text string of m characters.

In the implementation of the above mentioned match instructions, a modified form of the Boyer and Moore (BM) algorithm has been used, therefore the BM algorithm will be explained briefly.

Assume that we want to find the pattern (search pattern) in a (text) string. The BM algorithm starts by comparing the rightmost character of the pattern with the selected character from the beginning of the string (at the beginning, if the length of the pattern is n, n'th character of the pattern is compared with the n'th character of the string). If a match is found, a left shift will occur, then the character at the left of the rightmost character in the pattern will be considered in the next step. Meanwhile, the pointer to the string will be shifted to the left by 1. This process will continue until a complete match, or a mismatch is encountered. The ingenuity of the BM algorithm comes from the strategy used in the case of mismatches. There are two possibilities in determining the pointer shift (the number of text characters to be skipped) if a mismatch occurs :

a) The mismatched character of the string is searched in the unprocessed part of the search pattern. If it is found, the search pattern will be shifted to the right (called the Δ, shift) for the coincidence of the identical characters of the pattern and the string. If the mismatched string character does not occur in the pattern, move the pattern to the right such that leftmost character of the search pattern comes just to the right of the mismatched string character. The maximum value of Δ_1 is the length of the pattern.

b) If a portion of the pattern is matched in the text string, a search is done to find a repeating occurrence of the matched subpattern in the unprocessed part of the pattern. If it is found, the pattern will be shifted to the right (called Δ_2 shift) to coincide this subpattern with the matching portion of the text string.

In the meantime, the string pointer will be shifted to the right accordingly.

In determining the shift amound (Δ_1 or Δ_2) the criterion is to achieve the largest possible shift for the text string, so that one will reach the end of the string as fast as possible. For this purpose the maximum of the Δ_1, Δ_2 values will be chosen in determining the shift amount. The values of Δ_1 and Δ_2 are found and put into a tabular form by a prior analysis of the search pattern |7|.

This algorithm will work most effectively, if the number of matching characters between text string and search pattern is low. Also similarly, if pattern contains fewer repeating subpattern then the algorithm will behave more effecively.

An example for the BM algorithm is given in the following |7|. The example tries to find the pattern "AT THAT" in the string "...WHICH..." as follows

```
pat :       AT-THAT
string : ...WHICH-FINALLY-HALTS.--AT-THAT-POINT...
                  ↑
```

Since "F" does not occur in "pat", we can move "pat" to the right as much as the "pat" length (which is 7). Then we have :

```
pat :              AT-THAT
string : ...WHICH-FINALLY-HALTS.--AT-THAT-POINT...
                         ↑
```

In this case, "pat" is moved to the right to coincide with the hyphen.

```
pat :                 AT-THAT
string : ...WHICH-FINALLY-HALTS.--AT-THAT-POINT...
                             ↑
```

Since L does not occur in "pat" we can move "pat" to the right such that the leftmost character of "pat" comes just after the mismatched character L.

```
pat :                        AT-THAT
string : ...WHICH FINALLY-HALTS.--AT-THAT-POINT...
                                 ↑
```

In this case, mismatch occurs at H and hyphen. In order to shift "pat" to the right we can use Δ_1 (to coincide hyphens of "pat" and "string") or Δ_2 (to coincide "AT" at the leftmost of "pat" with the "AT" of "string" which appears between the two hyphens). Since Δ_2 provides more shift, it is selected.

```
pat :                             AT-THAT
string : ...WHICH-FINALLY-HALTS.--AT-THAT-POINT...
                                        ↑
```

Comparison between the elements of "pat" and "string" starts at the position indicated by the up-ward pointing arrow where an exact match terminates the algorithm.

The BM pattern matching algorithm and the algorithm due to Knuth et.al. require the preprocessing of the search pattern. For example, in the BM algorithm, this preprocessing is done to find Δ_1 and Δ_2 values. In a research done by Horspool |48| it is seen that the effect of using Δ_2 is negligible, or even one may achieve better performance if Δ_2 is ignored. During the implementation of the match instructions, for string searching, a modified form of the BM algorithm is used and it is summarized in the following (in this algorithm, the table "delta12" is the same as Δ_1 except that the "delta12(lastch)" is equal to Δ_2(pattern) :

```
delta12(*) = patlen; /*initialize whole array*/
do  j = 1 to patlen-1; /*perform the preprocessing*/
    delta12 (pat(j)) = patlen-j;
end;
lastch = pat(patlen);
i = patlen ;
do  while  i ≤ stringlen ;
    ch = string(i);
    if ch = lastch then
      if string (i-patlen+1...i) = pat then
        return i-patlen+1;
    i = i+delta12(ch);
end;
return 0;
```

5.3.4. Implementation of the Text Retrieval Operations with the New RAP Instructions

In this section, some algorithms are provided for the implementation of the typical text retrieval operations with the RAP database machine. In the RAP.3 text retrieval system, a great proportion of the context dependent query resolution takes place in the operation of the new text retrieval commands whose basic task is term matching. As seen in the semantics of the instructions described in the previous subsection, sentence structure and adjacency are recognized as an integral part of the operations. Such resolution operations constitute a totally separate task in most other text retrieval systems. For the execution of the context sensitive text retrieval operations of the more complex nature; however, more sophisticated resolutions are required. These operations are :

A and B in sentence	(specified context)
<A.n.B>	(directed proximity)
<A,B>n	(undirected proximity)
A...B	(immediate adjacency or variable number of words in between)
A * B	(variable length don't care)

The resolution for these operations require that small RAP.3 programs be written using both the DBMS and IR instructions of the RAP.3 language. In other words, these resolutions are embedded in the logic of RAP.3 programs. To aid the user, each of these programs are made general purpose full text retrieval macros running under the macro processor RAPMAC |71|. Appendix-E provides the properties of the macro processor RAPMAC, and a listing of the text retrieval macros for the typical text retrieval operations.

In the following, the algorithms for the implementation of the selected text retrieval operations are given :

Search for A and B in sentence :

In this text retrieval operation, a tuple containing the first term (i.e., A) may be considered more than once, since for the current occurrence of A the second term may not occur in the specified word proximity. In that case, the unprocessed portion of the tuple will be considered for the occurrence of A again and if it is found the tuple will be re-examined for the occurrence of the second term within the same sentence.

1) Mark all the tuples of the text relation for an "A" match.

2) Set the search offset for the term "A" to zero in all the tuples.

3) Test if there are tuples to be considered for a pending "A" match, if not, go to step 11.

4) Reset the mark bits which were set in a previous "A" match operation.

5) Perform a match operation for term "A" on the tuples which may be eligible. If there is no match, go to step 11.

6) Discard the unqualified tuples at the end of match operation, so that they do not take part in the possible re-execution of step 5.

7) Perform a "match-within-sentence" operation for the second term "B" on the tuples which are selected in step 5.

8) Test if any "B" match exists, if not go to step 3 (note that in a tuple, there might be more than one sentence and any of these sentences may contain the term "A").

9) Mark all tuples of the qualified documents and do not process them for an "A" match again.

10) Go to step 3.

11) Exit.

Search for <A.n.B>

This algorithm works like the previous one, but in this case the proximity of words is specified by the word count "n".

1) Mark all tuples of the text relation for an "A" match.

2) Set the search offset for the term "A" to zero in all tuples.

3) Test if there are tuples to be considered for a pending "A" match, if none, go to step 15.

4) If there are tuples marked for an "A" match, reset their mark bits.

5) Perform a match operation for term "A" on the tuples which may be eligible. If there is no match, go to step 15.

6) Discard the unqualified tuples at the end of the match operation, so that they do not take part in the possible re-executions of step 3.

7) Perform a "match within word count" search for the second term "B" on the tuples which were eligible at the end of step 5. The search offset begins right after the point of "A" match. If the match operation is not satisfied within the tuple, the value of remaining word distance to be considered in the next consecutive tuple is placed in a numeric attribute specified in the instruction.

8) Test if any "B" match exists, if not go to step 10.

9) Mark all tuples of the qualified documents. Reset the overflow indicators in the tuples of qualified documents and do not consider the tuples of these documents for an "A" match again.

10) Test if there is any tuple waiting for overflow resolution for a "B" match, if none, go to step 3.

11) Link-pass the necessary overflow resolution information from the tuples which contain a "B" overflow, then perform an overflow resolution for "B" on the tuples which are selected due to this link-pass instruction, by using the match_within_word_ count instruction.

12) Reset the mark bits on the tuples selected (marked) for overflow resolution.

13) Test if there are qualified tuples after overflow resolution, if so go to step 9 (notice that there might be overflow during an overflow resolution, this is checked in step 9).

14) Go to step 10.

15) Exit.

Search for <A,B>n

This operation is performed as two operations of the form <A.n.B> and <B.n.A> which are executed serially.

Search for A??B

The fixed length don't care is taken care of by the workings of the MATCH instruction itself. Search for "A" can be replaced by the fixed length search pattern "A??B".

Search for A...B

1) *Perform an "A" match search on all tuples.*

2) *Test if there is an "A" match, if none, go to step 7.*

3) *Mark all the tuples of the document following the first tuple which contain "A" (This step is mapped into a pair of link-pass instructions in the corresponding RAP program). The link-pass instruction will set the search offset of the following match operation. The search offset of the first tuple of a document containing an "A" will not be touched, but in all of the consecutive tuples, it will be set to zero.*

4) *Perform a "B" match search on the tuples marked in the previous step.*

5) *Test if there is a qualified tuple for the "B" match, if none, go to step 7.*

6) *Mark all the tuples of the qualified documents*

7) *Exit.*

Search for A * B

This search operation uses the same algorithm employed for A...B except for the fact that the search pattern should contain an * both at the right of "A" and at the left of "B", where * stands for don't care.

5.4. PERFORMANCE EVALUATION OF RAP IN TEXT RETRIEVAL ENVIRONMENT

The text retrieval instructions and operations are first realized on RAP.3 Version-I, then realized on RAP.3 Version-II. Therefore the performance evaluation will be given for these two versions.(It was mentioned in Section 5.1 that, RAP.3 Version-I and Version-II are I/O bound and processor bound, respectively.)

5.4.1. Performance of RAP.3 Version-I in Text Retrieval

It is observed that the LINK-PASS instruction does not require an investigation, since this instruction will be executed within time allowed without overloading the system in one RAP circulation. In estimating the RAP behavior in the new match instructions, the analytical model created in an earlier study is assumed |62,63|. The part of the model, which is used in the thesis is also given in Appendix C.

As mentioned previously, RAP.3 firmware driven and for the implementation the INTEL/8086 microprocessor has been used |73|. The INTEL/8086 assembly language programs written for the match instructions are analyzed to find the execution time of the instructions for a tuple (the same firmware routines are used in both of the RAP.3 versions). It is found that the programs written for MATCH, MATCH-WS, and MATCH-WWC approximately require as a fixed overhead about 200 INTEL/8086 machine instructions. The number 200 accounts for qualification evaluation, error checking, initializations, and resolutions for the match operation.

The matching algorithm (see Section 5.3.3) used for simple match and context specified matches, usually advances as much as the length of the search pattern at each match attempt. In the following analysis, the average length of a search pattern is taken as 8, which is the average length of an English word |76|. It is also observed that for each search attempt and advancement, 8 INTEL/8086 instructions are executed. This means that the number of INTEL/8086 instructions necessary for MATCH instruction is equal to the length of the literal text attribute (This is consistent with the theoretical analysis of the algorithm |7|). Therefore, the number of instructions which is necessary to process a tuple is equal to 200+ the length of the literal text attribute. During these calculations the search offset in the literal text attribute is taken as 0.

The number of INTEL/8086 instructions that are necessary for the context specified match instructions (MATCH-WS and MATCH-WWC) can be found as follows. First of all it is known that these instructions are used after matching the first term. In the selected tuples, the two extremes for the match position of the first term are the very beginning of the text attribute or the very last position of the text attribute. Therefore, it can be assumed that on the average the first keyword is matched at the middle of the text attribute. Similarly the match position for the second term can be just after the first term or can extend till the end of the literal text attribute. Hence on the average, the end of the search context for the second term will appear at the middle of the search area. As a result, one can assume that, in the context specified match instructions, one forth of the literal text attribute will be searched. In order to determine the end of the search context on the literal text attribute, on the average three INTEL/8086 instructions is needed to check each character. This means that the number of instructions necessary for determining the context ending point will be

$$3 * (\text{the length of the literal text attribute} * 1/4)$$

As stated above, during term matching one forth of the literal text attribute will be scanned, then the number of INTEL/8086 instructions needed in the context specified match instructions would be

$$200 + (\text{the length of the literal text attribute}) * (3/4 + 1/4)$$

As a result, it is shown that MATCH-WS and MATCH-WWC instructions require the same number of INTEL/8086 instructions as the MATCH instruction.

In order to arrive at a decision for :

a) the number of subcells/cell (k) to be used;

b) the data rate of the circulating memory (CM);

c) the tuple size;

various experiments were carried out by using the analytical model of RAP.3 Version-I which is given in Appendix C. During experimentation, the clock rate of the INTEL-8086 microprocessor is taken as 10 MHz and it is estimated that on the average each INTEL/8086 instruction requires 14 clock cycles. The data rate of the circulating memory is taken as 5, 10, and as an extreme case 40 MHz. The size of the circulating memory is taken as 4 megabits. 128, 512 and 1024 are the values assumed for the tuple size (in bytes). In each tuple, 18 bytes are reserved for 9 numeric attributes, 1 byte is reserved for delete flag and the mark bits, therefore the remaining length for the literal text attribute will be 109, 493, and 1005 respectively. Notice that in RAP.3 Version-I word and sentence overflows are allowed. During term matching, word splitting is resolved by an instruction called MATCH-OVFW (match-overflow). Since the execution time of this instruction is very small it will not come up in the remainder of the chapter.

The results of the experiments performed on the performance of RAP Version-I during the execution of the match instructions are given in figures 5.6 through 5.10.

In Figure 5.6 the plot of number of 10^6 characters scanned/ second is given versus number of subcells/cell with data rate as parameter. In this figure it is observed that, for lower data rates of CM, the increase in the number of subcells does not increase the number of characters scanned, since most of the time allowed for tuple processing will be wasted, since $T_{AVLNW} > T_{REQ}$. With the 10 MHz CM data rate, since the time allowed for processing is lowered due to fast circulation, the increase in the number of subcells will increase the number of character scanned in tuple sizes 128 and (at the beginning) 512. The increase in the number of subcells is not effective for the tuple size of 1024. Since the required tuple processing time (T_{REQ}) is already provided due to the length of the tuple. The CM data rate of 40 MHz will always benefit from the increase in both the tuple size and the number of subcells provided. This is because the processing time provided by the increase in both the tuple size and the number of subcells is not sufficient due to the speed of the CM data rate (since the data rate is high T_{BIT} will be low, this will lessen T_{AVLNW}).

Figure 5.7 shows the effects of varying data rates, tuple size, and the number of subcells on the normalized processing time (i.e., total cell processing time including waits divided by the total processing time if there were no waits involved). It shows that 40 MHz forces the architecture heavily, e.g., with a 4 subcell configuration with a tuple size of 128, it completes one circulation 8 times longer than architecturally allowed one RAP circulation time.

Figure 5.8 shows the real processing time. It shows that 5 MHz CM data rate with all the tuple sizes requires a very long time with respect to the higher data rates.

Figure 5.9 a shows when the CM data rate goes from 5 MHz to 40 MHz, the load factors (L) for the tuple sizes 128, 512, and 1024 increase about 8, 4.5 and 4 times respectively. Notice that, since the increase in the tuple size will increase the T_{AVLNW}, the load factor for longer tuple sizes are smaller with respect to the shorter tuple size(s). Figure 5.9.b shows the increase in the number of characters scanned/second as the CM data rate varies from 5 MHz to 40 MHz. These two figures are drawn for $k = 4$. However, this trend is generally valid for any number of subcells. For example, the same figures are drawn for the case when k is 6 in Figure 5.9.c and Figure 5.9.d. Since the increase in the number of subcells/cell will increase the degree of parallelism, the slopes of the curves of Figure 5.9.c are lower, and the slopes of the curves of Figure 5.9.d are higher with respect to the figures 5.9.a and 5.9.b. In other words, increase in the number of subcells will decrease the overloading in the system and at the same time it will increase the throughput.

Figure 5.10 shows the effect of in cell parallelism factor in RAP as a function of overload factor. Where, the in cell parallelism factor at an overload factor L ($ICPF_L$) is defined as follows (where $L > 1$):

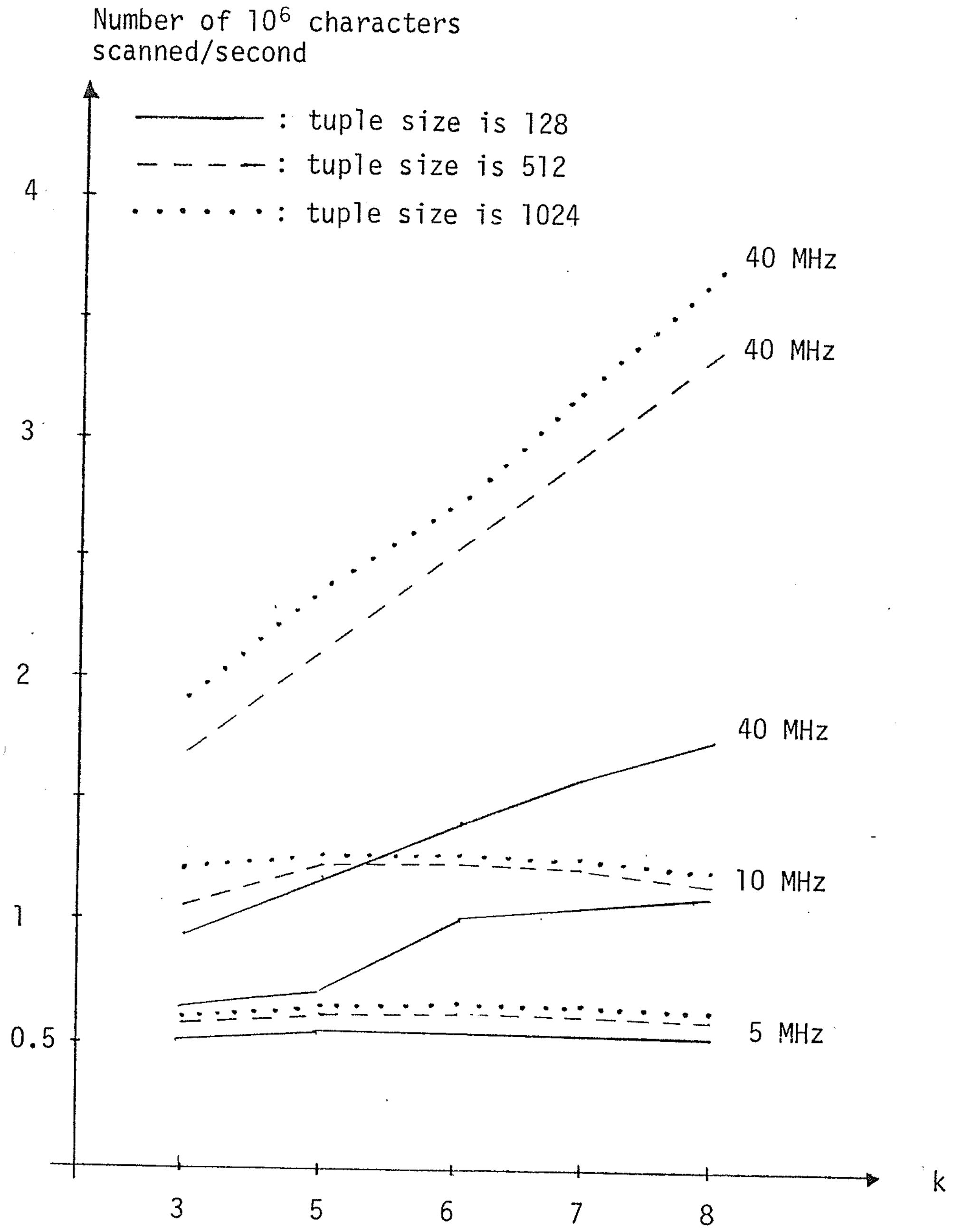


Figure 5.6 - Plot of number of 10^6 characters scanned/second vs number of subcells/cell (data rate as parameter).

$$\text{ICPF}_L = \frac{\text{the number of characters scanned when load factor is L (L>1)}}{\text{the number of characters scanned when L = 1}}$$

In Figure 5.10, if a curve is above the 45° line (i.e., the line having the slope of 1) then this is a favorable situation. The reverse situation is obviously an unfavorable case. This is clear from the definition of ICPF_L. In this figure, it can be observed that when an overload of L is imposed, the throughput returned by RAP is generally higher than the value of L. This is because of the reason that a wait time imposed for a subcell will also be used by its (k - 2) previous subcells.

The in cell parallelism, improvement for the tuple sizes of 128 and 1024 are shown in Figure 5.10 for the cases of k being 4 and 6. The step increase in "in cell parallelism improvement" with k = 4, tuple size of 1024 and with k = 6 with the tuple sizes of 128 and 1024 is not due to the in cell parallelism factor, actually there is no wait at all, but due to the CM data rate increase from 5 MHz to 10 MHz. The above mentioned k and tuple size combinations with the CM data rate of 5 MHz result with $T_{AVLNW} >> T_{REQ}$. The increase in the CM data rate causes slight decrease of T_{AVLNW}, but at the same time it will still be sufficient for T_{REQ}.

From these observations, it can be said that, the CM data rate should be chosen to benefit from the increase in the load factor (if any). For example, in tuple sizes 512 and 1024, when the data rate comes from 5 to 10 MHz there is an ignorable increase in load factors, but the gains in the number of characters scanned are considerable. Therefore, in the text retrieval operations with RAP, a wise combination of CM data rate and tuple size should be selected.

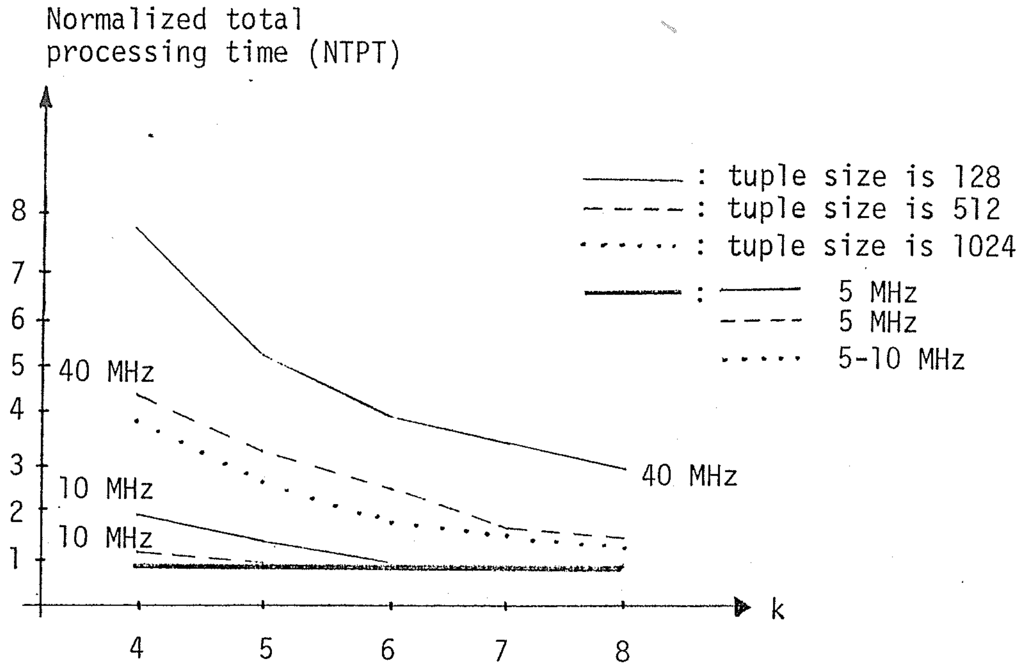


Figure 5.7 - Plot of normalized total processing time vs number of subcells/cell (data rate as parameter).

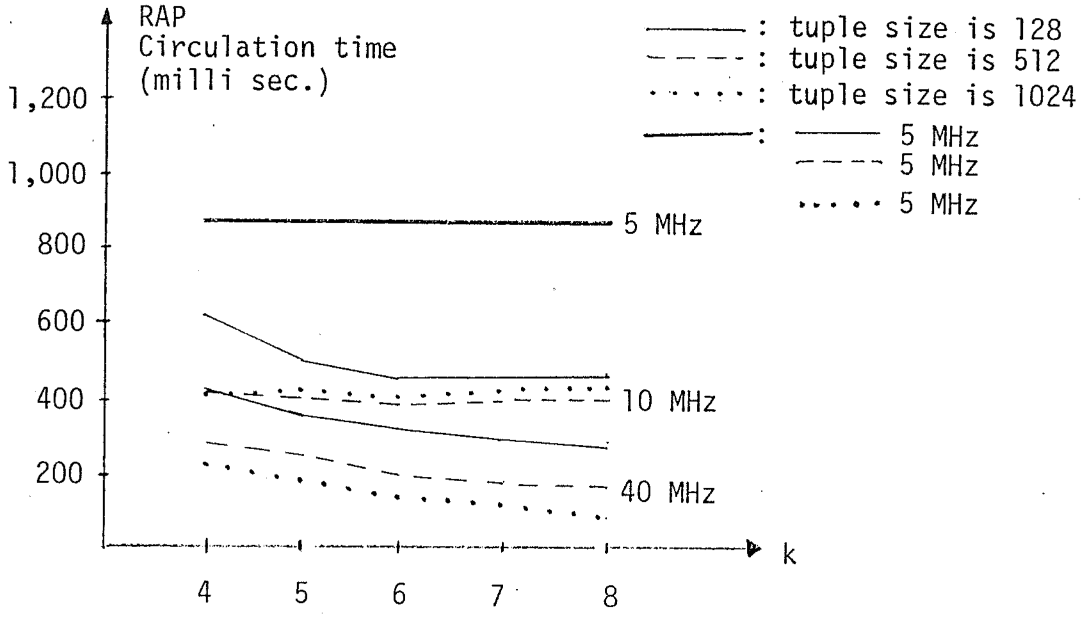


Figure 5.8 - Plot of real processing time (RAP circulation time) vs number of subcells/cell (data rate as parameter).

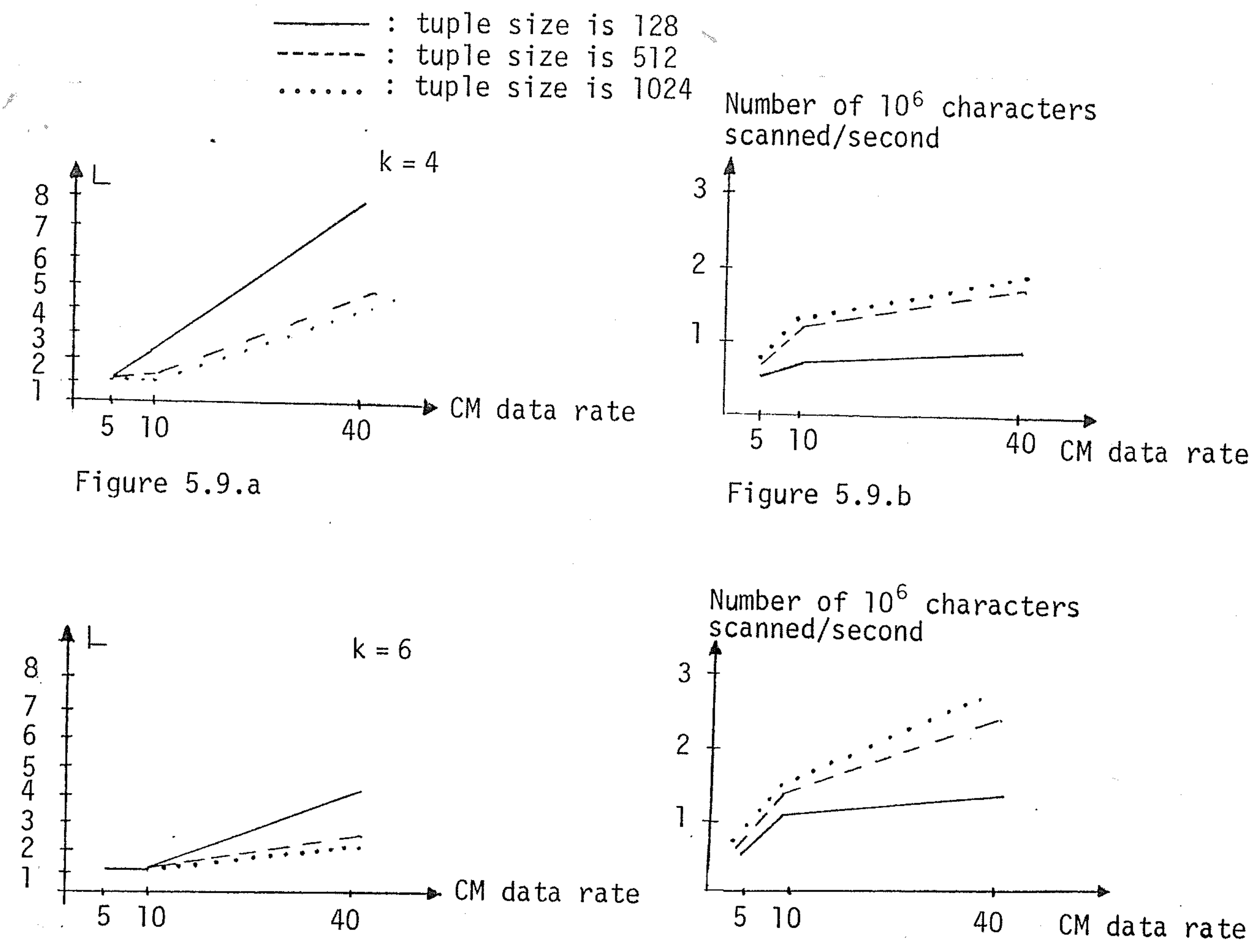


Figure 5.9.a

Figure 5.9.b

Figure 5.9.c

Figure 5.9.d

Figure 5.9 - Plot of load factor and number of 10^6 characters scanned/ second vs CM data rate for k = 4 and 6.

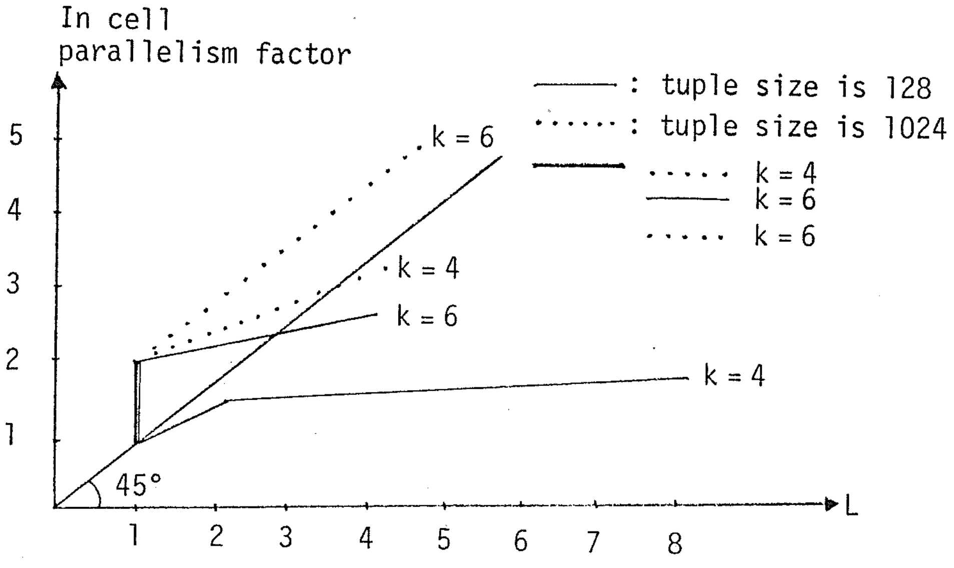


Figure 5.10 - Plot of in cell parallelism factor vs load factor.

It was decided that 4 subcells with a CM data rate of 10 MHz and with a tuple size of 1024 would be a cost-effective choice to reduce hardware complexity and impose practically no waits (L = 1.02). This provides a speed of 1,200,000 characters scanned per second per cell. One circulation of RAP would be completed in 425 milli seconds. Such a configuration with 16 cells (which is a practical number to realize) will result with 19,200,000 characters scanned/second.

If the tuples of a document relation need to be short in length, a literal attribute fragmentation may appear with large RAP tuple lengths, i.e., the considerable part of a literal text attribute may remain unused. To prevent this, a shorter tuple size must be chosen in such cases. For example, for abstracts the tuple size 512 may be suitable without introducing a performance degradation; with this size 1,100,000 characters would be scanned per second per cell while one RAP circulation would be completed within 450 milli seconds with the load factor of 1.16.

A time estimation for a search expression is given in the following (the search expression is written in an artificial high level text retrieval language):

within sentence [DATABASE *, (MACHINE or PROCESSOR)]
and within 5 words [RELATION, MODEL]

The above query finds all documents that contain DATABASE followed by any character and MACHINE or PROCESSOR in the same sentence and RELATION within five words of the word MODEL. The number of necessary RAP revolutions are given below (the RAP programs necessary for performing this text retrieval operation in RAP.3 Version-I is given in |11|, since the keywords MACHINE and PROCESSOR are searched within sentence with the same keyword, which is DATABASE*, there is a decrease in the number of revolutions needed to match DATABASE* and PROCESSOR within the same sentence). It is estimated that the above search expression will require 56 revolutions in RAP.3 Version-I |11|. With a RAP configuration of 16 cells, 4 subcells/cell, 4 Mega bit CM size (total memory size is 8 Mega byte), 1024 byte tuple size, 10 MHz data rate and with 10 MHz INTEL/8086 microprocessor used in the subcells, the total processing time will be

56 RAP rev * 425 milli sec./rev = 23.80 sec.

The above number means that 8 Mega byte is searched and necessary query resolution have been performed in about 24 seconds.

5.4.2. Performance of RAP.3 Version-II in Text Retrieval

The firmware routines of RAP.3 Version-I for text retrieval are also used in RAP.3 Version-II. In this version, due to variable tuple length feature there is no overflow during simple term matching. Accordingly, there is no instruction for simple term overflow resolution in the second version of RAP.3.

In estimating the performance of RAP.3 Version-II in text retrieval, the model given in Section 5.1 is used |62|. As mentioned previously, in this version, the necessary attributes of a tuple are accessed by using the DMA facility and the modified attributes are written back to the (pseudo) CM. In the match instructions, the text attribute is read, however it is never written back to the CM (since no change is made on the text by this set of match instructions). This feature of the match instructions lowers the processing time of a tuple.

In RAP.3 Version-II, due to variable tuple length feature, there is no overflow during simple term matching. The only overflow condition can appear in context sensitive text retrieval operations such as <A.n.B>, <A,B>n, A...B, and A * B. The context resolution for them is provided by the LINK-PASS instructions. Because of this reason the RAP programs for text retrieval operations are considerably simpler than the first version.

In Figure 5.11, the plot of 10^6 character scanned/second is given versus the number of subcells/cell. It is clear from the comparison of Figure 5.6 and Figure 5.11 that the processing speed of RAP.3 Version-II is considerably higher than the first version. The increase in number of subcells also increases the number of characters scanned per second (the tuple lengths specified are the average length for all tuples).

Figure 5.12 shows the real processing time versus number of subcells/cell. In this figure, instead of using the phrase RAP circulation time, the phrase RAP scan time is used. Since the RAP memory, which is a RAM, is scanned (RAM memory does not circulate. The design of RAP.3 Version-I is made for a circulating type memory. In the implementation RAM is used. The RAM contents are retrieved by DMA and are written back to its original place. Therefore in a sense, it emulates a circulating memory). The comparisons of Figure 5.12 and Figure 5.8 also shows the superior performance of RAP.3 Version-II with respect to RAP. Version-I.

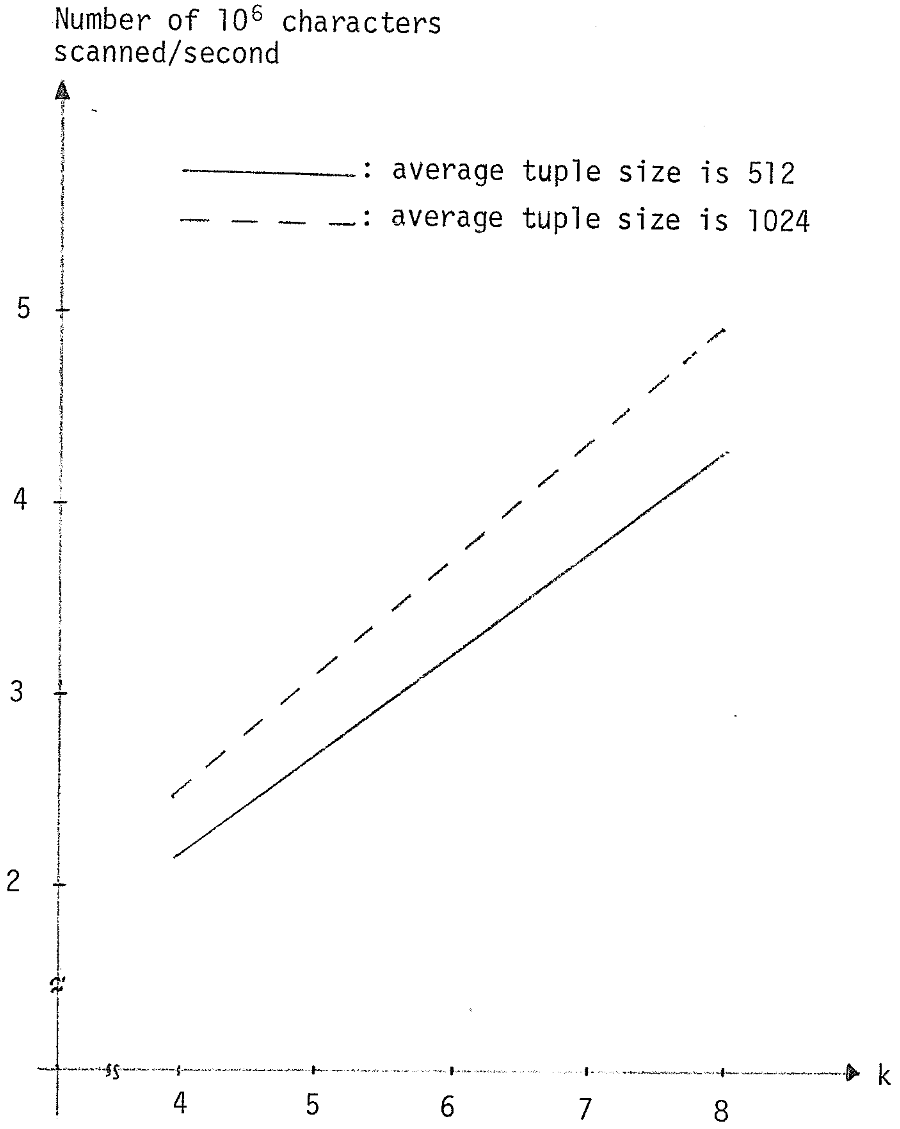


Figure 5.11 - Plot of number of 10^6 characters scanned/second vs number of subcells/cell

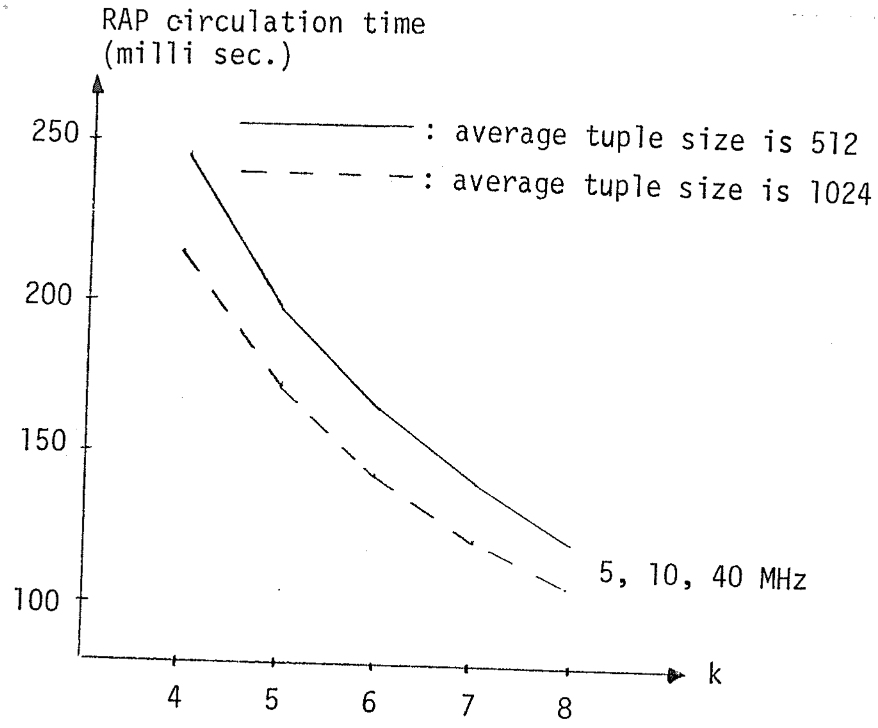


Figure 5.12 - Plot of real processing time (RAP scan time) vs number of subcells/cell.

In RAP.3 Version-I all tuples are read in all executions of the MATCH-WS and MATCH-WWC instructions although they are not searched for a term match. In Version-II only the tuples which are going to be searched will be accessed. This means that text retrieval operations, such as match within sentence and match within word count, will be executed much more faster in RAP.3 Version-II.

Due to variable length text attribute feature, no fragmentation will occur in Version-II. An average tuple size of 1024 bytes, 4 subcells provide 2,500,000 characters scanned per second per cell. Total scanning time of the 8 mega bit memory will be about 215 milli seconds. Such a configuration with 16 cells will result with 40,000,000 characters scanned/second.

The execution of the query, which is specified in the previous subsection would require roughly 40 RAP revolutions (or total scanning time) in Version-II. This makes 8.6 seconds. This states that RAP.3 Version-II is superior to RAP.3 Version-I in text retrieval operations.

5.5. SUMMARY AND CONCLUSIONS

In this chapter, after introducing a chronological overview of the RAP database machines, the new instructions of RAP for text retrieval operations are presented. The performance of RAP in the execution of these new instructions is analyzed. The implementation of the text retrieval operations with RAP is realized by a set of RAP programs. A macro processor, RAPMAC, is implemented and the RAP programs for the text retrieval operations are mapped into macro programs (see Appendix-E).

The work presented in this chapter shows that a very high pattern matching rate (2.5×10^6 characters per second per cell) is achieved in the latest version of RAP, namely RAP.3 Version II.

In the architectures proposed for text retrieval, there is a unit for query resolution. However, the architecture of RAP provides this subsection implicitly, i.e., the query resolution is done by the RAP assembler instructions in the RAP database machine.

A detailed analysis for the evaluation of the text retrieval operations in the RAP database machine is needed and left as a future research problem.

As a result it can be said that the performance of RAP in text retrieval operations is favorably comparable with performance of the architectures proposed to date.

6. A MODEL FOR INTEGRATED FACT/DOCUMENT INFORMATION SYSTEMS

In this chapter, an experimental system, which aims to synthesize a relational DBMS and IRS capable of context sensitive full text searches, is presented. The IR system relies on a clustering subsystem for database partitioning and relation fragmentation. For the implementation of this part the single-pass algorithm which is presented in Chapter 3 is proposed. The support architecture for the context sensitive search operations is the RAP.3 database machine. The use of the RAP.3 machine brings all of its capabilities into such an integration |16,71,75|.

The demands for effective and timely retrieval of facts and documents are constantly increasing. Also, the global information structures of applications are more sophisticated than ever before, requiring representations of complex relationships among facts as well as documents. An efficient and effective retrieval environment should therefore, integrate the mechanisms of both IRS and DBMS.

The answer for the proper integration lies in a synthesis approach that will combine IRS and DBMS by preserving their unique features in such a way that the advantages of both systems can be shared. Such a general purpose system should provide conceptual data modelling, relational algebra power (or equivalent) of DBMS, and at the same time the facilities provided by IRS, like indexing and classification, partial and/or full text search, and query feedback features of IR. It can be realized that the past efforts that simplifies the integration problem by confining one system within the framework of the other or simply combines IRS and DBMS in a semantically disjoint manner will not be sufficient, effective, and efficient within the context of the requirements of present applications. One cannot merely incorporate some string operations into a DBMS data language |99| or store keywords as an attribute of formatted database |24,94|. One cannot implement DBMS functionalities as file programs embedded within an IRS.Such an effort would lack the advantages of DBMS such as data independence, real time response, and ad hoc query formulation capability. The efforts of using the DBMS concepts in IRS environments can be seen in |23,57|.

6.1. INTEGRATION MODEL

In the integrated information system, a common framework has been established for the physical data structure so that both formatted and unformatted data can be manipulated together under an instruction set that embodies DBMS as well as context sensitive full text processing power. Refer to Chapter 5 for the representation of unstructed data in a structured frame. A given user request such as "On the documents written by the authors who are referenced by the papers written by JOHN DOE, search for those with the phrase SEMANTIC DATA MODEL and AGGREGATION", would require (this will be continued as an example later on) both DBMS and IRS specific operations in an interrelated sequence of processing steps. In the notation to be used in this chapter, a user query such as the one quoted will be denoted as Q, whereas the subsets of it dealing with DBMS and IR will be indicated by Q_R and Q_T, respectively. Q_T is the portion of the request that deals with the context sensitive search such as the "SEMANTIC DATA MODEL and AGGREGATION in the same sentence". While Q_T will be preserved for final processing, a query called the system query Q_S will be derived from Q_T to initate a search on the relevant clusters in the database. This search is an effort to narrow the search space, before the full text search can be conducted.

In simple terms, a Q_S query will include those terms in Q_T that are included in the set of index terms of the collection with a possible inclusion of certain other components to increase system precision and/or recall. Figure 6.1 shows an abstract view of the query processing environment. with Q_S, a hierarchy of clusters will be searched (where this hierarchy is constructed by using the single-pass clustering algorithm presented previously) and the corresponding data D_{Q_S} will be returned when the cluster search is optimally terminated. The two resulting sets of data; D_{Q_R} and D_{Q_S}, corresponding to DBMS and IRS operations respectively will be jointly processed by the integrated system to produce the user's response. The user, screening this response, may ask the system to repeat the operation by providing an indication of the relevant/irrelevant documents. With this information, Q_S will be modified and the previous cluster selection will be refined.

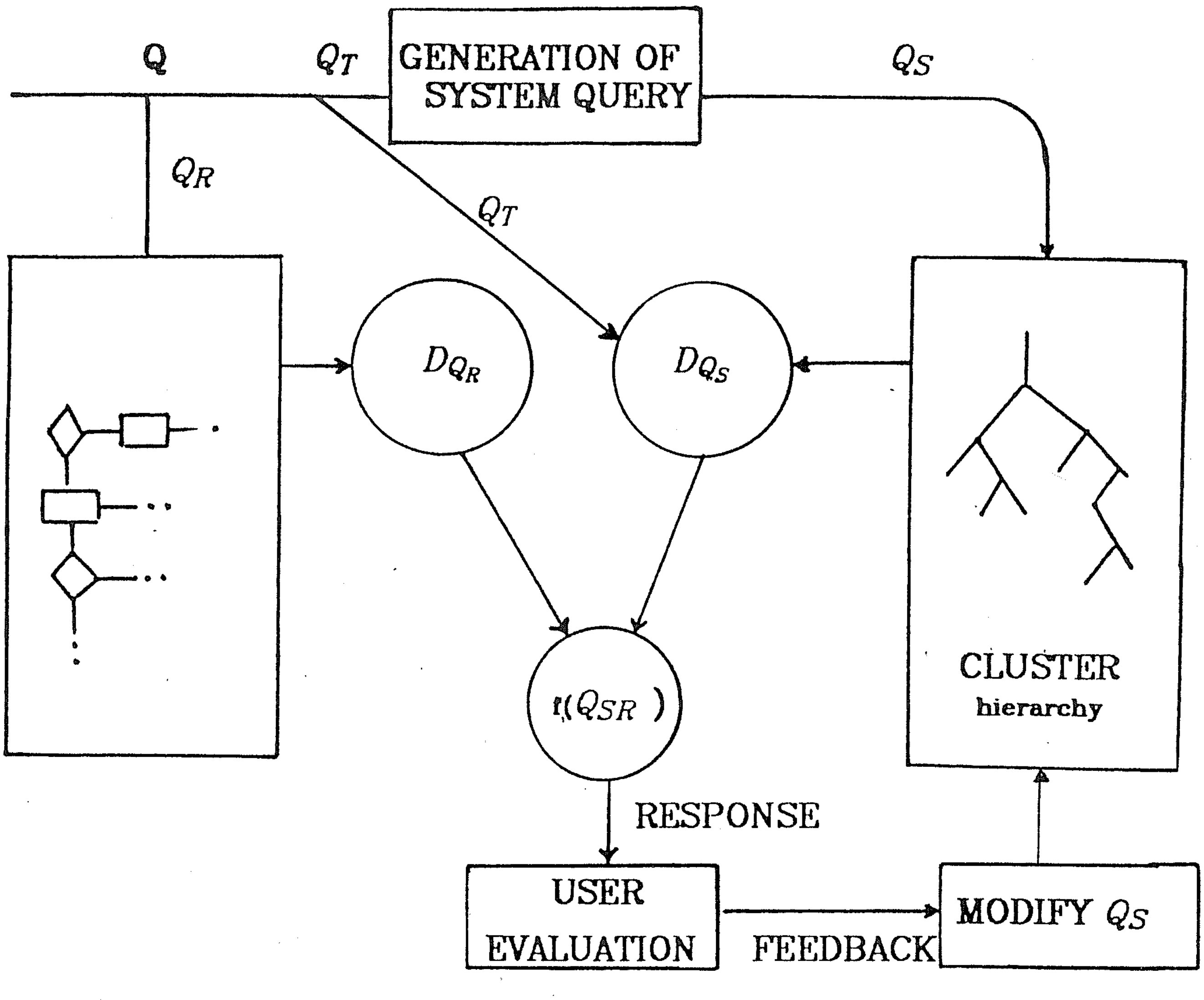


Figure 6.1 - Abstract view of the query processing environment.

6.1.1. A Mathematical Model

The integrated information retrieval system, IS, consists of a set of hierarchies and is described by a 6-tuple :

IS = < R, D, Q, C, E, T >   where

R represents a set of relations which contain structured data about the documents or document entities. The nature of the interrelationships among the structured entities will be described by means of a conceptual data model (E/R model) |17|.

D represents the set of documents stored as unstructured entities. The relationships of these unstructured entities among themselves and/or with those of R will be presented in the coming sections.

Q is a set of user queries. A given query, Q, is defined as

$Q = \{ Q_T, Q_R \}$ where

Q_T is the part of the query which holds the context sensitive document specification.

Q_R is the part of the query which holds search specification on the {R, D} set operable by the DBMS instruction set. As can be seen, the simplistic implication of $Q_R \rightarrow$ (IR on D) and/or (DBMS on R) is not always true. As shown in the example of the previous section, a search nevigation threads through both R and D several times in an interrelated manner. This is an important feature of the integrated system which is disregarded in the previous efforts.

C is a hierarchical structure of clusters of D.

E is a mapping function, called the evaluation function, $E:Q\rightarrow 2^D$, used to find the relevant documents to a query. In reality, a subset of the range of this mapping is reached through the complex operations of clustering, implementing search functions for these hierarchies, and a feedback function.

T represents the terms used for the description of documents.

6.1.2. Creation of System Queries

As shown earlier, a query Q consists of two parts Q_T and Q_R where Q_R is the relational DBMS operable subquery and Q_T corresponds to the context sensitive document retrieval subquery. Related with Q_T, there are a set of words, T_Q, such that

$$T_Q \cap T \subseteq \{T_1, T_2, \ldots, T_n\} \qquad \text{where } n = |T| > 0.$$

We cannot expect all of T_Q to appear in the filtered search query (system query) for the simple reason that not all of the words would be included in T since there are conditions for a word to be a (index) term (Furthermore, all the distinct terms of a collection might be used for indexing purposes and a user may specify a word which is distinct from all the words of the collection).

In Q_T, some terms will be used in the positive context, some others in the negative context. For that reason Q_T will be expressed as :

$$Q_T = Q_{TP} \cup Q_{TN}$$

where Q_{TP} and Q_{TN} corresponds to the parts of Q_T that deal with the terms specified in the positive and negative context respectively, and

$$Q_{TP} \rightarrow T_{TP} = \{ T_1, T_2, \ldots, T_k \} \qquad k \geq 0$$

$$Q_{TN} \rightarrow T_{TN} = \{ T_1, T_2, \ldots, T_\ell \} \qquad \ell \geq 0$$

$$Q_T \rightarrow T_T = T_{TP} \cup T_{TN}$$

$k, \ell \leq n$ and any relationship can hold between k and ℓ.

In the filtered search query for Q_T, which will be called a system query Q_S, there may be positive and negative term specifications. If a term T_i is the member of both sets, that is :

$$T_i \in T_{TP} \quad \text{and} \quad T_i \in T_{TN}$$

then T_i appears as a positive term in the system query. This selection is due to the fact that the information provided by the apparance of a term rather than its non-appearance is more important |106|. Therefore, a system query, Q_S, is

$$Q_S = Q_{SP} \cup Q_{SN} \qquad \text{where}$$

$$Q_{SP} = Q_{TP}$$

$$Q_{SN} = Q_{TN} - (Q_{TN} \cap Q_{TP})$$

hence, for the respective terms :

$$|T_{SP}| = |T_{TP}|, \; |T_{SN}| \leq |T_{TN}| \quad \text{and} \quad |T_S| \leq |T_T|$$

This is reasonable since, as stated earlier, one cannot expect all the words used in a context sensitive search request to appear in the terms used for indexing.

6.1.3. Context Sensitive Boolean Query Structure

The user query, Q, will be processed as follows :

a) For a more systematic query processing purposes the query Q will be converted into a disjunctive normal form, i.e.,

$$Q = (Q_{11} \wedge Q_{12} \wedge \dots \wedge Q_{1n_1}) \quad \dots \quad (Q_{m1} \wedge Q_{m2} \dots \wedge Q_{mn_m})$$

where each Q_{ij} $(i=1,\dots,m,\ j=1,\dots,n_m)$ may be a single word or or a context sensitive document retrieval operation (see Appendix A). Furthermore, each Q_{ij} may be either in a positive or negative context (e.g., $\neg$(A,B) in sentence implies (A,B) in sentence should not appear within the same sentence).

b) From the resulting query, generate a list of subqueries such that they will be in an ordered quadruple $<Q_R, Q_T, Q_S, Q_{SR}>$ where Q_R and Q_T are as explained before and Q_S and Q_{SR} are obtained in the following steps.

c) The words used in Q_T will be searched in the terms used for the description of clusters. Non-matching words will be dropped and the query, now left with only the terms, will be a system query Q_S.

d) Concept hierarchies, continuous word phrases, and/or citation linkages will be the possible condidates of additions to Q_S if expansion due to recall and precision will be necessary in the course of the experiments.

e) Q_{SR}, referring to Figure 6.1, is a further retrieval operation on D_{Q_R} and D_{Q_S} and the answer set returned by it can be expressed as :

$$f(Q_{SR}) = \pi D_{Q_R} \ f_{Q_T}(D_{Q_S})$$

In other words, $f(Q_{SR})$ is the data returned by a further retrieval function operating on the product of the data sets D_{Q_R} and f_{Q_T} (D_{Q_S}) which correspond in turn to the data returned by Q_R and the context sensitive operations of Q_T executed on the data set returned by the hierarchical cluster system search.

f) Repeat Q_{SR} through the feedback loop (if necessary) until the user and/or certain performance indicators are satisfied.

In (d) through (f) $f(Q_{SR}) \subseteq D$ will be always true.

6.1.4. Clustering Subsystem and Cluster Hierarchy

The clustering subsystem uses a seed oriented partitioning type classification based on the new concept called cover coefficient. This subsystem will utilize the single-pass clustering algorithm introduced previously. For its detailed explanation refer to Chapter 3.

Once the clusters are formed, a search will be performed by comparing the centroid vectors with that of Q_S returning D_{Q_S} for the full text search. If however, there are very large number of clusters making the linear, single level, cluster search prohibitive one can construct a hierarchy of clustering (tree), HCT, for a sublinear search.

In the mathematical model, C represents a hierarchical type cluster structure for D with ℓ levels. This is pictured in Figure 6.2. In this structure, if a node c_i has n children $c_{i1}, c_{i2}, \ldots, c_{in}$, then the documents contained in c_i is:

$$D_{c_i} = D_{c_{i1}} \bigcup D_{c_{i2}} \bigcup \ldots \bigcup D_{c_{in}}$$

Furthermore,

$$D_{c_{ij}} \bigcap D_{c_{ik}} = \phi \qquad \text{for } j \neq k$$

which leads to

$$D_{c_i} \bigcap D_{c_{ik}} = D_{c_{ik}}$$

These properties hold for all levels because the clustering algorithms used are non-overlapping.

As to the terms, if T_{C_i} is the set of terms used for the description of the centroid of cluster C_i, then $T_{C_{ij}} \cap T_{C_{ik}} = \phi$ is not necessary true for $j \neq k$ since a given term can also be used in the description of documents in a different cluster. Similarly,

$T_{C_{i1}} \cup T_{C_{i2}} \cup \cdots \cup T_{C_{in}} = T_{C_i}$ is not necessarily true for any i where $1 \leq i < 1$ for an ℓ level hierarchy because we cannot assume that all documents on both sides of the relationship are isomorphic.

The hierarchical document cluster implies a partial ordering relation. Assume R is a generalization relation for all the documents related with node-i. R will be the generalization of both the documents of the node as well as all the documents contained in the descendants of that node since the following conditions of reflexivity, antisymmetry, and transitivity will be applicable :

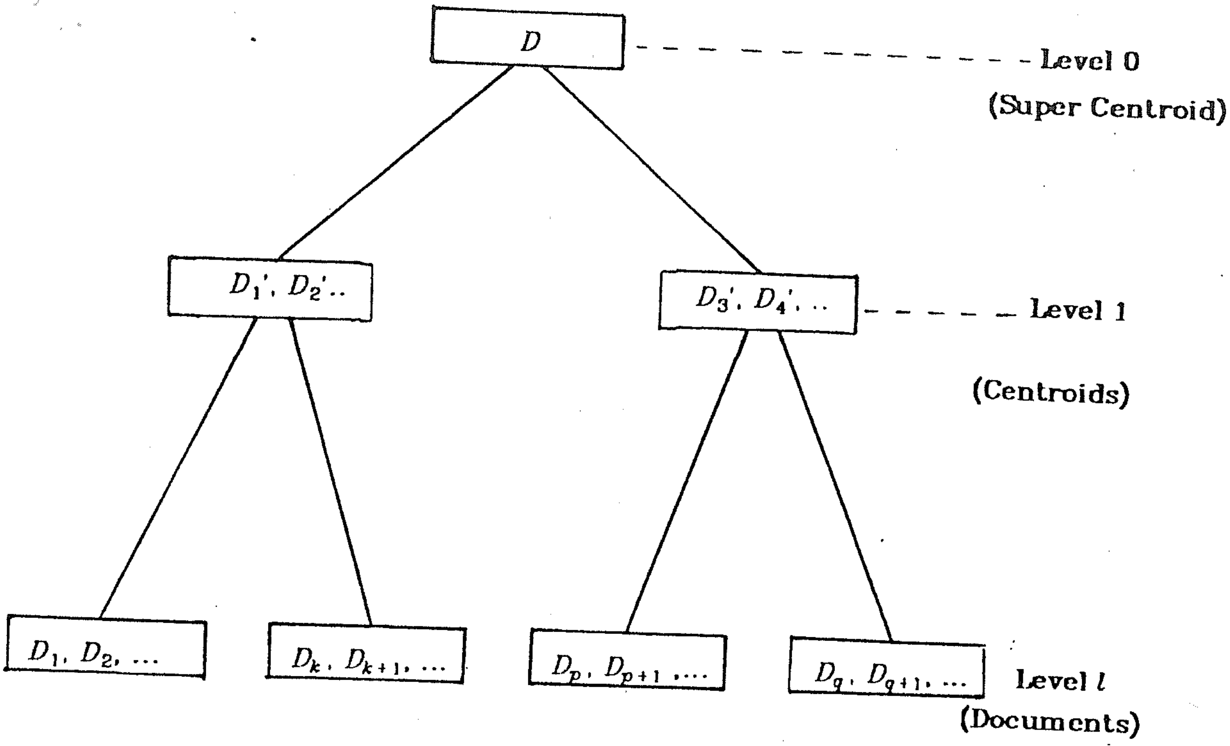


Figure 6.2 - Hierarchical clustering

The algorithm is the following :

```
a) Determine $n_c$ of $D$.
b) $D_s = \phi$;   $i = 0$;
c) repeat ;
      $i = i + 1$ ;
      Randomly select $k_i$ documents from the set of unclustered
      documents $(D - D_s)$. This means that $D_s$ is extended as
      $D_s = D_s \cup D_{k_i}$ ;
      Determine $n_{c_s}$ of $D_s$ ;
   until ;  $n_{c_s} = n_c$ ; /*or $n_{c_s}$ is considerably close to $n_c$*/
d) Generate the core clusters by using the documents of $D_s$.
   Create the hierarchical clustering tree (HCT) by using
   these croe clusters.
e) For the members of the unclestered documents $(d \varepsilon D - D_s)$
   perform a selection navigation on the HCT. Find the
```

most similar core cluster to this unclustered document. The document is assigned to this core cluster (notice that the core clusters are the lowest level clusters).

f) After clustering all the documents modify the centroids of the core clusters using the old and new members. Construct the HCT with these new centroids.

The above algorithm has one significant superiority : by the aid of the decoupling coefficient concept one can determine the number of clusters within the collection ($n_c = \sum_{i=1}^{m} \delta_i$). This is the guideline for the document selection process and determination of the core clusters.

A study by Jardine shows that about 1% of the documents of a very large collection would lead to a reasonable core clustering |51|.

In step-e, in the search of the HCT, search process may stop at an intermediate node. However, one should get to the lowest level (leaf) cluster under that node to add the document being processed. In such cases, one can pick the cluster with the maximum cluster seed power as the seed document.

6.1.5. Cluster Search and Matching Function

In the integrated system model, the nodes of the cluster hierarchy and the system query, Q_S, will be represented as vectors equal in length to that of the document definition vectors. In the query definition vector corresponding to Q_S, each element can have one of the three possible values of 1, -1, and 0 corresponding to the use of the related term in the positive context, negative context, and non-use of it respectively. As stated earlier, terms appearing in both context are assumed to be in the positive context. The search process will employ a top down search on the HCT. Depending upon the values of n_c, d_c, and the required precision/recall levels, the following two search strategies can be used:

a) Narrow search : At each node, the decendant which gives the highest value for the matching function is taken. The search terminates when none of the descendants of the processed node can exceed in match function value that of the node (or a given threshold) |106|. With this strategy, we always take a single branch out of a node.

b) Broad search : From a node, descend to children nodes that satisfy the match criterion (e.g., a similarity threshold). Stop when no child satisfies the match and send the parent (i.e., the documents related with the parent node) to full text search. Accordingly, in this search, we may send multiple nodes from different branches of the tree to full text search.

The searching algorithms for the above strategies are given in the following :

Narrow search :

```
cnode = top - node ;
mxval = mch - func (cnode) ;
repeat ;
  snode = cnode ;
  repeat ;
    node = nxt - child (snode) ;
    mchval = mch - func (node) ;
    if mchval ≥ mxval (or mchval ≥ treshold) then
```

```
            do ;
              mxval = mchval ;
              cnode = node ;
            end ;
      until nxt - child (snode) = 0 ;
    until cnode = snode ;
```

Meaning of the variables are as fololws :

top-node : top most node of the HCT.

cnode : the candidate node for selection.

snode : the node selected for searching.

mch-func : returns the matching function value of its argument.

mxval : matching function value related with the candidate node.

nxt-child: returns the next child of its node, if no more child then returns 0.

Broad search :

```
n = 0 ;
sp = 1 ;
stack(sp) = top-node ;
disqualified(sp) = false
sp = sp + 1 ;
repeat ;
  if mch-func(sp-1) ≥ treshold then
    do ;
      if sp _ 3 then disqualified (sp-2) = .true. ;
      node = nxt - child (stack (sp)) ;
      if node = 0 then
        do ;
          if ¬ disqualified(sp-1) then
        do ;

          n = n + 1 ;
          snode (n)   stack(sp) ;
          sp = sp - 1 ;

        end ;

      end ;

      else do ;

            stack (sp) = node ;
            disqualified (sp) = .false. ;
            sp = sp + 1 ;

          end ;

    end ;

    else  sp = sp - 1 ;

until  sp = 1 ;
```

Meaning of the (new) variables are as follows :

disqualified : is an array parallel to the "stack" array. If a child of a node satisfies the search condition then the node at the top is disqualified (notice that disqualified nodes are not selected).

snode : an array which contains the selected nodes.

n : number of selected nodes.

Two measures are proposed for the matching function. One is the "coupling function" and the other is a similarity measure. The coupling function comes from the idea of the coupling coefficient. Accordingly, one can start with an m x n matrix whose first row would be the query vector and the remaining m - 1 rows would correspond to the centroid vector of the lower level subclusters of the node(s) identified in the search of the HCT. By using this matrix one can find the coupling of the query with the centroids or coupling of the centroids with the query (Notice that special attention should be given to the -1 entries of the query vector. They will match with the centroid entries with a value of 0). The coupling coefficient among the query and the centroids will determine the clusters to be chosen for further consideration (see Chapter 7 for the use of this concept for the selection of the clusters).

The second measure would be a similarity function of the following form :

$$\mathrm{SIM}(Q_S, C) = \Big(\sum_{j=1}^{n} F_1(q_j, c_j)\Big) \Big/ \Big(\sum_{j=1}^{n} (F_1(q_j, c_j) + F_2(q_j, c_j))\Big)$$

where

$$F_1(q_j, C_j) = \begin{cases} q_i \times g_j & \text{if } q_j > 0 \qquad (1) \\ 0 & \text{if } q_j = 0 \\ \mathrm{abs}(q_j) \times (|C| - g_j) & \text{if } q_j < 0 \qquad (2) \end{cases}$$

$$F_2(q_j, C_j) = \begin{cases} q_j \times (|C| - g_j) & \text{if } q_j > 0 \qquad (3) \\ 0 & \\ \mathrm{abs}(q_j) \times g_j & \text{if } q_j < 0 \qquad (4) \end{cases}$$

In the above formula, (1) indicates that if a query term, q_j, position is a 1, then F_1 gives the number of documents containing that term under that centroid; g_j is the centroid entry. In (2), $|C|$ indicates the number of documents under the node, "abs" means absolute value and F_1 gives the number of documents that do not contain the negative query term. In (3), F_2 gives the number of documents in the nodes that do not contain the query term. In (4), F_2 gives the number of documents in the nodes that contain the query term. The SIM function is the conditional probability of hitting a one on the centroid vector of the HCT node given that a one was given in the query vector and hitting a zero on the centroid vector of the HCT node given that a minus one was given in the query vector.

6.1.6. Query Feedback

The query feedback to be used in the integrated system has some differences from the other studies. This difference is motivated by the fact that the integrated system will be concentrated on the context sensitive search specification. Accordingly, in this feedback scheme, the user query is kept fixed, but the system query, Q_s, is modified. Therefore, after each feedback process a new cluster(s) is(are) chosen, which would be most related with the modified Q_s, and the documents are processed with respect to Q_T (and Q_{SR} later). The feedback process will depend on the user choice although various measures of performance could be employed. The user will have the choice of accepting the results or triggering the feedback loop by giving an indication of the relevant and irrelevant documents. In the refinement procedure of the feedback subsystem, the terms appearing in the relevant documents will be emphasized while those in the irrelevant documents are deemphasized. The following criteria can be used for this purpose. If $|q_{ix}|$ denotes the number of appearances of term-i in set x where x can be one of system query, relevant documents (RL), or non-relevant documents (NR), then if

$$|q^p_{is}| + |q^p_{iRL}| > |q^p_{iNR}|$$

that is, if the sum of term-i's used in the positive context in the system query and those that appear in the relevant documents is greater than the number of positive context term-i's that appear in then non-relevant documents, then term-i will be kept in the refined Q_s, otherwise it will be dropped (or its weight might be reduced). Also, if

$$|q^N_{is}| + |q^N_{iNR}| > |q^N_{iRL}|$$

that is, if the sum of term-i's used in the negative context in the system query and those that appear in the non-relevant documents is greater than the number of negative context term-i's that appear in the relevant documents, then drop term-i (or reduce its weight) in the refined query.

The main goal of this feedback scheme is to preserve the context sensitive nature of the user query so that the advantages of full text search can be properly exploited.

6.1.7. Evaluation

For the evaluation of the system performance the precision (P) and recall (R) measures can be used. In this section an approach for performance evaluation will be proposed and an illustrative example will also be provided. Other performance evaluation measures depending on P-R values can also be derived very easily.

The definitions of recall and precision is given in the following:

$$\text{Recall} = \frac{\text{no of retrieved and relevant documents}}{\text{no of all relevant documents in the collection}} = \frac{c}{b}$$

$$\text{Precision} = \frac{\text{no of retrieved and relevant documents}}{\text{no of documents retrieved}} = \frac{c}{b}$$

This R and P figures can be defined both in terms of documents and clusters. The experimental procedure to be used is the following :

a) *Determine the documents and the thesaurus term covered by a context sensitive boolean query. Also determine the lowest level clusters which contain at least one document which stasifies the boolean query.*

b) *For each query perform the following :*

Create the list of selected nodes (i.e., the nodes which satisfy the search conditions).

The documents which are related to the selected nodes give the number of documents retrieved, some of these documents satisfy the boolean query and some of the subclusters connected to the selected nodes contain documents that satisfy the boolean query.

Calculate R and P for the query.

c) *Perform an overall evaluation for all queries (There are different averaging techniques for this purpose |92,106|).*

In the following an example is given for the evaluation process. In Figure 6.3 an HCT is given for the example. In this figure, the numbers outside the circles indicate the number of documents related with a node. Assume that clusters 8, 10, and 12 contain documents which satisfy the boolean query. Furthermore, assume that the number of relevant documents in these clusters are 5, 2 and 1 respectively (In R and P calculations, the variables a, b, and c will be used. The subscript D and C indicates that the variables are related with documents and clusters, respectively).

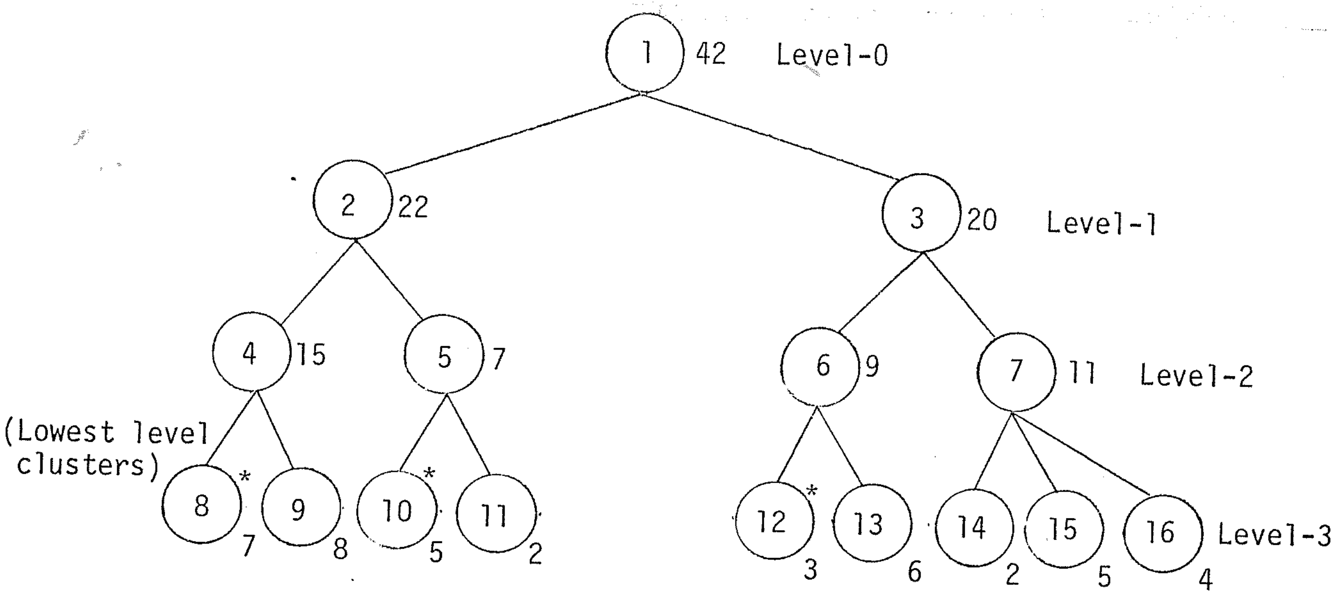


Figure 6.3 - Example HCT

As it is stated above the clusters 8, 10, and 12 contain documents satisfying the context sensitive boolean query and the number of relevant documents in these clusters are 5, 2 and 1 respectively. Hence :

$a_D = 5 + 2 + 1 = 8$ $\qquad a_C = 3$ (clusters 8, 10 and 12)

Notice that a_D and a_C are fixed for a query, but "b" and "c" changes according to the HCT node.

During the search navigation, assume that node-1 satisfies the search condition. P and R values for this node are as follows :

$a_D=8$, $b_{D_1}=42$, $c_{D_1}=8 \rightarrow R_{D_1} = c_{D_1}/a_{D_1}=8/8=1$, $P_{D_1}=c_D / b_{D_1}=8/42=0.19$

$a_C=3$, $b_{C_1} = 9$, $c_{C_1}=3$ $\quad R_{C_1} = c_{C_1}/a_{C_1}=3/3=1$, $P_{C_1}=c_{C_1}/b_{C_1}=3/9=0.33$

During navigation assume that node-2 satisfies the search condition. Accordingly :

$b_{D_2}=22$, $c_{D_2}=7 \rightarrow R_{D_2}=c_{D_2}/a_D=7/8=0.88$, $P_{D_2}=c_{D_2}/b_{D_2}=7/22=0.32$

$b_{C_2}=4$, $c_{C_2}=2 \rightarrow R_{C_2}=c_{C_2}/a_C=2/3=0.67$, $P_{C_2}=b_{C_2}/b_{C_2}=2/4=0.5$

Assume that node-3 does not satisfy the search condition, therefore forget the right hand side of the HCT in the navigation. At one lower level, assume that node-4 satisfies the search condition :

$b_{D_4}=15$, $c_{D_4}=5 \rightarrow R_{D_4}=c_{D_4}/a_D=5/8=0.63$, $P_{D_4}=c_{D_4}/b_{D_4}=5/17=0.33$

$b_{C_4}=2$, $c_{C_4}=1 \rightarrow R_{C_4}=c_{C_4}/a_C=1/3=0.33$, $P_{C_4}=c_{C_4}/b_{C_4}=1/2=0.5$

Lastly, assume that node-8 satisfies the search condition :

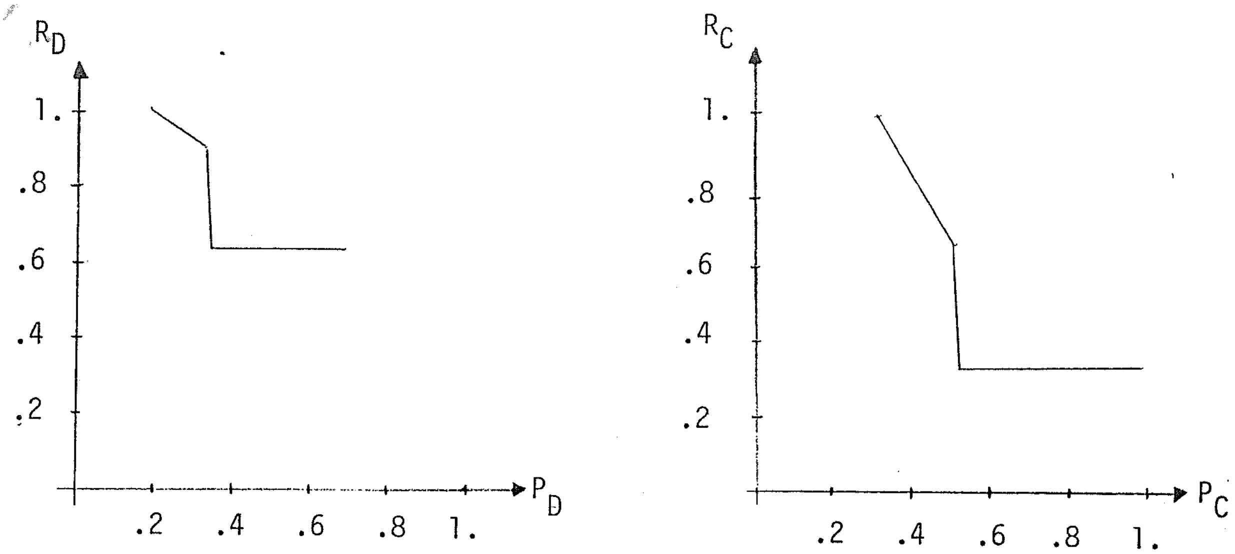
$b_{D_8}=7$, $c_{D_8}=5 \rightarrow R_{D_8}=c_{D_8}/a_{D_8}=5/8=0.63$, $P_{D_8}=c_{D_8}/b_{D_8}=5/7=0.71$

$b_{C_8}=1$, $c_{C_8}=1 \rightarrow R_{C_8}=c_{C_8}/a_{C_8}=1/3=0.33$, $P_{C_8}=c_{C_8}/b_{C_8}=1/1=1$

Assuming that node-5 and node-9 do not satisfy the search condition, only the documents related with node-8 will be moved to the RAP memory and the overall performance of the system in terms of R and P will be 0.63 and 0.71, respectively.

The above navigation process can be easily mapped into a R-P graph in terms of documents and clusters. Recall and precision values are usually plotted as "precision versus recall". However, the output pattern of the navigation call for the reverse order. The R-P curves for the example query navigation are depicted by Figure 6.4.

We can consider also another measure for evaluation. This would be a plot of R and P values versus the HCT level. The averaging of the



a. For documents

b. For clusters

Figure 6.4 - The plot of recall (R) versus precision (P) for the example

R and P values at each node position can be done in two ways :

a) Summation of R (P) values at a given level divided by the number of observations at that level. Notice that a query can reach to a level through more than one branch in case of the broad search strategy.

b) Summation of R (P) values at a given level divided by the number of queries at that level. Notice that, not all queries will reach to all levels.

Tabulation of the number of observations, number of queries, average recall, and average precision values at each level position can lead to a reasonable evaluation basis.

6.2. CONCEPTUAL MODELLING AND AN EXAMPLE APPLICATION

The universe of discourse of an integrated DBMS/IRS application will be defined by means of the Entity/Relationship (E/R) model. The diagrammatic representation of the E/R model uses rectangles to represent entities, diamonds for the multidimensional relationships among entities, labels to indicate the type of relationships or mappings (e.g. one to many, many to many, etc.), double lined rectangles for weak entities (or relationships), etc. |17|. In the RAP environment, an entity will correspond to a conceptualization whose representation will be in terms of formatted structures (i.e., a record type or specifically a relation) or a document whose representation will be in terms of unformatted structures. In the latter, the rectangle will be marked by an asterisk.

The conceptualization of the example application environment is depicted by Figure 6.5. To produce the RAP relations, the entities and relationships are mapped into relations in a one to one correspondence. One exception to this is with the entity called DOCUMENT, which is mapped into a set of relations for practical purposes. These relations are DOCUMENT, CITATION, ABSTRACTS, and SUB-HEADING. The relation definitions for the conceptual model given in Figure 6.5 are depicted by Figure 6.6.

In the resulting relations, most attribute names are self explanatory. Key attributes are underlined. Some of the attribute names are explained in the following ; JCODEN stands for journal code, the ABSTRACT attribute holds the full text of the corresponding document's abstract. Since an abstract can be mapped into more than one tuple, the attribute TUPID (tuple identifier) has been introduced in

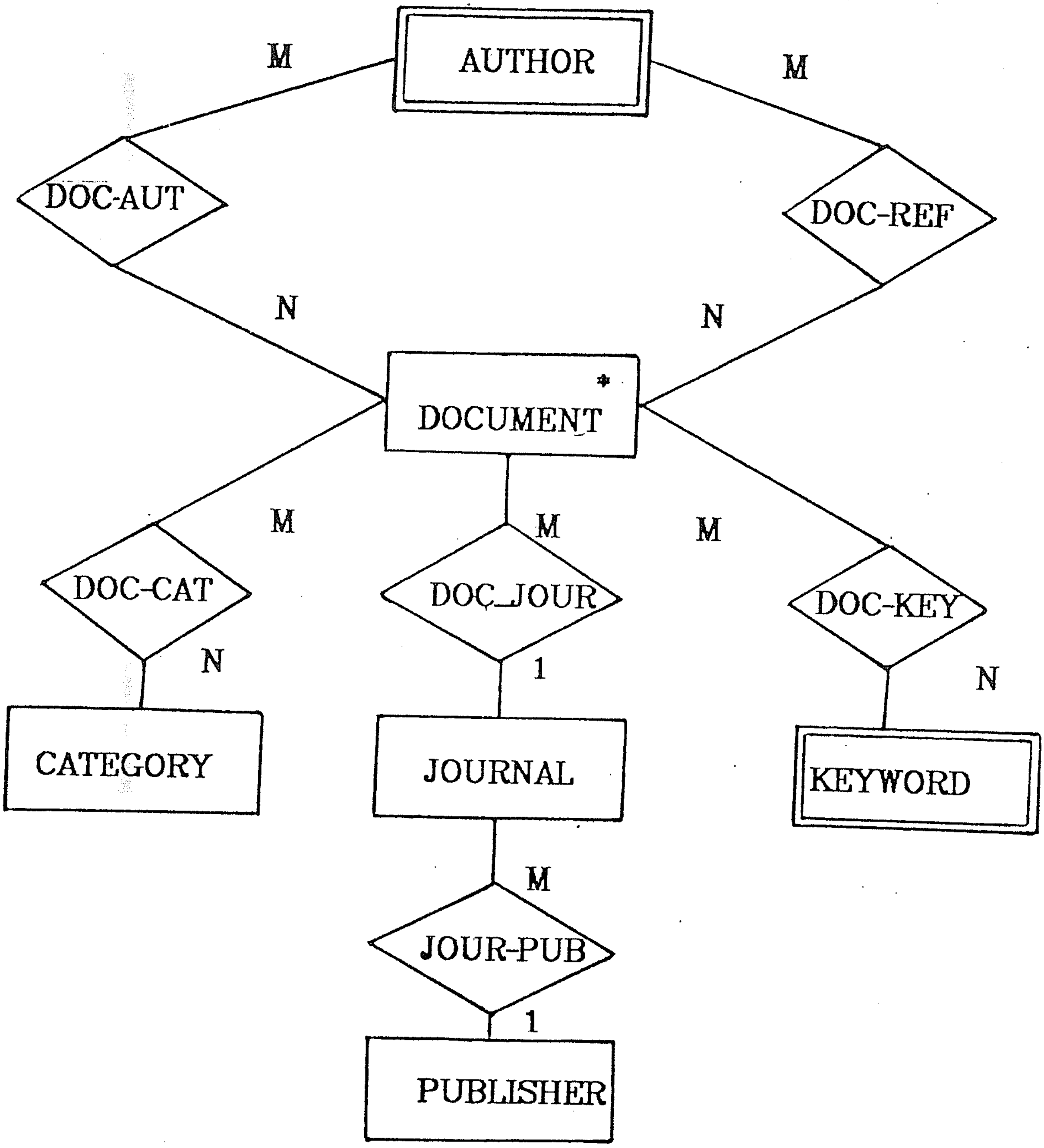


Figure 6.5 - Conceptual representation of the integrated application environment.

DOCUMENT < DOCID >

CITATION < DOCID, TITLE, DATE, JCODEN, VOLUME, PAGES, ND1, ND2 >

ABSTRACTS < DOCID, TUPID, ABSTRACT, ND1, ND2 >

SUB-HEADING < DOCID, STITLE, ND1, ND2 >

AUTHOR < AUTID, NAME, NPAPER >

DOC-AUT < DOCID, AUTID >

DOC-REF < DOCID, AUTID, NREF >

KEYWORD < KEYID, KEY, KFREQ >

DOC-KEY < DOCID, KEYID, WEIGHT >

CATEGORY < CATID, CATCODE, CFREQ >

DOC-CAT < DOCID, CATID >

JOURNAL < JCODEN, JNAME, NPAPER >

DOC-JOUR < DOCID, JCODEN >

PUBLISHER < PUBID, PNAME, NPUB >

JOUR-PUB < JCODEN, PUBID >

Figure 6.6 - A relational representation of the conceptual structure

the ABSTRACT relation. Similarly, the attributes TITLE and STITLE mean, respectively, full title and subtitle(s) of a document. The AUTHOR is a weak relation and tuple existence in this relation depends on the entity "DOCUMENT". A tuple (an author) can reside in the AUTHOR relation if he(she) has written a paper (document) or if he(she) has been referenced by a document which resides in the DOCUMENT relation. The attribute NPAPER (of the AUTHOR relation) and the NREF (of the DOC-REF relation) corresponds to the number of papers written by the author and number of times the author has been referenced in the documents respectively. Needless to say that the attribute DOCID uniquely identifies a document and the same is valid for the attribute

AUTID for the entity AUTHOR. The attributes KFREQ, WEIGHT, CFREQ, NPAPER, AND NPUB stand, respectively, for the number of times the corresponding keyword used in the documents, weight of the keyword (identified by the KEYID) in the document (identified by the DOCID), number of times the corresponding category used in the documents, number of papers published in the corresponding journal, and the number of journals published by the corresponding publisher.

The unary relation DOCUMENT has originally been introduced into the system to increase the speed of the semi-join operations (realized by the CROSS-MARK instruction of RAP) needed during query resolutions. In the new version of RAP.3 the sections of the tuples which are used in the instruction specification are moved into the buffer memory. Therefore, the unary relation approach will not make any contribution to the performance of the system. However, it seems that it is nice to have a separate relation to indicate the selected documents and the unary relation DOCUMENT will be used for this purpose.

A given user request such as "On the documents written by the authors who are referenced by the papers written by JOHN DOE, search for those with the phrase"SEMANTIC DATA MODEL and AGGREGATION" can be executed in this system. Based on Figure 6.5 and 6.6, a possible execution sequence for the above query can be given as follows (using the relations of Figure 6.6 and with no regard to semi-join optimization):

1. SELECT AUTHOR with name JOHN DOE.
2. SEMI-JOIN AUTHOR to DOC-AUT via AUTID.
3. SEMI-JOIN DOC-AUT to DOC-REF via AUTID.
4. SEMI-JOIN DOC-REF to DOC-AUT via DOCID.
5. SEMI-JOIN DOC-AUT to ABSTRACT via DOCID.

6. IR Search for "SEMANTIC DATA MODEL" and "AGGREGATION".

7. SEMI-JOIN ABSTRACT to DOCUMENT via DOCID.

8. READ DOCUMENT.

As can be seen in this query, operations 1 through 5, and 7,8 are DBMS operations and 6 is an IR operation and yet everything is linked and consequence of each other in the execution sequence. Although operation 6 is a single item in the list, it corresponds to a set of operations implemented as a RAP macro call which uses DBMS/IR instructions. To follow these, one should refer to Figure 6.1 which is the abstract representation of the integrated system. In this figure, Q represents the query program shown above (i.e., operations 1 through 8), Q_R corresponds to DBMS operations (operations 1 through 5, 7, 8) and Q_T to IR operation 6. In terms of the data sets and data manipulation functions, the following correspondence can be seen :

D_{Q_R} : $(\text{DOC-AUT})_R$, D_{Q_S} : $(\text{ABSTRACT})_R$-subscript R of relation namers indicates restriction-

$f(D_{Q_R})$: step 5, $f(D_{Q_S})$: step 7, $(\text{DOCUMENT})_R = f(Q_{SR})$

The following are the two examples of an DBMS/IR operation at the RAP program level:

6.3. SUMMARY AND CONCLUSIONS FOR THE INTEGRATION MODEL

In this chapter an integration model is proposed for DBMS and IRS. The proposed model relies on a clustering subsystem for database partitioning and relation fragmentation. For the implementation, the single-pass algorithm, which is presented in Chapter 3 is proposed. The support architecture for the search operations is the RAP.3 database machine. Since the implementation of the complete model would require much more time and effort then a single dissertation only some parts of it are implemented. These include the development of a macro processor, RAPMAC, as an interface to the software emulator of RAP (SERAP); implementation of the new match instructions in SERAP, and the development of the new clustering scheme.

The proposed integration is expected to be efficient due to following main features :

a) It uses an efficient implementation of a relational DBMS.

b) IR features are integrated with DBMS at the primitive level.

c) The IR primitives are based on the efficient string search operations and the context sensitive full text instructions carry out query resolution to a great extent.

d) Clustering is used as an efficient IR index in addition to the secondary key indexing of RAP.3 DBMS.

e) Integration of DBMS and IR at the primitive level results in minimum layers of software which contributes to overall system efficiency.

f) There are no need for a separate computer for query resolution and the necessary communication of term matches since all those functions are handled in the RAP.3 system.

The semi-join construct of the RAP.3 system (i.e., CROSS-MARK) is an important operation. In an integrated system where formatted and unformatted data are cross selected, the join operation must be implemented efficiently. The semi-join operation does not create a new relation to store the results and tansmits a minimum amount of data between relations.

7. THE PERFORMANCE OF THE TWO PARTITIONING TYPE CLUSTERING ALGORITHMS IN INFORMATION RETRIEVAL

In this chapter, the performance of the single-pass clustering algorithm (Algorithm-1) and the multi-pass clustering algorithm (Algorithm-2) will be evaluated in terms of their information retrieval behavior (the algorithms are described in Chapter 3). The effectiveness of the centroid generation policies associated with the algorithms and the matching functions used in the selection of the clusters will also be evaluated.

In order to deal with the first order variables a single level (non-hierarchical) clustering approach will be used.

The evaluation procedure is described in the following :

a) Map the natural language queries into the query vectors, (A query vector, Q_V is defined as follows : $Q_V = Q_W \cap T$, where Q_W is the words used in the query and T is the terms used for the description of the documents).

b) Perform a full search on the whole document collection then find the average performance for all of the queries in terms of recall and precision.

c) Cluster the documents.

d) Perform a cluster based retrieval (CBR) for all queries (i.e, find the clusters which are most relevant to a particular query and consider the documents of the selected clusters for *retrieval purposes) then find the average performance for all of the queries in terms of recall and precision.*

e) *Compare the results of the CBR with full search to find the effectiveness of the clustering algorithm in information retrieval.*

f) *Perform the steps c through e for both of the algorithms.*

g) *Compare the performances of the clustering algorithms.*

In the above testing procedure, two different matching functions will be used in the selection of the documents and the clusters. This means that there will be two performance results for the full search. Similarly, for the CBR the same two matching functions will be used for the selection of the centroids and documents. This means that there will be eight performance results for the CBR (=2 algorithms x 2 matching functions x 2 centroids). The evaluation procedure will test the suitability of the clustering algorithms, the new matching function proposed in this chapter, and the centroid generation policy introduced in Chapter 3 for information retrieval purposes.

In this chapter, first the evaluation measures second the matching functions to be used are introduced. This is followed by the illustration of the results of the evaluation experiments.

7.1. EVALUATION MEASURES

In this section, the measures of evaluation are illustrated. These are the following :

$$\text{Recall}_i = \frac{|D_{S_i} \cap D_R|}{|D_R|}$$

$$\text{Precision}_i = \frac{|D_{S_i} \cap D_R|}{|D_{S_i}|}$$

$$\text{Recall Ceiling (RC)} = \frac{|D_u \cap D_R|}{|D_R|}$$

$$\text{Correlation Percentage (CP)} = \frac{n_c + |D_S|}{|D|}$$

$$\text{RC for Target Clusters} = \frac{\sum_{i=1}^{n_T} |D_R \cap D_{T_i}|}{|D_R|}$$

where $|\cdot|$ indicates the cardinality of a set. The meaning of the variables used above and some additional variables are given in the following :

n_{sm} : maximum number of clusters to be selected (given as parameter).

n_s : the number of selected clusters which exhibit highest correlation with a particular query, i.e., the query under processing and $n_s \leq n_{sm}$. The inequality $n_s < n_{sm}$ can be true, since for some queries the number of clusters having a correlation greater than zero can be less than n_{sm}.

D_s : the documents of the selected clusters.

D_{s_i} : the subset of D_s ($D_{s_i} \subseteq D_s$) at ith recall level (this will be explained later).

D_R : the set of documents relevant to a particular query.

D_u : the set of documents analyzed by the user, $D_u \subseteq D_s$

(D_u is a subset of D_s, since some documents of D_s can have zero correlation with the query under processing and such documents are not presented to the attention of the user).

D : all documents of the collection.

n_c : the number of clusters for the document collection (determined by the cover coefficient concept).

n_T : number of target clusters ($n_T \leq n_{sm}$) (The target clusters are those clusters containing the largest number of relevant documents |8-Chap.13,86|. If C_i and C_j are two candidate target clusters with the same number of relevant documents and if $|C_i| < |C_j|$, then C_i will be chosen as the target cluster since its size is smaller. The number of target clusters to be considered is limited by the value of n_{sm} for experimental purposes).

D_{T_i} : the documents of the target cluster C_i.

The documents of the D_s are presented to the user in an order according to their correlation to the query which is determined by the matching function used in the particular experiment. A recall-precision pair is calculated for each consecutive document in the ranked output. In the experiments, the recall-precision values are given at some specific recall values (or recall levels), i.e., for (recall - i, i = 1,...,10) 0.1,0.2,...,1.0. The calculated recall value, R_c, is taken as R_i when $R_i \leq R_c < R_{i+1}$ and the calculated precision value corresponding to R_c is taken as P_i (i.e., as the precision value corresponding to R_i).

The recall-precision values for all queries are averaged to obtain the overall behavior for the specific experiment.

It is obvious that not all of D_R documents will appear in D_S, i.e., $|D_S \cap D_R| \leq |D_R|$ this is because $D_S \subset D$. Furthermore, some of the relevant documents may have no common terms (i.e., may have zero correlation) with the query. (This might be due to poor query formulation or due to poor indexing.) In the evaluation experiments to be followed, this is observed just for one query where in this query, the corresponding query vector was rather short). Accordingly $D_u \subseteq D_R$. Because of these reasons, recall ceiling can be less than 1. If $RC=R_j$, $j < 10$, the precision values for $R_{j+1}, \ldots, R_{10}$ are taken as P_j of the full search (this solves the averaging problem |27|).

Similar to recall ceiling, one can define the recall floor, i.e., the lowest recall value that can be observed for a query. For example, a query with 5 relevant documents will have a recall floor of 0.2(=1/5) not 0.1. Furthermore, in some queries some intermediate recall values (i.e., the recall values among recall floor and recall ceiling) may not be observed (for the query with 5 relevant documents the recall values 0.3, 0.5, 0.7, and 0.9 will never be observed). In such cases, i.e., for an unobserved recall value, R_j, the corresponding precision value P_j will be set to the value of P_k (where R_k is an observed recall value), where $k \geqslant j$ and there is no observed R_i value such that $R_j < R_i < R_k$ |106|.

The recall ceiling defined for the target clusters is used to observe the effectiveness of a clustering algorithm in putting the associated documents into the same clusters. A higher target cluster recall within a smaller subset of the document collection indicates the effectiveness of a clustering algorithm.

7.2. MATCHING FUNCTIONS

Two measures are being investigated for the matching functions. One is the coupling function and the other is a similarity measure. The coupling function comes from the coupling coefficient concept which is used in the clustering algorithms. Accordingly, one starts with an $(n_c+1) \times n$ matrix D_{qc} whose first row would be the query vector and the remaining n_c rows would correspond to the centroids of the generated clusters. The amount of coverage of the query under consideration by a cluster-i is found by the following formula (see Chapter 3 for the definition of C, S, and S') :

$$C(Q,G_i) = \sum_{j=1}^{n} S_{1j}\, S'^{T}_{ji} = \sum_{j=1}^{n} \begin{array}{l}(\text{significance of } t_j \text{ in } Q) \\ *(\text{significance of } G_i \text{ for } t_j)\end{array}$$

From the definition of S and S' matrices one can write the above formula as follows :

$$C(Q,G_i) = \frac{1}{|Q|} \left[\sum_{j=1}^{n} q_j * g_{ij} * \frac{1}{\text{COLVAL}_j} \right]$$

where $\text{COLVAL}_j = \sum_{k=1}^{n_c+1} d_{qc_{kj}}$ (i.e., sum of the jth column of D_{qc} matrix). Similarly, the amount of coverage of cluster-i by the query under processing would be

$$C(G_i,Q) = \frac{1}{|G_i|} \left[\sum_{j=1}^{n} g_{ij} * q_j * \frac{1}{\text{COLVAL}_j} \right]$$

In the above formulas $|Q|$ and $|G_i|$ indicate the query vector length and the centroid length, i.e.,

$$|Q| = \sum_{j=1}^{n} q_j$$

$$|G_i| = \sum_{j=1}^{n} g_{ij}$$

The above formulas are also valid for a weighted D_{qc} matrix.

According to the coupling function, the mutual coverage between the query and a cluster-i is defined as $C(Q,G_i) + C(G_i,Q)$.

It should be noticed that such an approach for matching function gives special attention to the rare terms. In other words, a rare term which appears in very few centroids and is also used in the query will increase the mutual coupling of the query with those centroids containing the term. This is because COLVAL is small for this kind of terms. Similarly, the centroids with fewer terms (i.e., with smaller $|G_i|$ values) will have higher coupling with the query. This will give more chance in selection to the clusters which are specialized in the information need of the query.

The other matching function uses the conventional Dice's similarity coefficient. Accordingly, the similarity between a binary centroid G_i and a query Q is defined as follows :

$$Sm(Q,G_i) = 2 * (\sum_{j=1}^{n} q_j * g_i) / (|Q| + |G_i|)$$

The definitions of $|Q|$ and $|G_i|$ are similar to the above matching function and they indicate the query vector length and the length of the centroid-i, respectively |106, p.39|.

The similarity of the query Q with a weighted G_i according to the Dice's coefficient is defined as follows |2,p.113|.

$$Sm(Q,G_i) = 1 - \frac{\sum_{j=1}^{n} abs(q_j - g_{ij})}{\sum_{i=1}^{n} (q_j + g_{ij})}$$

where, "abs" is the absolute value function.

In the above discussion, we have the phrase "weighted D_{qc} matrix" and "weighted G_i". A weighted D_{qc} matrix can occur if we have a weighted G_i (In the experiments, only binary query vectors are used).

The matching functions are used for ranking the documents and the cluster in full search and CBR. Similarly, the documents of the selected clusters are ranked by using the same functions.

7.3. THE EVALUATION EXPERIMENTS

In the evaluation experiments the papers of the ACM Transactions on Database Systems (ACM-TODS) were used to construct the document collection. The detailed information about the collection is given in Chapter 4. In the evaluation experiments the binary D matrix corresponding to the experiment T4 of Chapter 4 is used. The indexing policy for this experiment can also be seen in Chapter 3.

The characteristics of the document collection, and the generated clusters and centroids for both of the algorithms are depicted in Table 7.1 and 7.2, respectively. In the information retrieval experiments both the binary centroids and the frequency based (weighted) centroids of the clusters are used.Each entry, g_{ij}, for the frequency

Table 7.1 - Characteristics of the Document Collection

No of documents (m)	167
No of terms (n)	543
Avg no of terms/documents (X_d)	25.23
Avg no documents/term (tg)	7.76

Table 7.2 - Characteristics of the Generated Clusters and Centroids

	Algorithm-1/Algorithm-2
No of clusters (n_c)	22
Avg no of documents/cluster (d_c)	7.70
Avg no of terms used in the weighted centroids	135.18/134.59
Avg no of terms used in the binary centroids	44.36/ 13.55
No of distinct terms used in the binary centroids	444 /206
Ratio of the terms which are used in the binary centroids	0.82/ 0.38

based centroid corresponding to the cluster-i (C_i) is the frequency of term-j within the document vectors of C_i (The binary centroid generation policy corresponding to the Algorithm-1 and the Algorithm-2 can be seen in Chapter 3).

The characteristics of the queries are provided in Table 7.3

TABLE 7.3 - Characteristics of the Queries

No of queries	18
Avg query vector length	16
No of distinct documents found relevant to the queries	69
Total no of documents found relevant to the queries	146
Avg no of relevant documents for a query	8

The recall ceiling for the target clusters for both of the algorithms are given in Table 7.4. The same table also contains the proportion of the number of documents of the target clusters with respect to the whole collection size. This table states that 76% (78%) of the relevant documents are concentrated on (n_{sm} = 3) clusters containing the 26% (22%) of the documents in case of the Algorithm-1 (Algorithm-2). The above observations state that both of the algorithms are rather effective in putting the related (similar) documents into the same cluster. In doing so the Algorithm-2 is somewhat more effective, since the average recall ceiling for this algorithm is higher and is reached by considering a smaller subset of the document collection (see a/b in Table 7.4) (In the experiments $n_{T_i} = n_{sm}$ is observed for all queries, i.e., for $i = 1,...,18$.).

In the evaluation experiments both the binary centroids and frequency based centroids are used for cluster based retrieval (2 possibilities). As the matching function (i.e., for the correlation of a query and centroids) two functions which are introduced in the previous section, are used (2 possibilities). As stated previously, this gives us having the results of 8 experiments (=2 algorithms x 2 centroids x 2 matching functions).

The summary of the search statistics in terms of recall ceiling and correlation percentage is presented in Table 7.5. In this table (and also in Table 7.6), the processing codes 1, 2, 3, and 4 stand for the use of the following centroid types and matching functions : binary centroid-the coupling coefficient (BCCC), binary centroid-the Dice's coefficient (BCDC), frequency based centroid-the coupling coefficient (FCCC) and frequency based centroid-the Dice's coefficient (FCDC). Table 7.5 can be used for the evaluation of the algorithms, the centroid generation policies, and the matching functions :

TABLE 7.4 - Characteristics of the Target Clusters (n_{sm}=3)

	Algorithm-1/Algorithm-2
Avg recall ceiling for the target clusters (a)	0.76/0.78
Avg proportion of documents in the target clusters (b)	0.26/0.22
a/b	2.92/3.54

TABLE 7.5 - Summary of Search Statistics in Terms of Recall Ceiling and Correlation Percentage

Processing Code (PC)	Recall Ceiling (RC) Algo.1	Algo.2	Correl. Percentage (CP) Algo.1	Algo.2	RC / CP Algo.1	Algo.2
1 (BCCC)	0.62	0.56	0.30	0.26	2.05	2.16
2 (BCDC)	0.66	0.57	0.28	0.27	2.31	2.12
3 (FCCC)	0.65	0.63	0.31	0.28	2.10	2.21
4 (FCDC)	0.35	0.47	0.26	0.27	1.38	1.77

(a) Evaluation of the algorithms : For all processing codes the Algorithm-1 always yields a better recall ceiling, the only exception is processing code 4. However, correlation percentage (or RC/CP) is better for the Algorithm-2. This can be contributed to the iterations of the Algorithm-2 (the only exception for this observation is processing code 2). Due to the iterations of the Algorithm-2 the documents are blended among the clusters in a some what better way. However, the difference is infinitesimal. Furthermore, the Algorithm-1 might be considered preferable due to higher recall ceilings observed.

(b) Evaluation of the centroid generation policies : The centroid generation policy using the coupling coefficient concept (used in connection with the Algorithm-1) is better. Since it gives higher recall ceiling values (see the results corresponding to the processing codes 1 and 2 where only for these cases the binary centroids are used).

(c) Evaluation of the matching functions : It seems that the coupling coefficient concept and the conventional similarity measure have the same performance for binary centroids (processing codes 1 and 2). However, in case of the frequency based centroids the cover coefficient concept outperforms the similarity measure considerably (compare the RC values for the porcessing codes 3 and 4). This can be attributed to the fact that the coupling coefficient concept considers the distribution of a query term among the centroid vectors. However, the similarity concept only considers the common terms between a particular query and a document.

The recall-precision outputs for the full search and the CBR searches are given in Table 7.6-a and Table 7.6-b through Table 7.6-e, respectively.

The full search with the coupling coefficient and the Dice's coefficient begins with considerably high precision values. The precision values with the matching function using the coupling coefficient are better than the precision values with the matching function using the Dice's coefficient (six observations out of ten, where the higher -better- precision values are indicated with a bar at the left hand side of the associated precision value). This is an indication for the effectiveness of the coupling coefficient as a matching function.

The findings of the CBR searches (Table 7.6-b through Table 7.6-e) can be used for the evaluation of the algorithms, centroid generation policies, and the matching functions :

(a) Evaluation of the algorithms: The precision values associated with the Algorithm-1 are better than the precision values associated with the Algorithm-2 in processing codes 1, 2, and 3. However, the above argument is false for the processing code 4. It can be said that Algorithm-1 is preferable with respect to Algorithm-2. The effectiveness of the clustering algorithms can also be seen by a comparison with the full search results. The CBR precision values are always compatible with them, moreover the precision values associated with the Algorithm-1 are better than the full search precision values (except the results of processing code 4).

(b) Evaluation of the centroid generation policies: For this purpose one should examine the results corresponding to the processing code 1 and 2 (since for these cases binary centroids are used). Although the centroid vectors of Algorithm-1 is rather long (see Table 7.2 for a statistics of centroid vector lengths) the precision values associated with it is better (compare the recall-precision values of

TABLE 7.6 - Recall - Precision Values for the Experiments

a) Full Search		
Recall	Precision (CC)	Precision (DC)
0.1	0.8426	0.8352
0.2	0.7880	0.8399
0.3	0.7889	0.7242
0.4	0.6694	0.6743
0.5	0.6311	0.5939
0.6	0.5777	0.5113
0.7	0.4633	0.4239
0.8	0.3201	0.2946
0.9	0.2208	0.2379
1.0	0.2097	0.2207

the Algorithm-1 and Algorithm-2).In the IR literature, it is always pointed out that longer centroid vectors could lower the precision values. However, the results of the experiments are falsifying the above statement. This shows the effectiveness of the centroid generation policy associated with the coupling coefficient concept.

(c) Evaluation of the matching functions : In order to have a decision for the matching functions, one should compare the recall-precision values of the Algorithm-1 (Algorithm-2) corresponding to the processing codes 1 and 2, and similarly 3 and 4 (the processing codes 1, 2 and 3, 4 are compatible in terms of the centroids used). The performance of the matching function associated with the coupling coefficient is considerably better. This states the suitability of the coupling coefficient concept for query-centroid (document) correlation evaluation.

It is also possible to make an "all together" evaluation for the algorithms, centroid generation policies, and the matching functions. An inspection on the results shows the superiority of the algorithm (Algorithm-1), centroid generation policy, and the matching function associated with the coupling coefficient if they are used all together.

TABLE 7.6 - Recall-Precision Output for the Experiments (continued)

	b) Processing Code - 1 (BCCC)		c) Processing Code - 2 (BCDC)	
Recall	Precision (A1)	Precision (A2)	Precision (A1)	Precision (A2)
0.1	0.9028	0.8228	0.8509	0.7739
0.2	0.8041	0.7538	0.7963	0.7047
0.3	0.7594	0.6469	0.7083	0.5179
0.4	0.6762	0.5546	0.6157	0.4778
0.5	0.5251	0.4801	0.5275	0.3838
0.6	0.4291	0.3365	0.4625	0.3210
0.7	0.3890	0.2854	0.4000	0.2756
0.8	0.2672	0.2138	0.3067	0.2269
0.9	0.2614	0.2138	0.2811	0.2269
1.0	0.2347	0.2138	0.2526	0.2269

	d) Processing Code - 3 (FCCC)		e) Processing Code - 4 (FCDC)	
Recall	Precision (A1)	Precision (A2)	Precision (A1)	Precision (A2)
0.1	0.9398	0.8611	0.7094	0.7937
0.2	0.8741	0.8374	0.6130	0.6248
0.3	0.8025	0.7277	0.4641	0.5106
0.4	0.7062	0.5520	0.4192	0.4538
0.5	0.5929	0.5237	0.3311	0.3885
0.6	0.4291	0.4361	0.2687	0.3172
0.7	0.3548	0.3431	0.2478	0.2624
0.8	0.2672	0.2770	0.2207	0.2570
0.9	0.2614	0.2264	0.2207	0.2388
1.0	0.2347	0.2079	0.2207	0.2207

7.4. SUMMARY AND CONCLUSIONS

In this chapter the information retrieval behavior of the two partitioning type clustering algorithms are analyzed. In order to deal with the first order variables a single level clustering is used.

In the evaluation experiments, a collection of 167 documents and 18 queries were used. The characteristics of the document collection, queries, and the generated clusters were illustrated.

In the evaluation, the clustering algorithms, the centroid generation policy associated with the Algorithm-1, and the matching function associated with the cover coefficient concept were found effective. Although the Algorithm-1 is a one shot algorithm (i.e., no iterative optimization is associated with it) its performance is superior to the performance of the Algorithm-2. The evaluation experiments show the validity of the cover coefficient concept for clustering and the validity of the Algorithm-1 for information retrieval.

8. ADDITIONAL CONCEPTS/METHODOLOGIES THAT UTILIZE COVER COEFFICIENT CONCEPT

In this chapter various uses of the cover coefficient concept will be introduced. The concepts/methodologies of this chapter are the by products of the cover coefficient (decoupling/coupling coefficient) concept. The introduced concepts/methodologies can be used in an advanced experimental IR system.

In this chapter, firstly, a methodology for showing the correspondence of document and term clusters is proposed. In information retrieval literature, this correspondence is usually assumed. However, there is no study to show this correspondence. Therefore, the methodology proposed here is of pioneering nature.

In the second section the concept called "term discriminator" is revisited and a new measure based upon the cover coefficient concept is proposed. The new metric is computationally cheaper and very meaningful. In connection with this, in the third section, an iterative algorithm for finding the optimum weights of indexing terms is illustrated. The algorithm utilizes the user assigned weights as input and optimizes them by using the new concepts of term discrimination value and the identical concept for documents, which is the document discrimination value.

Section four presents the use of the C' matrix for the construction of effective indexing vocabularies. In the methodology proposed, very frequent and infrequent terms are mapped into terms which have better indexing capability.

8.1. ON THE CORRESPONDENCE OF DOCUMENT AND TERM CLUSTERS

In Chapter 3 it is shown that the number of document clusters (n_c) and the number of term clusters (n'_c) are identical if one employs the cover coefficient concept for determining the number of clusters within a collection ($n_c = \sum_{i=1}^{m} \delta_i = \sum_{i=1}^{n} \delta'_i$). This property also implies that the clustering of documents (terms) is nothing but also the clustering of terms (documents). A similar property is observed in Crouch's algorithm for clustering of documents |27| (This algorithm is briefly introduced in Section 3.3.6). However, in Crouch's case the clustering process was order dependent and there was no analytical formula for the number of clusters.

Testing of the correspondence of term and document clusters can be done in various ways. Two of them are presented in the following :

First approach :

a) Obtain the document clusters : D_c

b) Obtain the term clusters : T_c

c) Find the term clusters implied by the document clusters : T'_c For this purpose (i.e., as T'_c), the document cluster centroids can be used.

d) Find the document clusters implied by the term clusters : D'_c Similar to step-c, for this purpose, the term cluster centroids can be used. Notice that, definition of term cluster centroids will be made by using the documents.

e) Find an association measure between D_c:D'_c and T_c:T'_c

Second Approach :

a) Obtain the document clusters : D_c.

b) Obtain the term clusters : T_c.

c) Recluster the documents by assigning them to term clusters : D'_c.

d) Recluster the terms by assigning them to document clusters : T'_c.

e) Find an association measure between D_c:D'_c and T_c:T'_c.

In similarity association, one should use a method like the Rand's or Goodman-Kruskal's coefficient because one cannot assume a one to one correspondence between the clusters of D_c:D'_c and T_c:T'_c. In the first approach, the clusters D'_c and T'_c will be overlapping type. The Goodman-Kruskal's coefficient is defined for partitions. Therefore it cannot be used. The Rand's coefficient is also defined for partitions, however, it can be used for overlapping clusters after some minor modifications |80|. In the second approach, D'_c and T'_c are the partitions. Therefore, both the Rand's and Goodman-Kruskal's coefficients can be used directly.

In the second approach, during document assignment process (step-c) one could apply the cover coefficient concept again. A document to be clustered can be assigned to the term cluster which covers it maximally. For this purpose construct a matrix (say matrix A) whose first n_c rows correspond to term clusters. The entry a_{ij} is 1 if term-j is assigned to cluster-i. The row numbers n_c+1 to n_c+m correspond to the rows of the original D matrix, where each row of the D matrix corresponds to a document. In constructing the clusters of D'_c, a document will be assigned to the term cluster which covers it maximally, i.e., assign document-i to the term cluster-j where j is the second index of the entry chosen by the maximum function $\max\{C_{n_c+i,1}, C_{n_c+i,2},\ldots,C_{n_c+i,n_c}\}$. It is assumed that a C matrix is created from the A matrix. A similar approach can be used for constructing the T_c'.

It is obvious that a high degree of association between $D_c:D_c'$ and $T_c:T_c'$ is also an indication for the correspondence of term and document clusters.

8.2. THE USE OF THE COUPLING COEFFICIENT CONCEPT FOR TERM DISCRIMINATION PURPOSES

The study for finding the importance of indexing terms is one of the leading activities of information retrieval field. The metric called "term discrimination value" is well known |88,92|. This metric is usually computed by first creating a centroid, G, for the document collection. Each entry of G, g_j, (j=1,...,n) is then defined as the average weight of term-i in all of the m documents :

$$g_j = \frac{1}{m}\sum_{i=1}^{m} d_{ij}.$$

The centroid G is used to calculate the space density function, Q, defined simply as the sum of the similarity coefficients between centroid G and all documents D_i, that is,

$$Q = k\sum_{j=1}^{m} S(G,D_j). \qquad (0 \le S \le 1,\ 0 \le Q \le m, \text{ where } k \text{ is a constant})$$

The space density of the collection is then calculated once more, after removing term-k from all document vectors, call this new Q the Q_k. If t_k is a good discriminator then this will lead to $Q_k > Q$. Since the deletion of a term, which makes documents more distinct, will increase the similarity among documents. For poor discriminators the reverse will be true. The discrimination value of t_k can be taken as $DV_k = Q_k - Q$ |88,92|. According to this metric, it is seen that the best terms have average document frequency-neither too high nor too low-. The bad terms (discriminators) have high collection frequency, and are present in most documents of a collection. Zero discrimination values are obtained for very low frequency terms |92, p.69|.

A very efficient algorithm that computes discrimination values of terms is proposed by Crawford |22|. This algorithm requires 5t, t, 2t, 2t, 4t multiplications, additions, divisions, square root operations, and subtractions, respectively (where t is the total number of term assignments, i.e., $t = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij}$ in terms of our D matrix-see Chapter 3-).

In the remainder of this section, a method for using the coupling coefficient concept for term discrimination will be proposed and the findings of an experiment will also be presented.

According to the coupling coefficient concept, the number of clusters within a collection can be found as follows :

$$n_c = \sum_{i=1}^{m} \delta_i = \sum_{i=1}^{m} \alpha_i * (d_{i1} * \beta_1 + d_{i2} * \beta_2 + \dots + d_{in} * \beta_n)$$

(The definitions for α, β and the derivation of the formula for n_c can be found in Chapter 3.) If a term, t_ℓ, is deleted from the description of documents, its contribution for n_c can also be deleted to obtain the new n_c, $n_{\ell c}$, as follows :

$$n_{\ell c} = \sum_{i=1}^{m} \alpha_i^\ell * (d_{i1}*\beta_1 + d_{i2}*\beta_2 + \ldots + d_{i,\ell-1}*\beta_{\ell-1} + d_{i,\ell+1}*\beta_{\ell+1} + \ldots + d_{in}*\beta_n)$$

where α_i^ℓ indicates that in the description of d_i (i=1,...,m) the existence or t_ℓ is omitted. In other words, $\alpha_i^\ell = (\sum_{\substack{j=1 \\ j\neq\ell}}^{n} d_{ij})^{-1} = (\alpha_i^{-1} - d_{i\ell})^{-1}$. A way of calculating the $n_{\ell c}$ is the following

$$n_{\ell c} = \sum_{i=1}^{m} \alpha_i^\ell * \left(\frac{\delta_i}{\alpha_i} - d_{i\ell} * \beta_\ell \right)$$

In the above formula the term $-d_{i\ell} * \beta_\ell$ eliminates the contribution of term-ℓ on $n_{\ell c}$ via its individual term generality ($\beta_\ell = 1/t_{g_\ell}$). δ_i/α_i eliminates the contribution of term-ℓ on $n_{\ell c}$ due to its effect on depth of indexing ($\alpha_i = 1/x_{d_i}$); α_i^ℓ reintroduces the effect of the modified depth of indexing. It is obvious that the above formula for $n_{\ell c}$ is extremely inefficient. Notice that not all of the documents contain the deleted term, t_ℓ. After this observation one may use the value of n_c to calculate $n_{\ell c}$.

$$n_{\ell c} = n_c + \sum_{i=1}^{t_{g_\ell}} \left[\alpha_i^\ell * \left(\frac{\delta_i}{\alpha_i} - d_{i\ell} * \beta_\ell \right) - \delta_i \right], \forall\, d_i \varepsilon D_\ell$$

where, $t_{g_\ell} = |D_\ell|$ and $D_\ell = \{d_i \mid d_i \varepsilon D \text{ and } d_{i\ell} \neq 0\}$, i.e., t_{g_ℓ} is the generality of term-ℓ and D_ℓ is the set of documents which uses t_ℓ. In the above formula the contribution of a document on the number of clusters is deleted (by term-δ_i) and then the contribution of the same document is reintroduced by disregarding the existence of term-ℓ. (If the D matrix is weighted, then one should replace $d_{i\ell}$ with $d_{i\ell}^2$ in the above formula -refer to the general formula for C_{ij}.)

By deleting term-ℓ, it is expected that there will be a change in the number of clusters. It is expected that the deletion of the terms which has more importance (better discriminators) would decrease the number of clusters by decreasing the decoupling coefficient of the documents and/or terms, i.e., $n_{\ell c} < n_c$ (Notice that the number of clusters is given by the product of the number of documents-terms- with the overall decoupling of the documents-terms-, i.e., $n_c = m * \delta = n * \delta'$). The decrease in the number of clusters indicates that the documents (terms) of the D matrix are more similar to each other. Conversely, the deletion of the terms with less significance, or correspondingly worse discriminators, would increase the number of clusters, i.e., $n_{\ell c} > n$. The terms which do not have any significance for document description (i.e., the non-discriminators) would not change the number of clusters in the collection, i.e., $n_{\ell c} \cong n_c$.

After the above explanation, the term discrimination value for term-ℓ, DV_ℓ, would be defined as follows :

$$DV_\ell = \frac{n_c}{n_{\ell c}}$$

By this formula, better, worse, and non-discriminators will have a DV value of $DV > 1$, $DV < 1$, and $DV \cong 1$, respectively. The value of $DV_\ell > 0$, since n_c and $n_{\ell c}$ (ℓ=1,...,n) are always greater than zero.

The computational aspects of the new metric will be illustrated in terms of the computation of n_c, $n_{\ell c}$, and DV_ℓ. For this purpose the total number of operations (additions, subtractions, multiplications, and divisions) will be computed :

a) The computation of n_c : α's and β's require 2*m*n additions with an appropriate data structure (like inverted file this) this will reduce to 2t (i.e., the total number of term assignments) and m and n division, respectively. The summation for n_c will require (t + m) additions and (m + t) multiplications (t counts for $d_{ij} * \beta_j$, where D is assumed as weighted).

b) The computation of $n_{\ell c}$: in this computation all $n_{\ell c}$ values (ℓ=1,...,n) can be initialized as n_c and then the $n_{\ell c}$ values can be found by considering the documents one by one, i.e., the effect of a document is reflected on each $n_{\ell c}$ for the terms used by this document. This means that α_i^{ℓ} and δ_i/α_i will be computed once. Where they involve 2 divisions, 1 subtraction and 1 division, respectively. The computation of $n_{\ell c}$ involves 1 addition, 2*X_{davg} subtractions, 2*X_{davg} multiplications (D is assummed as weighted), and X_{davg} additions. X_{davg} is the average depth of indexing (i.e., t/m). For all documents this makes (m+t) additions, (m+2t) subtractions, 2t multiplications, and 3m divisions.

c) The computation of DV_{ℓ} : involves n divisions (for ℓ=1 to n).

The overall computational requirement is as follows :

Additions	: (3t+m)	+ (m+t)	+ (0)	=	4t + 2m
Subtractions	: (0)	+ (m+2t)	+ (0)	=	2t + n
Multiplications	: (m+t)	+ (2t)	+ (0)	=	3t + m
Divisions	: (m+n)	+ (3m)	+ (n)	=	4m + 2n

If we assume a weighted D matrix the additional cost due to this will be t and t multiplications, respectively, for the computation of n_c and $n_{\ell c}$ (Notice that these new costs count for $d_{ij} * \beta_{ij}$).

A comparison of the computation of term discrimination values with respect to cover coefficient and similarity concepts |22| is given in Table 8.1. In this table the value of $t \gg m$ and n. Hence, the computational requirements for additions, subtractions, multiplications, and divisions can be approximated as 4t, 2t, t, and 0, respectively. Furthermore, the method proposed does not require any square root operation which inherently involves iterations requiring multiplications and divisions.

TABLE 8.1 - The Computation Involved for the Term Discrimination Values with Respect to Cover Coefficient (DV_1) and Similarity (DV_2) Concepts.

Operation	DV_1	DV_2
Add.	4t(+2m)	t
Subt.	2t(+m)	4t
Mult.	3t(+m)	5t
Div.	(4m+2n)	2t
$\sqrt{\ }$	-	2t

The overall complexity of the computation can be given in terms of the multiplications and divisions involved. Accordingly, the complexity of the whole operation would be O(t) or more precisely O(3t). The computational simplicity of the method and its very evident meaning deserves consideration to be checked as a measure for term discriminator. If we assume a binary D matrix, the multiplication for the $d_{ij} * \beta$ would drop. This would decrease the number of multuplications from 3t to t (t number of less multiplications both for n_c and $n_{\ell c}$). The same assumption for the discrimination value tion based on the similarity concept would also decrease the number of multiplications from 5t to 3t (see equation-12 of |22|). The compu computational requirement of DV in terms of cover coefficient concept is always less than the one based on similarity concept.

8.2.1. An Experiment for Checking the Consistency of the Two Term Discriminator Measures

In this section an experiment for checking the consistency of the two term discriminators will be illustrated. For the experiment a FORTRAN program has been written and the term discrimination values are computed according to the Crawford's algorithm |22| and the proposed new method using the cover coefficient concept for all terms of a binary D matrix. After this, the terms are sorted according to their discrimination values. The consistency of the two measures can be compared by checking the rank of the terms in the two sorted lists. It is obvious that, the overall ranks of the term groups according to these two metrics might be rather consistent; however, any two terms may not always have exactly the same rank in the two lists. Therefore, it would be very strict to expect for equivalent ranks. Hence, it is better to use a range of ranks to check the consistency. This will be illustrated in the experiment which follows.

The experiment is performed on the D matrix of the T4 experiment discussed in Chapter 4. The D matrix for the T4 experiment consists of 167 documents and 543 terms (The program which creates the D matrix, GENMAT, is firstly written on an IBM machine by FORTRAN. The GENMAT is translated into FORTRAN77 for the Burroughs machine which is available at the METU. Because of this reason, there is a slight difference on the D matrix which is used in this experiment-the number of terms is 543 in the D matrix of this experiment while it was 539 in the D matrix of the T4 experiment-. However, this difference in the D matrix does not change the results of the T4 experiment). The rank difference comparison of the two metrics is given in Table 8.2.

TABLE 8.2 - The Rank Difference Comparison of the Two Term Discrimination Measures for the D Matrix of the T4 Experiment.

Absolute Rank Difference	No of Observations	% of Observations	No of Observ. ≤ Rank Difference	% Terms of ≤Rank Dif.
0	12	2.9	16	2.9
1- 5	143	26.3	159	29.2
6-10	94	17.3	253	46.5
11-15	56	10.3	309	56.8
16-20	57	10.5	366	67.3
21-25	35	6.4	401	73.7
26-30	26	4.8	427	78.5
31+	116	21.4	543	100.0

Table 8.2 states that 16 terms (2.9%) of the total 543 terms have exactly the same ranks; 143 terms have rank differences of 1 to 5 which makes the 26.3% of all the terms, and 159 terms (29.2%) have a rank difference less than or equal to 5. The table also shows that 309 terms (56.8%) have a rank difference less than or equal to 15.

The average rank difference for all terms is also calculated and found as 19.39.

Another way of checking the consistency is to compare the average rank of "k" terms at the beginning and at the end of the one of lists and to compare this average with the average rank of the same terms in the other list |88|. This is done for k=25 and depicted in Table 8.3. In Table 8.3 DV1 and DV2 indicate the term discrimination values, for cover coefficient and similarity concepts; respectively. The average rank for the first 25 terms is obviously 12.5, for the last 25 terms (terms 519 through 543) it is 531. The first 25 terms of the discrimination measure according to the cover coefficient concept has an average rank of 13.6 in the measure according to the similarity concept. From all the entries of Table 8.3 it is observed that the term ranks are rather consistent for the first and last 25 terms.

TABLE 8.3 - Comparison of Average Rank for the First 25 and Last 25 Terms of the D Matrix for the T4 Experiment.

	DV1 (cover coef.)	DV2 (simil.)
First 25 terms	12.5	13.6
Last 25 terms	531.0	521.1
First 25 terms	14.4	12.5
Last 25 terms	520.0	531.0

In the experiments it is observed that there is an inverse correlation between the generality of a term ($tg_j = \sum_{i=1}^{m} d_{ij}$) and its value as a discriminator. The terms with high generality (with a maximum of 29) are bad discriminators, the terms with the average generality (about 6) are non-discriminators, and the terms with low generality (with a minimum of 3) are good discriminators. This is the reverse of what is stated in the previous publications |22,88,92|, since the situation observed for the terms with the average generality and low generality should be reversed. No correlation is observed between the decoupling coefficient of a term (δ_i', i=1,...,543) and its value as a discriminator. This is valid for both of the discrimination measures. However, in an experiment with larger D matrix one would expect to observe the terms with the average frequency and with the average decoupling value as good discriminators.

The experiment depicted above shows the compatibility of the term discrimination value based on the cover coefficient concept with respect to the term discrimination value based on the similarity concept. The method proposed in this section requires less computation. This properties make the new approach attractive for use in the IR systems.

8.3. AN APPROACH FOR FINDING THE OPTIMUM WEIGHTS FOR INDEXING TERMS

In this section an iterative algorithm for finding the optimum weights of indexing terms is illustrated. The algorithm utilizes the user assigned weights as the input and optimizes them by using the concepts of term and document discrimination values.

In the previous studies, it was observed that the better term weighting strategies will decrease the similarity among the documents of a collection. Such strategies will make the clusters more separated from each other, while at the same time will make the members of a cluster more close to each other |87,88,91,92|. In other words, better indexing strategies will increase the homogeneity within clusters and the heterogeneity between clusters since the document clusters are based on a concept of cohesion among the members. The algorithm to be defined in this section will use the coupling coefficient concept.

For the implementation of the method, a concept called "significance value" of the documents will be introduced. Similar to a term, deletion of a document may change the number of clusters. After the deletion of a document, d_k, in order to find the new number of clusters, n_{kc}, for the collection, a computation similar to the one illustrated in Section 8.2 will be performed. This computation is illustrated in the following :

$$n_c = \sum_{i=1}^{n} \delta_i' = \sum_{i=1}^{n} \beta_i (d_{1i}\alpha_1 + d_{2i}\alpha_2 + \dots + d_{mi}\alpha_m)$$

(The definitions for α, β and the derivation of the formula for n_c can be found in Chapter 3). If a document, d_k, is deleted from the document collection, its contribution on n_c can be deleted as follows to obtain the new n_c (n_{kc}) :

$$n_{kc} = \sum_{i=1}^{n} \beta_i^k (d_{1i}\alpha_1 + d_{2i}\alpha_2 + \dots + d_{k-1,i}\,\alpha_{k-1} + d_{k+1,i}\alpha_{k+1} + \dots + d_{mi}\alpha_m)$$

where β_i^k indicates that on the description of t_i (i.e., the jth column of the D matrix) the existence of d_k is disregarded, i.e., $\beta_i^k = (\sum_{\substack{\ell=1 \\ \ell\neq k}}^{m} d_{\ell i})^{-1} = (\beta_i^{-1} - d_{ki})^{-1}$. Similar to $n_{\ell c}$ which is found for the terms, an efficient way of calculating n_{kc} by using the n_c is the following :

$$n_{kc} = n_c + \sum_{i=1}^{X_{d_k}} \left[\beta_i^k * \left(\frac{\delta_i'}{\beta_i} - d_{ki} * \alpha_k \right) - \delta_i' \right], \forall\, t_i \varepsilon d_k$$

where the summation is done for the terms which appears in the description of the document-k and X_{d_k} is the depth of indexing for document-k.

Similar to the $n_{\ell c}$ case, $-d_{ki} * \alpha_k$ eliminates the direct contribution of document-k via its depth of indexing ($\alpha_k = 1/X_{d_k}$); δ_i'/β_i eliminates the contribution of document-k due to its effect on individual term generalities ($\beta_i = 1/t_{gi}$); β_i^k re-introduces the effect of deletion of document-k on individual term generalities.

Like that of the term discrimination concept, it is expected that the deletion of the significant documents would decrease the number of clusters by decreasing the decoupling coefficient of the documents and/or terms, i.e., $n_{kc} > n_c$ (Notice that the number of clusters is given by the product of the number of documents-terms-with the overall decoupling of the documents-terms, i.e., $n_c = m * \delta = n * \delta'$). The decrease in the number of clusters indicates that the documents (terms) of the D matrix are more similar to each other. Conversely, the deletion of the documents with less significance would increase n_{kc}, i.e., $n_{kc} > n_c$. The documents which do not have any significance in the collection would not change the number of clusters, i.e., $n_{kc} \cong n_c$.

After the above explanation, one could define the document significance value for document-k, SV_k, as follows :

$$SV_k = \frac{n_c}{n_{kc}}$$

The interpretation of the SV is like that of DV. In other words, $SV > 1$, $SV < 1$, and $SV \cong 1$ will correspond to more, less, and non-significant documents, respectively. It is obvious from the definition of SV that,it is the counter part of DV.(For a document one can not mention about its discrimination value, but one can say the significance of a document. For that reason the concept "significance value" is introduced.)

For the modification of the weight of t_j in d_i (i.e., d_{ij}) the product $SV_i * DV_j$ will be used.

$$SV_i * DV_j = \frac{n_c}{n_{ic}} * \frac{n_c}{n_{jc}} = \frac{n_c^2}{n_{ic} * n_{jc}}$$

The above product will be referred to as "weight modification factor" for the weight of term-j in document-i and it will be shown by Δ_{ij}. The D matrix will be redefined as follows :

$$d_{ij} = \Delta_{ij} * d_{ij} \qquad \text{for} \quad i=1,\dots,m \quad \text{and} \quad j=1,\dots,n.$$

The rationale of using the above approach for term weight modification is the following :

a) $n_{ic} < n_c$, $n_{jc} < n_c$: occurence of a good discriminator (t_j) in a significant document (d_i) should be more important than the occurrence of a worse discriminator- $-DV_\ell < DV_j-$ (better discriminator-ℓ $-DV_\ell > DV_j-$) in a more significant document-k $-SV_k > SV_i-$ (in a less significant document-k $-SV_k < SV_i-$).

b) $n_{ic} < n_c$, $n_{jc} < n_c$: occurrence of a bad discriminator (t_j) in a significant document (d_i) should be more important than the occurrence of a bad discriminator (like term-j) in a less significant document-k $-SV_k < SV_i-$.

c) $n_{ic} > n_c$, $n_{jc} > n_c$: occurrence of a good discriminator (t_j) in a non-significant document (d_i) should be more important than the occurrence of a worse discriminator-ℓ $-DV_\ell < DV_j-$ nonsignificant document like d_i.

d) $n_{ic}\ \ n_c$, $n_{jc}\ \ n_c$: occurrence of a bad discriminator (t_j) in a non-significant document (d_i) should be less important than the occurrence of a bad discriminator in a significant document or the occurrence of a good discriminator in a bad document.

The product $SV_i * DV_j$, i.e., Δ_{ij} will satisfy the above requirements.

The optimum weights of the indexing terms will be found by an iterative method. The following algorithm will be used for this purpose.

```
sc = .false. /*stopping criterion*/
repeat ;
  Find $DV_\ell$ ($\ell$=1,...,n), $SV_k$ (k=1,...,m);
  $d_{ij} = \Delta_{ij} * d_{ij}$ (i=1,...,m; j=1,...,n);
  Set sc according to the new D matrix definition;
until  sc ;
```

In each iteration of the above algorithm it is expected that there will be a feedback from the previous state of the D matrix and this feedback process will continue till D reaches to the steady state condition which is indicated by setting the sc (stopping criterion) to true. (see Figure 8.1).

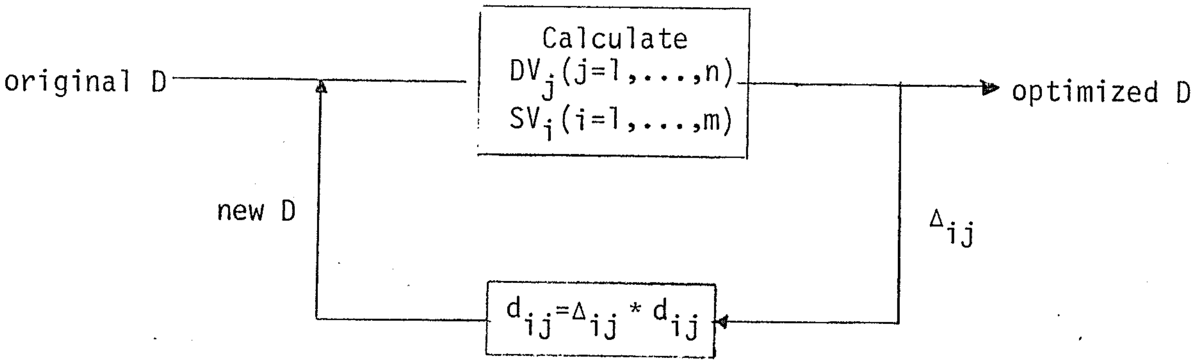


Figure 8.1 - The feedback process in modifying the D matrix

At the iteration step that obtains the optimum, it is expected that the average coupling of the cluster members with the corresponding cluster seed (Ψ_c) and the normalized coupling of the cluster members (Ψ_N) will reach a steady state. Ψ_c and Ψ_N are defined as follows :

$$\Psi_c = \frac{\sum_{j=1}^{n_c} \sum_{i=1}^{n_{c_j}} c_{ij}}{\sum_{j=1}^{n_c} n_{c_j} - n_c} = \frac{\sum_{j=1}^{n_c} \sum_{i=1}^{n_{c_j}} c_{ij}}{n - n_c}$$

$$\Psi_N = \frac{\Psi_c}{\Psi}$$

In the above formula, n_{c_j} is the number of documents in the jth cluster ($d_i \neq d_j$, cluster seeds are assumed as the natural members of the clusters, where d_i is the document to be clustered and d_j is the cluster seed).

The increase in Ψ_c means that documents within a cluster will be more and more similar to the cluster seeds and consequently more similar to the other members of the clusters. The increase in Ψ_N is not only due to the increase in Ψ_c but also due to the decrease in Ψ. Since collection documents become less and less similar to (or decoupled from) each other.

The stopping criterion of the algorithm can be one of the following :

a) The number of clusters of two consecutive iterations are equal to each other.

b) The documents chosen as the cluster seeds in two consecutive itreations are identical, or at least have a number of common seeds greater than a given threshold.

c) If one sorts the significance value of the documents in two consecutive iterations, then the ranks of the documents in two consecutive iterations should be compatible. For this purpose a rank consistency measure like that of the previous subsection can be utilized.

d) Generate the cluster of documents after each step and check the stability of the clustering process according to a cluster stability criterion |2,37,82| and/or observe Ψ_c and Ψ_N of the clusters until they reach a steady state.

The complexity of the weight optimization algorithm can be found as follows :

Computation of δ_i $(i=1,\ldots,m)$: $O(m*n_{avg})$, which is also equal to $O(t)$.

Computation of δ'_i $(i=1,\ldots,n)$: $O(m*n_{avg})$.

Computation of n_{kc} $(k=1,\ldots,m)$: $O(m*n_{avg})$.

Computation of $n_{\ell c}$ $(\ell=1,\ldots,n)$: $O(m*n_{avg})$.

Computation of new d_{ij} $(i=1,\ldots,m\ ;\ j=1,\ldots,n)$: $O(m*n_{avg})$.

Computation of the stopping criterion : The first three criteria are less expensive therefore preferable. The computation of n_c is very easy. The computations of (b) and (c) firstly require the calculation of seed powers and significance values of the documents. The computational complexity of the seed powers is $O(m)$, the significance values are already taken into consideration. The sorting activity will have a complexity of $O(m*\log m)$. The overall complexity for one iteration is then

$$O(5*m*n_{avg}) + \{0 \text{ or } O(m) \text{ or } O(m\log m)\}.$$

8.4. THE USE OF THE C' MATRIX FOR THE CONSTRUCTION OF EFFECTIVE INDEXING VOCABULARIES

It is known that not all of the indexing terms are good for document representation |88, 92, 106|. Because of this reason, it is necessary to make some transformations on index terms to make them better document descriptors |88|. For this purpose, the following two approaches are used :

a) Try to use very frequent terms to construct a higher order term or continuous word phrase. This means that if t_i and t_j are very frequent terms (t_{g_i} , $t_{g_j} >> t_g$), -where t_{g_i} (t_{g_j}) and t_g indicate the generality of term-i (term-j) and the average term generality, respectively- then do not use them in the D matrix, but use a new term t_{ij} instead of them. Obviously, there should be a considerable association between t_i and t_j for the construction of t_{ij} (This concept can be used among n number of terms, where $n \geq 2$). The terms like t_{ij} will be called "composite terms" in this section (the sequence of the terms in a composite term is immaterial, i.e., AB = BA).

b) The use of very infrequent terms also does not have a good contribution to the effectiveness of an IR system. Instead of using these terms a new "virtual term" is introduced. Very infrequent terms, which are assigned to the same term cluster will be represented by a single "virtual term".

The above process are referred to as "right-to-left" and "left-to-right" transformations, respectively |88, p.43|. The reason for this is depicted in Figure 8.2. By left-to-right transformation, low

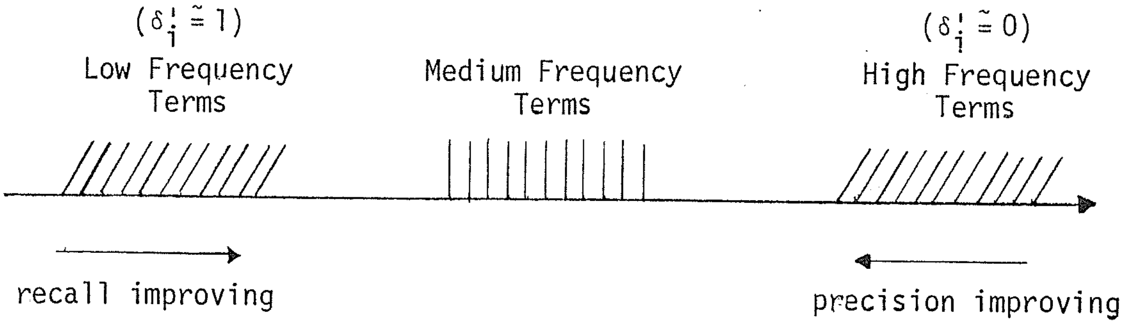


Figure 8.2 - Term characterization in document frequency ranges.

frequency terms will have medium frequency. The same argument is valid for the right-to-left transformation. The effect of these processes on recall and precision are also shown in the figure.

If one considers the decoupling coefficients, δ_i' (i=1,...,n), of the terms, it will be observed that very frequent (infrequent) terms have a very low (high) δ_i' value $(0 < \delta_i' \leq 1)$. For the above vocabulary transformation one can use the following methods.

Treatment of very frequent terms :

a) Obtain δ_i' of all terms, i.e., i=1,...,n.

b) Sort the terms in ascending order according to their δ_i' values. By this way very frequent terms will appear at the beginning of the list. Select first n_d number of terms for deletion (n_d can be specified beforehand or can be found experimentally).

c) Resort the selected terms according to their seed power, $p_i' = \delta_i'\Psi_i'\beta_i^{-1}$, where β_i^{-1} is the generality of term-i.

d) The consecutive terms of this newly sorted list can be used for constructing the composite terms. Assuming that the terms t_i and t_j are consecutive, in order to create t_{ij} from these two, the following conditions should be *satisfied : $p'_i \cong p'_j$, $\delta'_i \cong \delta'_j$, $c'_{ij} \cong c'_{ji}$. Furthermore, the mutual coverage of the terms (c'_{ij} and c'_{ji}) should have a value comparable with δ'_i (δ'_j).*

Notice that step-c will put the terms which have the same or close seed power consecutively into the resorted list. If the consecutive terms, satisfy the conditions set forth in step-d, then this means that the terms have the same pattern of distribution among the documents. If c'_{ij}, c'_{ji} have a value comparable with δ'_i (δ'_j) this means that, these are assigned to the same set of documents. Therefore, they can be used for constructing a composite term t_{ij}. Although the above conditions are given for two terms they are easily for more than two terms.

Treatment of very infrequent terms :

a) Obtain the δ'_i values for all terms, i.e., $i=1,...,n$.

b) Sort the terms in descending order according to their δ'_i values. The terms with very high decoupling coefficient and with very little indexing capability will appear at the beginning of this list.

c) Obtain the term clusters.

d) The terms which appear at the beginning of the sorted list will be mapped into "virtual terms" if they appear in the same term cluster.

9. SUMMARY AND CONTRIBUTIONS OF THE THESIS AND DIRECTIONS FOR FUTURE RESEARCH

9.1. SUMMARY

The thesis, firstly presents the information systems and in particular information (document, text) retrieval systems. This presentation also contains the information explosion problem, the evaluation criteria and the hardware solution for information retrieval systems.

After this, the concept of clustering and a model for seed oriented clustering are presented. This is followed by the illustration of two partitioning type clustering algorithms. Both algorithms use the "seed power" concept for selecting the cluster seeds; however, the assignment of documents to the seeds is different. The first algorithm uses the new "cover coefficient" concept and it is a single-pass algorithm. The second one uses the Dice's similarity measure for document assignment to the cluster seeds and is a multi-pass algorithm. A new centroid generation method in connection with the cover coefficient concept and an illustration of both of the algorithms are also presented. A cluster maintenance algorithm for dynamic (expanding) document environments is presented. The thesis also includes the complexity analysis of the algorithms.

The similarity and stability analysis of the clustering algorithms are presented with respect to two different metrics. A set of experiments using a document collection consisting of the titles, abstracts, and keywords of 167 ACM TODS publications are performed and the findings of them are presented in detail.

In the experiments a close relationship is observed between the policy of indexing and the clustering behavior. These relationships are formulated by using the concepts of the indexing theory, such as depth of indexing and term generality (In the thesis, these relationships are referred to as "indexing-clustering association" basic relationships.).

For the text retrieval operations a set of new instructions are introduced into the RAP database machine. The implementation of the typical text retrieval operations with the RAP programs incorporating the new text retrieval instructions is presented. Performance of the RAP database machine during the execution of the new instructions is examined. In the RAP.3 architecture the term matcher and query resolver subsystems of a text retrieval computer are combined into one.

A model for integrated fact/document information systems, which aims to synthesize a relational DBMS and IRS capable of context sensitive full text searches, is presented. The IR system relies on a clustering subsystem for database partitioning and relation fragmentation. The support architecture for the context sensitive search operations is the RAP.3 database machine.

A comparative performance analysis of the single-pass and the multi-pass clustering algorithms in information retrieval is presented. The performance of both of the algorithms validates their usage for information retrieval purposes.

Various new concepts/methodologies based on the cover coefficient concept are introduced. These are the following: (a) a methodology showing the correspondence of document and term clusters; (b) a computationally cheaper method for "term discrimination" value; (c) an iterative algorithm for finding the optimum weights of indexing terms; and (d) the use of the C' matrix for the contruction of effective indexing vocabularies.

9.2. CONTRIBUTIONS OF THE THESIS

The major contributions of the thesis can be summarized as follows :

(a) Some new concepts for clustering algorithms are introduced; these are : cover coefficient, decoupling coefficient (δ), coupling coefficient (Ψ), and cluster seed power (p).

(b) The identification of robust clustering procedures for the partitioning type clustering algorithms is important, and the thesis provides a novel algorithm for doing this by using the cover coefficient and cluster seed power concepts. Another clustering algorithm has been developed to show the validity of the use of the cover coefficient concept for clustering purposes.

(c) A new algorithm for the maintenance of clusters in dynamic (expanding) document collections is proposed. This algorithm also uses the cover coefficient and cluster seed power concepts.

(d) A novel centroid generation policy is introduced in connection with the cover coefficient concept. A new methodology to test the goodness of centroid generation policy is also presented.

(e) A systematic evaluation of the stability and similarity of two clustering algorithms is presented. This kind of evaluation of clustering algorithms is usually overlooked in the literature.

(f) A set of dependencies, which are referred to as "indexing-clustering association" basic relationships, are observed in the experiments. These observations state close relationship between the policy of indexing and the clustering behavior. Such relationships have not yet been encountered in the literature.

(g) The RAP.3 database machine is enhanced with the new text retrieval instructions. Context sensitive free text retrieval operations are implemented by using the new instructions. It is observed that the RAP.3 database machine when used in the IRS architecture will considerably enhance the system efficiency; furthermore, it will bring the capability of DBMS operations into the system. Hence the system will easily become an integrated IR and DBMS.

The changes made in the RAP.3 is such that its performance in the text retrieval operations is favorably comparable with the other existing architectures.

(h) The thesis presented a new model for the integration of fact and document retrieval systems. Unlike the previous studies aimed at this purpose, which more or less reduce one system into the other, the proposed model aimed to accomplish this integration by a synthesis of the techniques and methadologies of both systems.

(i) Additional concepts/methodologies are introduced in connection with the cover coefficient concept.

9.3. DIRECTIONS FOR FUTURE RESEARCH

In this section, a list of future research possibilities will will be given

a) A proposal for cluster validity checking :

Testing cluster validity is one of the active research issues in clustering analysis |32,33|. Here an intutive way of testing the correctness of n_c and the validity of the generated clusters will be illustrated. The method will utilize the maximal cover coefficient concept for clustering. In the single-pass algorithm the test for maximal coverage takes place between an ordinary (non-seed) document and all the seed documents. The test procedure proposed here tests maximal coverage among all documents (no discrimination is made among the documents). The document which is maximally covered by any other document is assumed to appear in the same cluster. If there are more than one candidates for the maximally covering document, the one which has the highest seed power is chosen as the maximal cover.

The clusters generated by the above method will provide the following : (1) a comparison of the number of clusters implied by this method and implied by the cover coefficient concept; (2) a check to see whether all the seeds determined by the seed power method appear in different clusters in the above method; (3) a similarity analysis of the clusters generated by the above method and by the single-pass algorithm.

The validity analysis of the generated clusters requires rigorous mathematical analysis |32,33,34|, such as finding a hypothesis testing. This is another possibitily for future research.

(b) Some observations on the C^{-1} and C^TC matrices :

The rows of the C matrix are linearly independent. This means that $k_1c_1 + k_2c_2 + \ldots + k_mc_m = 0$ if $k_i = 0$ $(i = 1,\ldots,m)$; where c_i $(i = 1,\ldots,m)$ indicates the ith row of the C matrix. Because of this property, the inverse of the C matrix does always exist. Each row sum of the C^{-1} matrix is equal to 1 (a formal proof is not provided yet). After careful analysis of the C^{-1} matrix one may arrive at interesting conclusions. It would worth to work on the C^{-1} matrix and to make observations.

The summation of the entries of the C^TC matrix is equal to m (i.e., number of documents). This is trivial to prove. Since, if a matrix A is obtained by the multiplication of B^TB (B is any matrix without negative entries), then the summation of the elements of A is equal to the sum of the square of the row summations. In the C^TC case, the row sum of the C matrix is equal to 1 by definition. Therefore, squared sum of its row sums will be equal to 1. Furthermore, C^TC is a symmetric matrix, since it is the multiplication of a matrix and its transpose (Notice that the summation of the elements of CC^T is not equal to 1, since the summation of its elements will be equal to the sum of the square of the column summations).

An entry of the C^TC matrix, C_m, is defined as follows :

$$c_{m_{ij}} = \sum_{k=1}^{m} c_{ik} * c_{kj} = c_{m_{ji}}$$

For example, for $m = 3$, $c_{m_{11}} = c_{11}^2 + c_{21}^2 + c_{31}^2$, $c_{m_{12}} = c_{11} * c_{12} + c_{21} * c_{12} + c_{31} * c_{32}$. After these observations, it is easy to see that, c_{ij} is the mutual coverage among documents d_i and d_j. If $i = j$, it is the independent coverage of document-i. These properties of C_m may lead to interesting results.

The only drawback of C^{-1} and C_m matrices is their high computational requirements for creation.

(c) Cluster maintenance experiments :

In Chapter 4 an algorithm for cluster maintenance is proposed. Some experiments can be employed to test the performance of the algorithm. Such as clustering all the documents (m) of the collection together, then clustering the documents by parts, i.e., first clustering the m_1 documents, then clustering the m_2 documents by the cluster maintenance algorithm (where $m = m_1 + m_2$). Then a similarity analysis may be employed between these two clustering patterns and information retrieval performance of the maintenance algorithm in terms of recall and precision can be studied.

(d) Stemming in D matrix generation :

In the creation of the D matrix a stemming algorithm can be incorporated for better language normalization |17, 84-pp.88-89|.

(e) Physical database organization considerations for the clustered documents :

The use of the integrated database management and information retrieval system for very large document collections requires an efficient way of distributing the cluster documents on the backup storage. The methodology to be introduced should provide the full utilization of the data channels among the RAP memory and the secondary storage that will carry the document contents, i.e., it should provide a uniform distribution of the load among the data channels.

(f) The concepts/methodologies which are introduced in Chapter 8 can be tested experimentally.

(g) All of the concepts and the algorithms of the thesis can be tested by the well known document collections of the IR literature.

REFERENCES

1. Akman, V. "Design and Implementation of the Front-end and Controller Hardware/Software Systems for the RAP Database Machine." M.Sc. Thesis, Dept. of Comp. Eng., Middle East Tech. Univ., August 1980.

2. Anderberg, M.R. Clustering Analysis for Application. New York : Academic Press; 1973.

3. Angione, P.V. "On the Equivalence of Boolean and Weighted Searching Based on the Convertibility of Query Forms." Journal of the American Society for Information Science. 26(3-4):112-124;1975.

4. Bates, M.J. "Information Search Tactics." Journal of the American Society for Information Science. 30(7):205-214; 1979.

5. Bertziss, A.T. Data Structures Theory and Practice, 2nd ed. New York : Academic Press; 1975.

6. Bird, R.M., Tu, J.C., Worthy, R.M. "Associative/Parallel Processors for Searching Very Large Textual Data Bases". Workshop on Comp. Arch. for Non-Numeric Proc., pp.8-16, May 1977.

7. Boyer. D.E., Moore, J.S. "A Fast String Searching Algorithm." Communications of the Association for Computing Machinery. 20(10): 762-772; 1977.

8. Bookstein, A. "A Comparison of Two Systems of Weighted Boolean Retrieval." Journal of the American Society for Information Science. 31(7):240-247; 1980.

9. Bookstein, A. "Fuzzy Requests : An Approach to Weighted Boolean Searches." Journal of the American Society for Information Science. 32(7) : 275-279; 1981.

10. Buchanan, B.G., Duda, R. "Principles of Rule-Based Expert System." In: Yovits- M.C. Ed. Advances in Computers. New York: Academic Press; pp.163-210; 1983.

11. Can, F., Özkarahan, E.A. "Text Retrieval with the RAP Database Machine." Comp. Eng. Dept., Middle East Tech. Univ., Tech. Rep. ISDB-10; June 1981.

12. Can, F., Özkarahan, E.A. "A Clustering Scheme." Proceedings of the Sixth Annual SIGIR Conference, Association for Computing Machinery, Bethesda, Maryland, pp.115-121; 1983.

13. Can, F., Özkarahan, E.A. "Two Partitioning Type Clustering Algorithms." Journal of the American Society for Information Science. 35(5) : 268-276; 1984.

14. Can, F., Özkarahan, E.A. "Similarity and Stability Analysis of the Two Partitioning Type Clustering Algorithms." Journal of the American Society for Information Science. 36(1): 3-14; 1985.

15. Can, F. "Bilimsel Araştırmaların Vazgeçilmez Aracı, Bilgi Erişim Sistemleri." Bilgisayar. 36: 42-44, 45-47; 1984.

16. Can, F., Özkarahan, E.A. "Belge Erişim Sorununa Bir Yaklaşım." Bilişim'84 Bildiriler Kitabı. s.202-214, İstanbul; 1984.

17. Chen, P.P. "The Entity-Relationship Model--Toward a Unified View of Data." ACM Transactions on Database Systems. 1(1):9-36; 1976.

18. Codd, E.F. "A Relational Model of Data for Large Shared Data Banks." Communications of the Association for Computing Machinery. 13(16): 377; 1970.

19. Copeland, G.P. "Editing Requirements for Database Applications and Their Implementation on the Indy Backend Kernel." Fourth Workshop on Comp. Arch. for Non-Numeric Proc., Syracuse, N.Y., pp.8-17; Aug. 1978.

20. Cormack, R.M. "A Review of Classification." Journal of Royal Statistics Society. (A134):321-367; 1971.

21. Corneil, D.G., Woodward, M.E. "A Comparison and Evaluation of Graph Theoretical Clustering Techniques." INFOR. 16:74-89; 1978.

22. Crawford, R.G. "The Computation of Discrimination Values." Information Processing and Management. 11:249-253; 1975.

23. Crawford, R.G., Coughlin, L.A., Mayes, L.W. "MISTRAL/11 Users' Manual." Technical Report 78-69, Queen's University, Kingston, Ontario, Canada; Feb. 1979.

24. Crawford, R.G. "The Relational Model in Information Retrieval." Journal of the American Society for Information Science. 32(1): 51-64; 1981.

25. Croft, W.B. "Clustering of Large Files of Documents Using the Single-Link Method." Journal of the American Society for Information Science. 28:341-344; 1977.

26. Croft, W.B. "A File Organization for Cluster-Based Retrieval." Int. Conf. on Info. Stor. and Ret, SIGIR. 13(1); 1978.

27. Crouch, D.B. "A File Organization and Maintenance Procedure for Dynamic Document Collections." Information Processing and Management. 11:11-21; 1975.

28. Date, C. An Introduction to Database Management Systems, 2nd ed. Reading Massachusetts: Addison Wesley; 1977.

29. Day, W.H.E. "The Complexity of Computing Metric Distances Between Partitions." Technical Report No.7901, Memorial Univ. of Newfoundland, st. John's, Newfoundland, Canada.

30. Doğaç, A. "Design and Implementation of a Generalized Database Management System-METUGDBMS." Ph.D. Thesis, Dept. of Comp. Eng., Middle East Tech. Univ., Sept. 1980.

31. Doszkocs, T.E., Rapp, B.A., Schoolman, H.M. "Automated Information Retrieval in Science and Technology." Science. 208:25-30; 1980.

32. Dubes, R., Jain, A.K. "Clustering Methodologies in Explanatory Data Analysis." In: Yovits, M.C. Ed. Advances in Computers. New York: Academic Press; pp.113-228; 1980.

33. Dubes, R., Jain, A.K. "Validity Studies in Clustering Methodologies." Pattern Recognition. 11:235-254; 1983.

34. Duda, R.O., Hart, P.E. Pattern Classification and Scene Analysis. Wiley, New York; 1973.

35. Everitt, B.S. "Unresolved Problems in Cluster Analysis." Biometrics. 35:169-181; 1979.

36. Everitt, B.S. Cluster Analysis. New York: Halsted Press Div. of John Wiley and Sons; 1980.

37. Goodman, L.A., Kruskal, W.H. "Measures of Association for Cross Classifications." Journal of the American Statistical Association. 49:732-764; 1954.

38. Gotlieb, C.C., Kumar, S. "Semantic Clustering of Index Terms." Journal of the Association for Computing Machinery. 15(4):493-513; 1968.

39. Hall, J.L., Marjorie, J.B. Online Bibliographic Databases An International Directory, 2nd ed. Old Woking Surrey, The Gresham Press (Published by Aslib), 1981.

40. Harding, A.F., Willet, P. Indexing Exhaustivity and the Computation of Similarity Matrices. Journal of the American Society for Information Science. 30:298-300; 1979.

41. Harris, J.L. "Terminology Change: Effect on Index Vocabularies." Information Processing and Management. 15:77-88; 1979.

42. Hartigan, J.A. Clustering Algorithms. New York: John Wiley and Sons; 1975.

43. Haskin, R.L., Hollaar, L.A. "Operational Characteristics of a Hardware-based Pattern Matcher." Association for Computing Machinery Transactions on Database Systems. 8(1):15-40; 1983.

44. Heaps, H.S. Information Retrieval Computational and Theoretical Aspects. New York: Academic Press; 1978.

45. Hollaar, L.A. "Specialized Merge Processor Networks for Combining Sorted Lists." Association for Computing Machinery Transactions on Database Systems. 3(3):272-284; 1978.

46. Hollaar, L.A. "Text Retrieval Computers." Computer. 12(3):40-50; 1979.

47. Hollaar, L.A. "Hardware Systems for Text Information Retrieval." Proceedings of the Sixth Annual SIGIR Conference, Association for Computing Machinery. Bethesda, Maryland, pp.3-9; 1983.

48. Horspool, R.N. "Practical Fast Searching in Strings." Software Practice and Experience. 10(6):501-506; 1980.

49. Hsiao, D.K. "Database Computers." In: Yovits, M.C. Ed. Advances in Computers. New York: Academic Press; 1980:pp.1-64.

50. Jardine, N., Sibson, R. Mathematical Taxonomy. London and New York, John Wiley; 1971.

51. Jardine, N., Van Rijsbergen, C.J. "The Use of Hierarchical Clustering in Information Retrieval." Information Storage and Retrieval. 7:217-240; 1971.

52. Knuth, D.E., Morris, J.H., Pratt, V.R. "Fast Pattern Matching in Strings." SIAM Journal of Computing. 6(2):323-350; 1977.

53. Köksal, A. "Bilgi Erişim Sorunu ve Bir Belge Dizinleme ve Erişim Dizgesi Tasarım ve Gerçekleştirimi." Doçentlik Tezi. Ankara; 1979.

54. Köksal, A. "Tümüyle Özdevimli Deneysel Bir Belge Dizinleme ve Erişim Dizgesi: TÜRDER. Bilişim'80 Bildiriler Kitabı. pp.37-44, Ankara; 1980.

55. Lancaster, F.W. Information Retrieval Systems: Characteristics, Testing and Evaluation, 2nd ed. New York: John Wiley and Sons; 1979.

56. Lefkovitz, D. File Structures for on Line Systems. New York: Hayden Book Co.; 1969.

57. Macleod, I.A. "Towards an Information Retrieval Language Based on a Relational View of Data." Information Processing and Management. 13:167-175; 1977.

58. Maron, M.E. "Depth of Indexing. "Journal of the American Society for Information Science. 29:224-228; 1978.

59. McCarn, D.B. "MEDLINE: An Introduction to On-line Searching." Journal of the American Society for Information Science. 31(2):181-192; 1980.

60. Meadow, C.T., Cochrane, P. (A) Basics of Online Searching. New York: John Wiley and Sons; 1981.

61. Mukhopadhyay, A. "Hardware Algorithms for Nonnumeric Computation." IEEE Transactions on Computers. C-28(6):384-394; 1979.

62. Oflazer, K. "A Microprocessor Based Approach to RAP Database Machine Cell Structure: Design and Analysis." M.Sc. Thesis, Dept. of Comp. Eng., Middle East Tech. Univ., June 1979.

63. Oflazer, K., Özkarahan, E.A., Smith, K.C. "RAP.3-A Multi-microprocessor cell Architecture for the RAP Database Machine." Proc. of Int. Workshop on High Level Language Computer Architecture. pp.108-119; May 1980.

64. Oflazer, K. Ph.D. Thesis Draft Proposal, Carnegie Mellon University; 1983.

65. Özkarahan, E.A., Schuster, S.A., Smith, K.C. "RAP-An Associative Processor for Database Management." Proc. of AFIPS NCC. 44:379-387; 1975.

66. Özkarahan, E.A. "An Associative Processor for Relational Data Bases-RAP." Ph.D. Thesis, Department of Computer Science, University of Toronto, Jan.1976.

67. Özkarahan, E.A., Schuster, S.A., Sevcik, K.C. "Performance Evaluation of a Relational Associative Processor." Association for Computing Machinery Transactions on Database Systems. 2(2): 175-195; 1977.

68. Özkarahan, E.A., Sevcik, K.C. "Analysis of Architectural Features for Enhancing the Performance of a Data Base Machine." Association for Computing Machinery Transactions on Database Systems. 2(4): 297-316; 1977.

69. Özkarahan, E.A., Oflazer, K. "Microprocessor Based Modular Database Processors." Proceedings of the 4th Int. Conference on Very Large Databases. Berlin, Sep. 1978, pp.300-311.

70. Özkarahan, E.A. "RAP Veri Tabanı Bilgisayarının Genel Amaçla Kullanımının Sağlanması." Doçentlik Tezi, Orta Doğu Teknik Üniversitesi, 1980.

71. Özkarahan, E.A., Can, F. "Integration of Fact/Document Retrieval Systems Within a Database Machine." Dept. of Computer Science, Technical Report TR-82-02, Arizona State University, Nov.1982.

72. Özkarahan, E.A., Tansel, A.U., Smith, K.C. "Database Machine/Computer Based Distributed Databases." Proc. of the 2nd International Symposium on Distributed Databases. Berlin, 1982. Published by North Holland Pub. Co.

73. Özkarahan, E.A. "Implementations of the Relational Associative Processor (RAP) and Its System Configurations." Arizona State University, Tech. Rep. 82-05, 1982.

74. Özkarahan, E.A. "Desirable Functionalities of Database Architectures." Proc. of IFIP World Congress. 1983.

75. Özkarahan, E.A., Can. F. "An Integrated Fact/Document Information System for Office Automation." Information Technology: Research and Development. 3(3); 1984.

76. Peterson, J.L. "Computer Programs for Detecting and Correcting Spelling Errors." Communications of the Association for Computing Machinery. 23(12):676-687; 1980.

77. Porter, M.F. "An Algorithm for Suffix Stripping." Program. 14: 130-137; 1980.

78. Raghavan, V.V., Yu, C.T. "A Comparison of the Stability Characteristics of Some Graph Theoretical Clustering Methods." IEEE, Transactions on Pattern Analysis and Machine Intelligence. PAMI-3(4): 393-402; 1981.

79. Raghavan, V.V. "Approaches for Measuring the Stability of Clustering Methods." Association for Computing Machinery SIGIR Forum. 17(1):6-20; 1982.

80. Raghavan, V.V., Ip, M.Y.L. "Techniques for Measuring the Stability of Clustering: A Comparative Study." Association for Computing Machinery SIGIR Conference, W. Berlin; 1982.

81. Radecki, T. "A Model of a Document-Clustering-Based Information Retrieval System with a Boolean Search Request Formulation." Joint BCS & ACM Symposium. pp.334-344; June 24-26 1980.

82. Rand, V.M. "Objective Criteria for the Evaluation of Clustering Methods." Journal of the American Statistical Association. 66:846-850; 1971.

83. Roberts, D.C. "A Specialized Computer Architecture for Text Retrieval." Fourth Workshop on Comp. Arch. for Non-Numeric Proc., Syracuse, N.Y., Aug. 1978, pp.51-59.

84. Salton, G. Automatic Information Organization and Retrieval. New York: McGraw Hill; 1968.

85. Salton, G. (editor) The Smart Retrieval System Experiments in Automatic Document Processing. Englewood Cliffs, New Jersey: Prentice-Hall; 1971.

86. Salton, G. Dynamic Information and Library Processing. Englewood Cliffs, New Jersey: Prentice Hall; 1975.

87. Salton, G., Wong, A., Yang, C.S. "A Vector Space Model for Automatic Indexing." Communications of the Association for Computing Machinery. 18(11):613-620; 1975.

88. Salton, G. "A Theory of Indexing." Regional Conference Series in Applied Mathematics No.18, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1975.

89. Salton, G., Wong, A. "Generation and Search of Clustered Files." Association for Computing Machinery Transactions on Database Systems. 3(4):321-346; 1978.

90. Salton, G. "Automatic Information Retrieval." Computer. 13(9): 41-56; 1980.

91. Salton, G., Wu, H., Yu, C.T. "The Measurement of Term Importance in Automatic Indexing." Journal of the American Society for Information Science. 32(3):175-186; 1981.

92. Salton, G., McGill, M.J. Introduction to Modern Information Retrieval. New York: McGraw Hill; 1983.

93. Salton, G., Fox, E.A., Wu, H. "Extended Boolean Information Retrieval." Communications of the Association for Computing Machinery. 26(12):1022-1036; 1983.

94. Scheck, H.J. "Methods for the Administration of Textual Data in Database Systems." Proc. Joint BCS and ACM Symp., pp.218-235; 1980.

95. Schuster, S.A., Özkarahan, E.A., Smith, K.C. "A Virtual Memory System for a Relational Associative Processor." Proc. of AFIPS. 45:291-296; 1976.

96. Schuster, S.A., Nguyen, H.B., Özkarahan, E.A., Smith, K.C. "RAP.2-An Associative Processor for Databases and Its Applications." IEEE Transactions on Computers. C-28(6):446-458; 1979.

97. Smith, L.C. "Artificial Intelligence in Information Retrieval Systems." Information Processing and Management. 12:189-222; 1976.

98. Stellhorn, W.H. "A Processor for Direct Scanning of Text." Workshop on Comp. Arch. for Non-Numeric Proc. Oct., 1974.

99. Stonebraker, M. et.al. "Document Processing in a Relational Database System." Association for Computing Machinery Transactions of Office Information Systems. 1(2):143-158; 1983

100. Tague, J.M., Nelson, M.J. "Simulation of User Judgements in Bibliographic Retrieval Systems." Proceedings of the Fourth International Conference on Information Storage and Retrieval, ACM-SIGIR. Okland, California, pp.66-71, 1981.

101. Tansel, U.A. "Design and Evaluation of a Query Processing System for Distributed Database Machine Networks." Ph.D. Thesis, Dept. of Comp. Eng., Middle East Tech. Univ., Sept. 1981.

102. Ünlü, S. "Design and Implementation of a Software Emulator for the Relational Associative Processor-RAP." M.Sc. Thesis, Dept. of Comp. Eng., Middle East Tech. Univ., August 1979.

103. Van Rijsbergen, C.J., Sparck Jones, K. "A Test for the Separation of Relevant and Nonrelevant Documents in Experimental Retrieval Collections." Journal of Documentation. 29:251-257; 1973.

104. Van Rijsbergen, C.J. "Further Experiments with Hierarchical Clustering in Document Retrieval." Information Storage and Retrieval, 10:1-14; 1974.

105. Van Rijsbergen, C.J. The Best-Match Problem in Document Retrieval." Communications of the Association for Computing Machinery. 17(11): 648-649; 1974.

106. Van Rijsbergen, C.J. Information Retrieval, 2nd ed. London: Butterworths; 1979.

107. Willet, P. "Document Clustering Using an Inverted File Approach." Journal of Information Science. 2:223-231; 1980.

108. Ziman, J.M. "The Proliferation of Scientific Literature: A Natural Process." Science. 208:369-371; 1980.

APPENDICES

APPENDIX - A

TYPICAL FREE TEXT RETRIEVAL OPERATIONS

A KEYWORD SEARCH

Finds any document that contains the word A

A OR B EITHER WORDS

Finds any document that contains either the word A or the word B.

A AND B BOTH WORDS

Finds any document that contains both the word A and the word B.

A AND NOT B ONE BUT NOT THE OTHER

Finds any document that contains the word A but not the word B.

(A,B) IN SENT SPECIFIED CONTEXT

Finds any document that contains both the word A and word B in the same sentence.

A B FINDS A WORD PHRASE

Finds any document that contains the word A immediately followed by the word B.

A...B ONE FOLLOWED THE OTHER

Finds any document that contains the word A followed (either immediately or after an arbitrary number of words) by the word B.

<A.n.B> DIRECTED PROXIMITY

Finds any document that contains the word A followed by the word B within n words.

<A,B>n UNDIRECTED PROXIMITY

Finds any document that contains the words A and B within n words of each other.

(A,B,C,D) # n THRESHOLD OR

Finds any document that contains at least n of the different words A,B,C,D. Note that if $n = 1$ this is an "OR" operation; hence the retrieved document contains one or more of the strings A,B,C, or D. If n equals the number of different words specified, then this is an "AND" operation; the retrieved document must contain all the specified words in any order.

A???B FIXED_LENGTH DON'T CARE

Matches the character string A, followed by three arbitrary characters, then followed by the string B.

A*B VARIABLE_LENGTH DON'T CARE

Matches the character string A, followed by zero or more characters, then followed by the character string B.

APPENDIX - B

STOP WORD LIST USED IN THE EXPERIMENTS OF CHAPTER 4

A ABOUT AFTER AGAINST ALL ALLOWS
ALMOST ALONG ALTHOUGH AMONG AN AND
ANOTHER ANY APPROACH ARE AROUND AS
ASPECTS AT BE BECAUSE BEEN BEING
BETWEEN BOTH BUT BY CALLED CAN
CANNOT CASE CASES CHANGES CONSIDER COULD
DEFINED DOES DONE EACH EITHER ELSEWHERE
END ETC EVERY FAULTS FOR FROM
GIVEN HAS HAVE HERE HOC HOW
HOWEVER IMPORTANT IN INTO INTRODUC IS
IT ITS ITSELF MADE MANY MAY
MORE MOST MUCH MUST NEW NOT
OF ON ONE ONLY OR OTHER
OUR OVER PAPER PER POSSIBLE PROBLEM
REQUIRED RULES SOME SECOND SEVERAL SHOULD
SHOW SHOWN SIMILAR SINCE SOME STUDY
SUCH THAN THAT THE THEIR THEN
THERE THEREFORE THESE THEY THOSE THROUGH
THUS TIME TIMES TO UNDER UPON
USE USING VARIOUS VERY WAS WAY
WELL WERE WHAT WHEN WHERE WHETHER
WHICH WHILE WHOSE WILL WITH WITHIN
WITHOUT WOULD

APPENDIX - C

TIMING ANALYSIS OF THE RAP.3 VERSION-I CELL OPERATIONS

The following analysis describes the relationships among certain timing parameters |7,8|

Let

T_{BIT} ≡ CM bit time = 1/CM data rate.

TUPLEN ≡ length of a tuple in bits.

k ≡ number of subcells/cell (k > =3 because of the data move strategy incorporated).

T_{LS} ≡ time to load (store) via DMA = $T_{BIT} * \text{TUPLEN}$.

T_{AVLNW} ≡ available time to process a tuple if no wait is imposed (this time is naturally available in the architecture). $T_{AVLNW} = (k-2) * T_{LS}$.

NT ≡ number of tuples in the circulating memory (CM).

T_{TOTAL} ≡ total circulation of CM from the start of loading of the first tuple to the end of storing the last tuple.

T_{REST} ≡ extra time needed to store the last (k - 1) tuples (It should be noted that CM circulation is completed only after the last tuple is restored. Some extra time is needed to restore the last (k - 1) tuples because the total dynamic capacity of the cell memory is equal to the CM capacity plus the capacities of the (k - 1) subcell buffers).

$T_{REST} = (k - 1) * T_{LS}$

Assume that all tuples require exactly L times the time allowed by the architecture (the value of L is dependent on the complexity of the RAP instruction), that is :

$$T_{REQ} = L * T_{AVLNW} ; \quad (L \geq 1)$$

Assuming also that mod(NT,k) = 0, then during the circulation, NT/k tuples will be processed by a subcell.The time to handle a tuple is :

$$T_{TUPLE} = T_{REQ} + 2 * T_{LS}$$

where the last term accounts for the load and store times.

Since the processing of the tuples are overlapped over the k subcells, the total time for a circulation will be :

$$T_{TOTAL} = (NT / k) * T_{TUPLE} + T_{REST}$$
$$+ (L - 1) * (k - 2) * T_{LS}$$

where the first term is the time to process NT tuples with k subcells in parallel, the third term is the initial extra time (beyond the allocated time) required by $subcell_1$ for $tuple_1$.

APPENDIX - D

A BRIEF SUMMARY OF RAP INSTRUCTIONS

Mapping and retrieval instructions

(over all occurrences)

MARK : Selects and marks (tags) record occurrences.

RESET : Selectively removes tags.

READ : Selects and reads qualified tuple (or attribute value) occurrences.

READ-MARKS : The same as read but output includes also the mark bits.

CROSS-MARK : Performs a RAP semi-join between two record types.

CROSS-RESET : Combines two semi-joins on a target record type.

GET-FIRST : Sets and moves cursor within a record type also saves attribute values from the tuple occurrence indicated by the cursor.

SAVE : Saves attribute value from a single record occurrence.

Update instructions

(Select and in-place update over all occurrences)

ADD,SUB,MUL,DIV(AOP) : Select and Attribute value - 1 ← Attribute Value - 1 AOP Attribute Value - 2 (or constant)

REPLACE : Select and Attribute Value - 1 ← Attribute Value - 2 (or constant)

Statistical (set or aggregate) function

(Select and compute functions in-place over all occurrences)

SUM,COUNT,MAX,MIN,AVERAGE (SOP) : Select and compute the SOP function over the selected occurrences.

Definition and storage manipulation instructions

RELATION : Defines a new relation (record type). Size, type, length parameters for the data are declared. (Key atributes and access paths are defined it the software emulator rather than the actual machine is used). User capabilities, access rights, and the protection parameters are also declared with the use of this command.

CREATE : Populates the database for the specified record types which have been defined by the RELATION command.

DESTROY : Delets a record type.

DELETE : Selects and deletes record occurrences from the record type.

INSERT : Inserts a single record occurrence into the record type.

System instructions

AUTHORIZE : Grants access by user authentication.

LOCK : Specified record types are locked against concurrent accesses.

RELEASE : Releases locks.

Multiprocessing/multiprogramming and DDB related instructions

LOCATE : Returns the site address of the record type to the requesting site.

STATUS : Provides status information on a record type for the requesting site for branch control in iterated RAP programs.

SAVE_MARKS : Current mark bit values in each record occurrence are saved in a specified data attribute within the record occurrence.

RESTORE_MARKS : Restores previously saved mark bits in each record occurrence into the active mark bit positions.

Register manipulation instructions

(Operate on the specified RAP registers)

READ_REG : Reads out register contents.

RSET : Enters immediate data into the operand registers.

DEC_REG : Decrements register contents.

INC_REG : Increments register contents

RADD,RSUB,RMUL,RDIV (ROP) : Performs specified arithmetic operations on the register as : <reg> ← <reg> <ROP> <operand>

Control instructions

BC : Branch conditional or unconditional (condition includes test of marked record occurrence within a record type).

EOQ : Indicates end of a query program.

Document retrieval instructions

MATCH, MATCH-WS, MATCH-WWC : Search for specified pattern combinations with respect to instruction context within the record type containing the document(s).

LINK-PASS : Perform text match overflow resolution between the record occurrences.

APPENDIX - E

TEXT RETRIEVAL MACROS FOR THE RAP SYSTEM

As an aid to the user of RAP Text Retrieval System, a macro processes and a number of macros have been implemented. In this appendix the properties of the macro processor and the text retrieval macros will be described.

PROPERTIES OF THE MACRO PROCESSOR

A macro processor for RAP, called RAPMAC, has been implemented on the INTEL MDS225 development system under the operating system ISIS-II. FORTRAN-80 has been utilized for the implementation in order to provide portability for RAPMAC. After the RAPMAC macros are expanded into RAP code, the development system passes this code to the RAP emulator: SERAP, running on the INTEL single board computer iSBC 86/12A, for execution.

A RAPMAC macro definition contains the following :

a. a header statement,

b. a prototype,

c. model statements, and

d. a trailer statement.

The header and trailer statements are very simple. The header contains the keyword MACRO between the columns 2-72 and the trailer contains the keyword MEND (for Macro END) again between columns 2-72. After these keywords, a commend can be entered on the same card.

The prototype follows the header statement. This statement first contains the name of the macro. The name starts with an alphabetic character and should not exceed 16 characters. The user defined macros cannot have the same name with the library macros. The prototype statement also includes the formal parameters of the macro (if any). The parameters are separated with commas and obey the rules of macro names, however, the first symbol of a parameter can be '&'. A non-blank character on column 72 indicates the continuation of the prototype card.

Model statements are any strings which may include the formal parameters. The occurrence of a formal argument is recognized between non-alphabetic and non-numeric characters. An actual argument concatenated with a alphanumeric string can appear with formal arguments that start with '&'.

A maximum of 40 macros can be defined with each macro holding up to 30 parameters.

A valid macro definition can be as follows :

```
MACRO                     Header
REGSET REGNAM,PARAM       Prototype
RSET[REGNAM, PARAM]       MODEL Statements
MEND                      Trailer
```

Macro Calls

Macro definitions end with a *JOB card. After this card, a sequence of cards which may include macro calls will follow. To increase the scanning speed all macro call cards should start with the symbol '%' on their first column. A literal constant as an actual parameter starts and ends with a quotation mark end if it contains a quotation mark quotes must be doubled. The maximum length of a literal constants is 30. A non-blank character on column 72 indicates the continuation of the macro call card.

Some typical macro call statements are shown below:

```
% MOVE        'LITERAL STRING', SYMBOLA
% REGSET       REGU_1,''' TEXT '''
% REGSET       REGU_1,'10'
```

The macro names and formal and non-literal actual parameters may include the symbol '$' in their body. This symbol is used to improve readability and ignored by the macro processor.

During macro processing the first column and columns 73 through 80 of all the input cards are ignored.

An actual parameter can be concatenated with a string by making the first symbol of the former a '&' as shown below :

```
RSET     REGU1, '&A.BASE'
```

will cause the generation of the following statement :

```
RSET    [REGU1, 'DATABASE']
```

if the corresponding actual parameter for A is DATA.

A complete example, for macro definition and macro call, is shown in the fololwing :

```
     MACRO
     CMARK R1, A1, R2, A2, R3, A3
     CROSS_MARK (T2) [R1:A1 = R2.A2] [MKED (T1)]
     CROSS_MARK (T2) [R3:A3 = R1.A1] [MKED (T2)]
     MEND
*JOB
     MARK (T1)   [REL2:ATT1 <=10]
% CMARK REL1, ATT1, REL2, ATT1, REL3, ATT1
+CROSS_MARK (T2) [REL1:ATT1 = REL2.ATT1] [MKED (T1)]
+CROSS_MARK (T2) [REL3:ATT1 = REL1.ATT1] [MKED (T2)]
     other SERAP Statements
     EOQ
```

In the above illustration, the RAP commands generated due to the model statements of the macro definition are preceeded by '+'.

The output file generated by RAPMAC as input to the SERAP will look like the following :

```
*JOB
     MARK (T1)   [REL2 : ATT1 <=10]
     CROSS_MARK (T2)  [REL1:ATT1 = REL2.ATT1]  [MKED (T1)]
     CROSS_MARK (T2)  [REL3:ATT3 = REL1.ATT1]  [MKED (T2)]
          other SERAP statements
     EOQ
```

During macro call, unspecified actual parameters will be replaced by the null symbol. For example, in the following calling statements:

```
% CMARK REL1,,REL2,ATT1,REL3

% CMARK REL1,,REL2,ATTI,REL3,
```

second and the last, i.e., sixth, actual parameters are not porvided.

The macro bodies defined may contain labels that are necessary for the transfer of control during execution of the generated RAP assembler instructions. These labels are replaced by distinct strings as follows. If the symbol "@" is followed by an integer "n", the symbol and the following integer is replaced with the number equal to SYSNDX+n, where SYSNDX is a system variable defined in RAPMAC. This system variable is incremented by the maximum "n" (@n), which appears in the macro body, after each macro expansion. For example, if we have the following statements, within a macro body :

```
      BC LOOP @1
LOOP @1
      BC LOOP @2
LOOP @2
```

If the macro containing the above lines is called twice, the following sequence of RAP instructions will be generated:

```
+     BC LOOP1
+LOOP1
+     BC LOOP2
+LOOP2
      second call follows
+     BC LOOP3
+LOOP3
+     LOOP4
+LOOP4
```

Macro calls within macros are not allowed in RAPMAC.

During macro processing, the macro bodies are saved in a macro definition file and this file is used during macro expansion. After macro processing this file is saved and can be specified as a macro library in the subsequent uses of RAPMAC.

LISTING OF THE RAP TEXT RETRIEVAL MACROS

Text retrieval applications in the RAP system are programmed with the use of RAP instruction set and nine available text retrieval macros which are listed in the following :

1. MATCH$A : Finds any document that contains the word A.
2. MATCH$ALL$A : Finds any document that contains the word A. Different than MATCH$A it searches for all occurrences of A.
3. MATCH$A$OR$B : Finds any document that contains either the word A or the word B.
4. MATCH$A$AND$B : Finds any document that contains the word A and the word B.
5. MATCH$A$AND$NOT$B : Finds any document that contains the word A but not the word B.
6. MATCH$A$AND$B$IN$SENT : Finds any document that contains both the word A and the word B within the same sentence.
7. MATCH$A$ANYWORDS$B : Finds any document that contains the word A (either immediately or after an arbitrary number of words) followed by the word B.

8. MATCH$A$NWORDS$B : Finds any document that contains the word A followed by the word B within n words.

9. MATCH$THRESHOLD$OR : Finds any document that contains at least n of the m distinct words : A_1, A_2,..., A_m; where n <= m.

In the following, the meanings of the macro parameters are explained together with their actual use in the macro call statements.

$STRING_i$ (i <= 2) : the strings to be matched (i.e., A and B) in the text.

REL : the relation name containing the literal domain to be searched.

LA : the literal attribute of relation REL which corresponds to the literal domain that holds the text to be searched.

NA_i (i <= 2) : the numeric attributes of relation REL which are needed during the execution of the macro body.

T_i (i <= 5) : the distinct mark bits of relation REL needed for the execution of the macro body. Before calling a text retrieval macro, all of the mark bits specified in the macro call statement, should be cleared. After execution of the macro body, the first mark bit indicates (shown as T_1 in the call statements) the qualified tuples for the text retrieval operation on the unary relation DOCUMENT (whose only attribute is DOCID). The first mark bit given in the macro call remains clear in relation REL.

QUAL : the tuples for text retrieval are selected with respect to the qualifier which is a literal constant and contains a RAP qualification expression. If all the tuples of relation REL are to be considered, this parameter is omitted. A typical QUAL specification would be ':MKED(T1)'.

The macro call statements for the text retrieval operations in RAPMAC are demonstrated in the following (the character '%' is omitted in the following illustration) :

1. Macro MATCH$A :

MATCH$A STRING, REL,LA,$T_1$,QUAL

2. Macro MATCH$ALL$A :

MATCH$ALL$A $STRING_1$,N,REL,LA,NA_1,NA_2,NA_3,T_1,T_2,T_3, QUAL

After execution of the macro body, the documents which have at least N number of occurrences of $STRING_1$ are marked T_1; T_2 and T_3 remain clear.

3. Macro MATCH$A$OR$B :

MATCH$A$OR$B $STRING_1$,$STRING_2$,REL,LA,T_1, T_2, QUAL

4. Macro MATCH$A$AND$B :

MATCH$A$AND$B $STRING_1$,$STRING_2$,REL,LA,T_1,T_2,QUAL

After execution of the macro body, T_2 remains clear.

5. Macro MATCH$A$NOT$B :

MATCH$A$NOT$B $STRING_1$,$STRING_2$,REL,LA,T_1,T_2,QUAL

After execution of the macro body, T_2 remains clear.

6. Macro MATCH$A$AND$B$IN$SENT :

MATCH$A$AND$B$IN$SENT $STRING_1$,$STRING_2$,REL,LA,NA_1,T_1,T_2,T_3,T_4,QUAL

After execution of the macro body, the mark bits T_2 through T_3 remain clear.

7. Macro MATCH$A$ANYWORD$B :

MATCH$A$ANYWORD$B $STRING_1$,$STRING_2$,REL,LA,NA_1,T_1,T_2, QUAL

After execution of the macro body, T_2 becomes garbage.

8. Macro MATCH$A$NWORDS$B :

MATCH$A$NWORDS$B $STRING_1$,$STRING_2$,N,REL,LA,NA_1,T_1,T_2,T_3,T_4,T_5,QUAL

After execution of the macro body, T_2, T_3 and T_4 remain clear, T_5 becomes garbage.

9. Macro MATCH$THRESHOLD$OR :

MARCH$THRESHOLD$OR M,N,REL,LA,NA_1,NA_2,T_1,QUAL

In the call statement of this macro, M is a literal constant indicating the total number of the search operands. Similarly, N is a literal constant that indicates the minimum number of matches required out of M search operands.

MACRO DEFINITIONS FOR THE TYPICAL TEXT RETRIEVAL OPERATIONS

Macro definition of the keyword search

```
   MACRO
   MATCH$A A, TEXT, D1, T1, QUAL
   /* ... ** Search for A.  */
   RSET [REGU_1, 'A']
   MATCH(T1) [TEXT(D1) QUAL]
   BNC LL@1, TEXT.RAIL_STAT(T1)
     CROSS_MARK(T2) [DOCUMENT:DOCID = TEXT.DOCID] [MKED(T1)]
 LL@1
   MEND
```

Macro definition of the N number of occurrences of a term

```
  MACRO
  MATCH$ALL$A A, N, TEXT, D1, D2, D3, T3, T1, T2, QUAL
  /* ... ** Search for all 'A's, the documents which has at least N number of
            occurrences will be selected.  */
  RSET [REGU_2, N]
  RSET [REGU_1, 'A']
  REPLACE [TEXT(D2) QUAL] [0]
  MATCH(T1) [TEXT(D1, D2, D3) QUAL]
  BNC LL@3, TEXT.RAIL_STAT(T1)
 LL@1
  GET_FIRST [TEXT(DOCID): MKED(T1)]
    MARK(T2) [TEXT: DOCID = REGC_1]
    SUM [TEXT: MKED(T2)] [REGF_1]
    BC LL@2, REGF_1 < N
      MARK(T3) [TEXT: DOCID = REGC_1]
LL@2
    RESET(T1 T2) [TEXT: DOCID = REGC_1]
 BC LL@1, TEXT.RAIL_STAT(T1)
 CROSS_MARK(T3) [DOCUMENT:DOCID = TEXT.DOCID] [MKED(T3)]
LL@3
 MEND
```

Macro definition of the A OR B

```
 MACRO
 MATCH$A$OR$B A, B, TEXT, D1, T2, T1, QUAL
 /* ... ** Match A or B: */
 RSET [REGU_1, 'A']
 MATCH(T1) [TEXT(D1) QUAL]
 RSET [REGU_1, 'B']
 MATCH(T1) [TEXT(D1) QUAL]
 BNC LL@1, TEXT.RAIL_STAT(T1)
   CROSS_MARK(T2) [DOCUMENT:DOCID = TEXT.DOCID] [MKED(T1)]
LL@1
 MEND
```

Macro definition of the A AND B

```
 MACRO
 MATCH$A$AND$B A, B, TEXT, D1, T1, T2, QUAL
 /* ... ** Match A and B:  */
 RSET [REGU_1, 'A']
 MATCH(T1) [TEXT(D1) QUAL]
 BNC LL@1, TEXT.RAIL_STAT(T1)
   RSET [REGU_1, 'B']
   MATCH(T2) [TEXT(D1) QUAL]
   BNC LL@1, TEXT.RAIL_STAT(T2)
   CROSS_MARK(T1) [DOCUMENT:DOCID = TEXT.DOCID] [MKED(T1)]
   CROSS_MARK(T2) [DOCUMENT:DOCID = TEXT.DOCID] [MKED(T2)]
LL@1
 MEND
```

Macro definition of the A AND NOT B

```
 MACRO
 MATCH$A$AND$NOT$B A, B, TEXT, D1, T1, T2, QUAL
 /* ... ** Match A NOT B:  */
 RSET [REGU_1, 'A']
 MATCH(T1) [TEXT(D1) QUAL]
 BNC LL@1, TEXT.RAIL_STAT(T1)
   CROSS_MARK(T1) [DOCUMENT:DOCID = TEXT.DOCID] [MKED(T1)]
   CROSS_MARK(T1) [TEXT: DOCID = DOCUMENT.DOCID] [MKED(T2)]
   RSET [REGU_1, 'B']
   MATCH(T2) [TEXT(D1): MKED(T1)]
   BNC LL@1, TEXT.RAIL_STAT(T2)
     CROSS_MARK(T2) [DOCUMENT:DOCID = TEXT.DOCID] [MKED(T2)]
     RESET(T2) [TEXT: MKED(T2)]
LL@1
 MEND
```

Macro definition of the (A,B) IN SENT

```
 MACRO
 MATCH$A$AND$B$IN$SENT A, B, TEXT, D1, D2, T3, T1, T2, T13, QUAL
 /* ... ** Match within sentence A & B:  */
 MARK(T13) [TEXT QUAL]
 REPLACE [TEXT(D2) QUAL] [0]
LL@1
 BNC LL@3, TEXT.RAIL_STAT(T13)
 RSET [REGU_1, 'A']
 BNC LL@2, RAIL_STAT(T1) RESET(T1) [TEXT: MKED(T1)] LL@2
 MATCH(T1) [TEXT(D1, D2): MKED(T13)]
 BNC LL@3, RAIL_STAT(T1)
 /* ... ** Reset the unnecessary tuples.  */
 RESET(T13) [TEXT: UNMKED(T1)]
 RSET [REGU_1, 'B']
 MATCH_WS(T2) [TEXT(D1, D2):MKED(T1)]
 BNC LL@1, RAIL_STAT(T2)
   CROSS_MARK(T1) [DOCUMENT:DOCID = TEXT.DOCID] [MKED(T2)]
   CROSS_MARK(T3) [TEXT:DOCID = DOCUMENT.DOCID] [MKED(T1)]
   RESET(T13) [TEXT:MKED(T3)]
   BC LL@1
LL@3
 MEND
```

Macro definition of the A...B

```
 MACRO
 MATCH$A$ANYWORDS$B A, B, TEXT, D1, D2, T2, T1, QUAL
 /* ... ** Match A (any words) B:  */
 RSET [REGU_1, 'A']
 REPLACE [TEXT(D2) QUAL] [0]
 MATCH(T1) [TEXT(D1, D2) QUAL]
 BNC LL@1, TEXT.RAIL_STAT(T1)
   LINK_PASS(T1, T1) [TEXT(D1, D2)]
   LINK_PASS(T1, T1) [TEXT(D1, D2)] /* For cell boundary. */
   RSET [REGU_1, 'B']
   MATCH(T2) [TEXT(D1, D2): MKED(T1)]
   BNC LL@1, TEXT.RAIL_STAT(T2)
     CROSS_MARK(T2) [DOCUMENT:DOCID = TEXT.DOCID] [MKED(T2)]
LL@1
MEND
```

Macro definition of the <A,B>n

```
 MACRO
 MATCH$A$NWORDS$B A, B, N, TEXT, D1, D2, T3, T2, T4, T13, T1, QUAL
 /* ... ** Match A (N words) B  */
 MARK(T13) [TEXT QUAL]
 REPLACE [TEXT(D2) QUAL] [0]
LL@1
 BNC LL@5, TEXT.RAIL_STAT(T13)
 BNC LL@2, RAIL_STAT(T1) RESET(T1) [TEXT: MKED(T1)] LL@2
 RSET [REGU_1, 'A']
 MATCH(T1) [TEXT(D1, D2): MKED(T13)]
 BNC LL@5, TEXT.RAIL_STAT(T1)
 RESET(T13) [TEXT: UNMKED(T1)]
 RSET [REGU_1, 'B']
 MATCH_WWC(T2) [TEXT(D1, D2, D3): MKED(T1)] [N]
 BNC LL@4, TEXT.RAIL_STAT(T2)
LL@3
 CROSS_MARK(T1T3) [DOCUMENT:DOCID = TEXT.DOCID] [MKED(T2)]
 CROSS_MARK(T3) [TEXT: DOCID = DOCUMENT.DOCID] [MKED(T1)]
 RESET(T3T13T14) [TEXT: MKED(T3)]
LL@4
```

```
 BNC LL@1, TEXT.RAIL_STAT(T14)
   LINK_PASS(T4, T14) [TEXT(D3)]
   MATCH_WWC(T2) [TEXT(D1, D3): MKED(T4)]
   RESET(T4) [TEXT: MKED(T4)]
     BC LL@3, RAIL_STAT(T2)
     BC LL@4
LL@5
 MEND
```

Macro definition of the (A,B,C,D) # n

```
 MACRO
 MATCH$THRESHOLDOR M, N, TEXT, D1, D2, T1, QUAL
 /* ... ** Match threshold or: match M words out of N words:  */
 RSET [REGU_2, N]
 REPLACE [TEXT(D2) QUAL] [0]
LL@1
 READ_REG [REGU_1]
 MATCH(T1) [TEXT(D1) QUAL]
 BNC LL@2, TEXT.RAIL_STAT(T1)
   CROSS_MARK(T1) [DOCUMENT:DOCID = TEXT.DOCID] [MKED(T1)]
   CROSS_MARK(T1) [TEXT:DOCID = DOCUMENT.DOCID] [MKED(T1)]
   ADD [TEXT(D2): MKED(T1)] [1]
   RESET(T1) [TEXT: MKED(T1)]
LL@2
 DEC_REG [REGU_2]
 BC LL@1, REGU_2 ) 0
 MARK(T1) [TEXT: D2 )= M]
 CROSS_MARK(T1) [DOCUMENT:DOCID = TEXT.DOCID] [MKED(T1)]
 MEND
```

VITA

VITA

Fazlı CAN was born in Ankara, Turkey, on December 21, 1951. He received the BS degree from the Department of Electrical Engineering of Middle East Technical University (METU), Ankara, Turkey, in 1976.

He continued his studies in the Department of Computer Engineering at METU and obtained his MS in 1979. During his graduate studies he worked as a research assistant and as an instructor in the Department of Electrical Engineering and thought courses on basic electrical engineering, numerical analysis, data structures, and information retrieval. During his Ph.D study, he went to the USA and worked as a research assistant at the Department of Computer Science of Arizona State University in 1982 and 1983. There, he was awarded an industrial fellowship. During the summer of 1982, he worked as a summer hire engineer in the RAP project at INTEL corporation of Arizona. His research interests include problems relating to automatic information retrieval and database management systems. He has published articles in the Journal of the American Society for Information Science, and the Information Technology : Research and Development. He has also presented papers at the national and international conferences.

He is a citizen of the Republic of Turkey.