1-1-1993

Concept-based text classification

Shanthi Kumar Adloori

University of Nevada, Las Vegas

Follow this and additional works at: https://digitalscholarship.unlv.edu/rtds

Repository Citation
https://digitalscholarship.unlv.edu/rtds/296

This Thesis is brought to you for free and open access by Digital Scholarship@UNLV. It has been accepted for inclusion in UNLV Retrospective Theses & Dissertations by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
Concept-based text classification

Adloori, Shanthi Kumar, M.S.
University of Nevada, Las Vegas, 1993
CONCEPT-BASED TEXT CLASSIFICATION

by

Shanthi K. Adloori

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Department of Computer Science

University of Nevada, Las Vegas

August, 1993
The thesis of Shanthi K. Adloori for the degree of Master of Science in Computer Science is approved.

Chairperson, Kazem Taghva, Ph.D

Examinining Committee Member, Tom Nartker, Ph.D

Examinining Committee Member, Ajoy Kumar Datta, Ph.D

Graduate Faculty Representative, Ashok Singh, Ph.D

Graduate Dean, Ronald W. Smith, Ph.D

University of Nevada, Las Vegas

May, 1993
The objective of this thesis is to do automatic concept based document classification. Classification or clustering is the process of grouping similar objects together so that they can be effectively retrieved when queried upon. An experimental system that does this concept based document classification is built by a series of steps such as - indexing the documents, assigning weights to the keywords, generating the document vectors, building concept lists, generating concept vectors, finding the similarity between the concept and document vectors and classifying the documents under concepts depending on the similarity value. The performance of this experimental system is evaluated against a manual system by measuring precision and recall.
Contents

ABSTRACT ............................................................................................................................. iii
LIST OF FIGURES .............................................................................................................. v
ACKNOWLEDGMENTS ................................................................................................... vi

1 Introduction 1
1.1 Text Retrieval Systems ................................................................. 1
    1.1.1 Inverted File Model ............................................................... 2
    1.1.2 Vector Space Model ............................................................ 6
    1.1.3 Probabilistic Retrieval Model ............................................ 14

2 A Review of Clustering Techniques 17
2.1 Automatic Document Classification .................................. 17
    2.1.1 Single-link Clustering ...................................................... 19
    2.1.2 Complete-link Clustering ................................................. 23
    2.1.3 Centroid method ............................................................. 25
    2.1.4 Graph theoretical Clustering ............................................ 28

3 Experimental System 33
3.1 Experiment Description ......................................................... 33
    3.1.1 Document Vector Generation ......................................... 34
    3.1.2 Concept Vector Generation ............................................. 37
    3.1.3 Selection of A Similarity Coefficient ............................... 40
    3.1.4 Classification Phase ....................................................... 41
    3.1.5 Performance Evaluation ................................................ 46

4 Conclusions 51
4.1 Conclusions and Future Work ................................................. 51

BIBLIOGRAPHY ................................................................................................... 53
List of Figures

1.1 Vector representation of a document space ....................................................... 9
2.1 Single-link clustering process ................................................................. 22
2.2 Complete-link clustering process .......................................................... 25
2.3 A clustered document organization ......................................................... 26
2.4 A structure representing the notion of cluster profiles ......................... 28
2.5 Connected component and its maximal complete subgraph ................. 32
3.1 precision and recall .................................................................................... 47
ACKNOWLEDGMENTS

At the outset, I would like to thank Dr. Kazem Taghva, for all his painstaking efforts and the help he has extended to me during the course of my graduate program. But for his constant encouragement and thoughtful guidance, completion of this thesis would have been beyond my capability. I extend my sincere gratitude to Dr. Tom Nartker for his constructive criticism and many useful discussions. I would like to thank Dr. Ajoy Kumar Datta for all his help and advise during various phases of my graduate study at UNLV. Special thanks are due to Dr. Ashok Singh, for agreeing to be on my thesis committee. I would also like to thank Dr. John Minor, Chairman of Computer Science Department for all the help extended to me during my graduate study here. And last, but not the least, I would like to thank Andrew Bagdanou, Allen Condit and Julie Borsack for their help on various portions of this thesis. With love and affection I dedicate this humble effort to my parents Mr. Lakshman Raj, Mrs. Sujatha Devi and my wife Shailaja. Finally I thank God for making all this happen.
Chapter 1

Introduction

1.1 Text Retrieval Systems

A Text Retrieval system is an information system which is designed to store information in an organized way such that only relevant portion of it is retrieved upon a user's request. The information here, in this case, is the natural language text existing in books, journals or technical reports. If the information is stored in a document, then the text retrieval system would deal with the content representation and storage structure of that document. In an operational environment, the text retrieval system will take the natural language text as input and produces a collection of references as output in response to a user query.

Database management systems and text retrieval systems although share certain common features, yet they differ in how information is stored. In a database management system, a set of homogeneous records constitute a file and these records are in turn characterized by a set of attributes. The retrieval of any particular record is very easy as the query consists of a subset of the attributes describing that record. For a text retrieval system,
that handles bibliographic records and textual data, more emphasis should be given to the descriptors which identify the actual text content. This emphasis is required because most of the retrieval operations depend on the content representation of the stored document.

The descriptors which identify the text content are known as *keywords* or *index terms*. A document can be represented as a collection of these keywords. And a particular document can be retrieved based on the similarity between the sets of keywords identifying the query and document rather than depending on an exact match between the query and document terms.

There are three basic models of text retrieval systems. They are:

- Inverted file model
- Vector space model and
- Probabilistic retrieval model

In the following sections a detail description of the above models is presented.

1.1.1 Inverted File Model

A set of keywords, known as term vector, represent a stored document. These keywords may be assigned weights depending on how important they are in identifying that document. Similarly a query can also be thought of as a set of weighted keywords.
There are two important aspects of any retrieval strategy. The first being the ability to give very quick access to the stored documents. And the second being the capability to handle very large number of keywords in a given situation.

A search strategy that does linear search of documents cannot be used if a quick access is needed. In such a case, it becomes mandatory to maintain a dense index for each term. In other words, building a separate index for each term, containing information about the document identifiers, enables fast access to the stored documents. The collection of all these indexes for all the terms that represent the document content is known as inverted index or inverted file.

An inverted file mechanism can be thought of as a three step process. In the first step, the whole document collection is represented as a two dimensional array. The rows of this array represent documents and columns represent the keywords attached to these documents. In the second step, the rows and columns of the document-term array are interchanged. That is, each row of the inverted document-term array now represents all the documents in a collection identified by a particular term. In the third step, a few boolean operations are done on the rows of the inverted document-term array so that only relevant documents are retrieved that satisfy a user's query.

A detailed illustration of the above process is given in tables 1.1 and 1.2. Consider a collection of three documents, \(DOC_1, DOC_2, DOC_3\), where each document is represented by a set of four keywords, \(Term_1, Term_2, Term_3, Term_4\), as shown in Table 1.1. The presence
or absence of a keyword in a particular document is indicated by a 1 or 0 respectively. The first row in Table 1.1, which is the term vector (0, 1, 0, 1), identifies DOC1. It is evident from this term vector that only two terms, Term2 and Term4, are assigned to DOC1 and not the other two terms. After inverting the document-term array of Table 1.1, we get the inverted file as shown in Table 1.2, where each row corresponds to the set of all documents identified by that term. That is, the vector represented by (0, 1, 0) in the first row of Table 1.2 corresponds to the list of documents identified by Term1. In this case, it can be observed that Term1 is present in DOC2 only and not in DOC1 or DOC3. The process of inverting the document-term array to an inverted file is exactly the same as transposing a matrix.

One main advantage of using an inverted file is that the retrieved documents in response to a query need not be analyzed for their relevance because it is known beforehand that the query term is used to index these particular documents.

<table>
<thead>
<tr>
<th></th>
<th>Term1</th>
<th>Term2</th>
<th>Term3</th>
<th>Term4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Document2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Document3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1.1: Document-term array

<table>
<thead>
<tr>
<th></th>
<th>Document1</th>
<th>Document2</th>
<th>Document3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Term2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Term3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Term4</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1.2: Inverted File
method which is used in inverted file operation is the use of boolean expressions. A detailed description of this method is given below.

Query statements using Boolean Expressions

In an information retrieval environment a user may wish to retrieve only those documents that are related to his topic of interest. That is, he needs a more precise collection of documents in response to his query. Consider, for example, a user wants to find all the documents containing the terms ‘Science’ and ‘Engineering’. In order to accomplish this it becomes more appropriate to search for these terms in the inverted index instead of searching for them in the documents themselves. This can be done with the implementation of Boolean Logic to form the search requests using the boolean operators AND, OR and NOT.

- If the user submits his query to find all the documents represented by “Science AND Engineering”, then the retrieval system would first compute two sets of indexes from the inverted index corresponding to each term. These sets will contain the list of documents identified by these terms. The intersection of these two sets will give the list of documents that contain both ‘Science’ and ‘Engineering’ terms.

- If the user submits his query as “Science OR Engineering”, then it implies that he is interested in all the documents which are identified by ‘Science’ or ‘Engineering’ or by both terms. Proceeding in the same way as for the AND operator, two sets of indexes are constructed from the inverted index corresponding to each term. But here the union of the two sets gives the list of documents that satisfy the above query.
• If the user submits his query as "Science NOT Engineering", then it implies that he needs only references to those documents that are identified by 'Science' and not by 'Engineering'. Then, in such a case, the retrieval system would find the two sets of indexes from the inverted index for each term and the difference of these two sets would yield the answer to the user's query.

Many of the commercially available text-retrieval systems are based on this inverted file mechanism. A few of them are:

1. The DIALOG System developed by Lockheed Information Systems.

2. The STAIRS System developed by IBM Corporation.

3. The Bibliographic Retrieval Services (BRS) System which has its origins in the biomedical communication network of the State University of New York.

4. The MEDLARS System developed by National Library of Medicine.

5. The LEXIS System developed by Mead Data Central.

1.1.2 Vector Space Model

Vector space model is the simplest and most efficient of all the information retrieval models. The reason for it being the simplest is the fact that all the terms, documents and queries are represented as vectors in a vector space. And also, since the basic vector operations can be performed on them, the model tends to be more effective and efficient in storing and retrieving information.
In a vector space, if each term is represented as a vector of unit length and considering a document to be a collection of terms, then each document can be represented as a linear combination of these term vectors. Say, for example, a document and a query can be represented as

\[
\text{Doc}_1 = (t_1, t_2, t_3, \ldots, t_n) \quad \text{and} \quad \text{query}_1 = (q_1, q_2, q_3, \ldots, q_n)
\]

where each \( t_i \) represent term identifying the document and each \( q_i \) represent the term identifying the query. These terms can have either binary values or weighted values. That is, in a binary form, a term can be assigned a value of 1 if it is present in the document and a value of 0 is assigned if it is absent. And similarly a term can be assigned a weight depending on how important the term is in identifying the document content. Given the vector representation for the documents and queries, similarity can be measured between these two vectors and relevant documents can be retrieved based on their similarity value.

There are various ways of computing similarity values. Out of these measures, the one most commonly used is the cosine coefficient. The cosine coefficient, which is given below, is measured as a function inversely related to the angle between two vectors.

\[
\cosine(Doc_i, query_j) = \frac{\sum_{k=1}^{n}(t_{ik} \cdot q_{jk})}{\sqrt{\sum_{k=1}^{n}(t_{ik})^2 \cdot \sum_{k=1}^{n}(q_{jk})^2}}
\]

The cosine similarity coefficient can be used to measure the angle between two document vectors or between document and query vectors. In particular, if two documents are
represented as

\[ \text{Doc}_1 = (0.3, 0, 0.1) \]
\[ \text{Doc}_2 = (0.4, 0.8, 0) \]

and a query is represented as

\[ \text{query}_1 = (0.3, 0.5, 0) \]

then using the cosine coefficient, the similarity between the query and the documents is found as

\[ \text{sim} (\text{Doc}_1, \text{query}_1) = \frac{(0.3)(0.3)+(0)(0.5)+(0.1)(0)}{\sqrt{(0.3^2+0^2+0.1^2)(0.3^2+0.5^2+0^2)}} \]

\[ \text{sim} (\text{Doc}_1, \text{query}_1) = 0.488 \]

\[ \text{sim} (\text{Doc}_2, \text{query}_1) = \frac{(0.4)(0.3)+(0.8)(0.5)+(0)(0)}{\sqrt{(0.4^2+0.8^2+0^2)(0.3^2+0.5^2+0^2)}} \]

\[ \text{sim} (\text{Doc}_1, \text{query}_1) = 0.997 \]

Pictorially, \text{Doc}_1, \text{Doc}_2 \text{ and } \text{query}_1 \text{ can be viewed as shown in Figure 1. The dimension of vector space depends on the number of terms present in a vector. For the sake of simplicity only three terms(\text{Term}_1, \text{Term}_2, \text{Term}_3) are considered. Since similarity value is judged by the angle between two vectors, the two vectors represented by \text{Doc}_2 \text{ and } \text{query}_1 \text{ are nearer to each other. That is, } \text{Doc}_2 \text{ and } \text{query}_1 \text{ are more similar to each other having a value} \]
of 0.997 than $Doc_1$ and $query_1$ which have a similarity value of 0.488. In fact, the more similar two vectors are the smaller is the angle between them. Following this fact it can be stated that if a document vector and a query vector are similar to each other, then the angle between them would be equal to zero and their corresponding vectors in the vector space would be superimposed on each other.

![Vector representation of a document space](image)

Figure 1.1: Vector representation of a document space

There are many measures of similarity which can be used to compute the closeness between any two vectors. Considering $X$ and $Y$ to be two vectors and the counting measure $|X|$ gives the length of the vector, the different similarity coefficients are summarized in Table 1.3.

The inner product similarity coefficient is the simplest of all. It measures the number of common terms between two vectors in a binary form and it measures the product of the weights when the terms are assigned weights. The inner product does not take into account
The length of the vector whereas others such as Dice, Cosine and Jaccard do account for the length of the vector. The process of normalizing the vector, dividing the vector by its length, is important because all the documents are not of equal length. By normalizing the vectors, the similarity between varying lengths of text can be properly judged.

One of the most important aspect of information retrieval is the representation of the document. That is, how to represent the document so that it can be efficiently stored and effectively retrieved when queried upon. The process of identifying terms that represent document content is known as indexing. Indexing can be done both manually and automatically. When trained subject experts pick up the terms that represent the document, it is known as manual indexing. When this process is carried out with the help of a computer program, it is known as automatic indexing.

Indexing aids in studying the characteristics of a term. That is, we can know how frequently it occurs and where it occurs in the natural text. Based on the frequency of

<table>
<thead>
<tr>
<th>similarity(X,Y)</th>
<th>for binary vectors</th>
<th>for weighted vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inner product</td>
<td>(</td>
<td>X \cap Y</td>
</tr>
<tr>
<td>Dice coefficient</td>
<td>(\frac{2</td>
<td>X \cap Y</td>
</tr>
<tr>
<td>Cosine coefficient</td>
<td>(\frac{</td>
<td>X \cap Y</td>
</tr>
<tr>
<td>Jaccard coefficient</td>
<td>(\frac{</td>
<td>X \cap Y</td>
</tr>
</tbody>
</table>

Table 1.3: Measures of vector similarity
occurrence, words can be categorized into high frequency, medium frequency and low frequency words. It has been studied by Luhn [6] that the medium frequency words are the best descriptors to represent the document content and have the capability to distinguish relevant from non-relevant information. This capability of distinguishing relevant from non-relevant information is known as “resolving power” of words. High frequency words cannot be used as the document descriptors because they appear in almost every document of the collection and do not have the capability to distinguish documents from each other. And even low frequency words cannot be used for content identification because they rarely appear in documents and hence cannot distinguish documents from each other.

The frequency of occurrence of words has an important role in assigning weights in a term weighting system. There are various ways of assigning weights to terms depending on how important they are in identifying the document content. Words can be assigned a weight values ranging from 0 to 1. A weight value nearing 0 can be assigned to terms of low significance and a weight value nearing 1 can be assigned to higher significant terms. A few of the term weighting strategies are discussed below:

(1) Weighting based on term frequency alone:

A frequently occurring term has definitely something to do with the document content. In such a case, the weight of such a term may be equal to the number of times it appears in a document. That is,

\[ wt_{ij} = freq_{ij} \]
where $wt_{ij}$ represents the weight of the term $j$ in document $i$ and $freq_{ij}$ represents the frequency of the term $j$ in document $i$.

(2) Weighting based on collection frequency:

The term frequency weighting system considers the frequency of a term in one particular document and does not consider for the frequency in all other documents of the collection. According to theoretical studies, it has been found that the weight or importance of term is inversely proportional to the document frequency, i.e, the number of documents in which term $j$ appears. This gives rise to a factor called inverse document frequency ($idf$). Consider a collection of $N$ documents, the term frequency ($freq_{ij}$) and a document frequency ($docfreq_j$), a possible measure for the inverse document frequency is given by

$$idf = \log \frac{N}{docfreq_j} + 1$$

$$idf = \log(N) - \log(docfreq_j + 1)$$

A typical weighting function which is based on term frequency and the inverse document frequency is given by

$$wt_{ij} = freq_{ij} \cdot idf$$

$$wt_{ij} = freq_{ij} \cdot [\log(N) - \log(docfreq_j) + 1]$$
It can be observed from the above weight functions that the weight of term \( j \) increases when \( freq_{ij} \) increases and the weight decreases when the \( docfreq_j \) increases. This weighting strategy is best suited for terms appearing in only a few documents of the collection and it assigns a high degree of importance for such terms [5].

(3) Weighting based on term discrimination value:

According to the term discrimination theory [9], the discrimination value (dv) of a term is measured as a degree to which the assignment of this term to a particular document will decrease the document space density. Document space density is defined as the average distance between the documents in a collection. And it is computed as the average pairwise similarity between all pairs of documents.

The discrimination value of a particular term \( j \), denoted by \( dv_j \), can be computed as the difference of space densities before and after assignment of the term \( j \). That is, if the space density with term \( j \) is represented by \( sd \) and the space density without term \( j \) is represented by \( sd_j \) then the discrimination value of term \( j \) is given by

\[
dv_j = sd - sd_j
\]

The discrimination value \( dv_j \) obtained for the term \( j \) may be positive, negative, or has a value nearing zero. If the discrimination value is positive, then such a term is known as good discriminator and the assignment of such a term will definitely decrease the document space density. If the discrimination value is negative, then such a term is called a bad dis-
classifier and the assignment of such a term will leave the document space more compact. The terms which have a discrimination value nearing zero do not have any effect on the document space density even if they are present or absent.

A typical term weighting strategy using the term discrimination value and the term frequency in a particular document is given by

\[ wt_{ij} = freq_{ij} \cdot dv_j \quad (dv_j > 0) \]

where \( wt_{ij} \) represents the weight of a particular term \( j \) in a document \( i \), \( freq_{ij} \) represents the frequency of the term \( j \) in document \( i \) and \( dv_j \) represents the discrimination value of term \( j \). Using this weighting system which incorporates the term discrimination value, high weight values are assigned to those terms which help in decreasing the document similarities and thereby decreasing the document space density.

### 1.1.3 Probabilistic Retrieval Model

A probabilistic retrieval model is based on the probability theory and the retrieval process in such a model includes the dependencies and relationships between terms which are not considered in other models. Consider the binary representation for a document vector \( D = (t_1, t_2, t_3, \ldots, t_n) \), where \( t_i \) is equal to zero or one depending on whether the term is absent or present. It is necessary for a retrieval system to classify the document to be either relevant or nonrelevant for a particular query. This can be done by retrieving the documents in response to a query if the probability of relevance is sufficiently larger[2]. Probabilistic models are built not only for binary terms but also for weighted terms[3].
In a probabilistic model there are two major parameters, Prob(relevant) and Prob(nonrelevant). Prob(relevant) refers to the probability of relevance and Prob(nonrelevant) refers to the probability of nonrelevance of a document. Considering relevance to be a binary function, we can say that Prob(nonrelevant) = 1 - Prob(relevant). Assuming $c_1$ to be the loss associated with retrieving a nonrelevant document and $c_2$ to be the loss associated with not retrieving a relevant document, a loss minimization function can be computed. If the loss associated with the retrieval of a nonrelevant document is given by $c_1 \cdot [1 - \text{Prob(relevant)}]$ and the loss associated with the rejection of a nonrelevant document is given by $c_2 \cdot \text{Prob(relevant)}$ then the loss minimization function is given by

$$c_2 \cdot \text{Prob(relevant)} \geq c_1 \cdot [1 - \text{Prob(relevant)}]$$

Similarly, a retrieval function $M$ is given by

$$M = \frac{\text{Prob(relevant)}}{1 - \text{Prob(relevant)}} - \frac{c_1}{c_2}$$

A document is retrieved whenever its retrieval function value, $M$, is positive. In fact, calculating these probabilities directly is very difficult. But using Bayes' theorem and assuming the cost parameters, $c_1$ and $c_2$, to be equal, a new retrieval function is given by

$$\log M(D) = \log \frac{\text{Prob}(D|\text{relevant})}{\text{Prob}(D|\text{nonrelevant})} + \log \frac{\text{Prob}(\text{relevant})}{\text{Prob}(\text{nonrelevant})}$$

In the above function, the parameters Prob(relevant) and Prob(nonrelevant) are the a priori probabilities. There are different models because of different expressions for
Prob(D|relevant) and Prob(D|nonrelevant). In a term independence model it is assumed that the keywords occur independently in the relevant and nonrelevant documents. Then

\[
Prob(D|relevant) = Prob(t_1|relevant)Prob(t_2|relevant) \ldots Prob(t_n|relevant)
\]

Similarly Prob(D|nonrelevant) can be defined. Let \( a_i = Prob(t_i|relevant) \) and \( b_i = Prob(t_i|nonrelevant) \). The terms \( a_i \) and \( b_i \) represent the probabilities that a term is found in a relevant and nonrelevant document respectively. Then

\[
Prob(D|relevant) = \prod_{i=1}^{n} a_i^t_i (1 - a_i)^{1-t_i},
\]

\[
Prob(D|nonrelevant) = \prod_{i=1}^{n} b_i^t_i (1 - b_i)^{1-t_i} \quad \text{and}
\]

\[
M(D) = \sum_{i=1}^{n} t_i \log \frac{a_i(1-b_i)}{(1-a_i)b_i} + \sum_{i=1}^{n} \log \frac{1-a_i}{1-b_i}
\]

Probabilistic models are known for their usefulness in judging the theoretical concepts. But for practical retrieval, there is no improvement in efficiency over other models because of the estimation of the parameters for relevance.
Chapter 2

A Review of Clustering Techniques

2.1 Automatic Document Classification

Classification or clustering can be defined as grouping of similar objects together so that they can be effectively stored and efficiently retrieved. The objects here refer to either the keywords identifying the document content or the documents themselves. When keywords are involved it is known as term clustering and much work has been done in this area by S. Jones [4]. On the other hand, when documents are involved then it is known as document clustering. Document classification can be based on an hypothesis called cluster hypothesis stated by Van Rijsbergen [11]. According to his hypothesis, closely associated documents tend to be relevant to the same request. Storing of documents into related classes will make the retrieval process very fast and also, since these classes contain only relevant documents for a particular query, it is very effective. Fast access and effectiveness are the two key
There are two methods of grouping documents. They are nonhierarchic and hierarchic methods. In a nonhierarchic clustering method, a set of documents is divided into subsets with similar documents in a cluster separated from nonsimilar documents in other clusters. The clusters generated in this manner do not exhibit any hierarchic relationship between each other. In a hierarchic clustering method, the classification process results in a tree-like structure. The leaves of this tree represent the individual documents and each node represents the cluster generated while the classification operation is in progress. The root of this tree represents the single cluster containing all the documents of the collection.

Hierarchical clustering is in turn divided into two main strategies. One is known as agglomerative clustering and the other is divisive clustering. The agglomerative clustering method can be thought of as a process which builds a tree upwards starting from the leaves and ending in the root. That is, if there is a collection of n documents then there are (n-1) fusions to result in a tree-like classification. In contrast to this, a divisive clustering method starts with all documents in a collection as a single cluster and then splits it into smaller and smaller clusters.

Few examples of an agglomerative hierarchical clustering are single-link clustering and complete-link clustering. All the agglomerative hierarchical clustering methods can be described by one single algorithm which is given below:
Algorithm CLUSTER:

begin

FOR index1 = 1 to (n-1) DO

FOR index2 = index1 to n DO

compute sim-coeff[index1, index2];

REPEAT

find the most similar remaining pair of clusters;

form a new cluster by grouping this pair, M and N;

recompute the similarity values between MN and each of the remaining

clusters and update sim-coeff

UNTIL there is only a single cluster of n items.

end

In the following subsections the single-link clustering and the complete-link clustering methods are discussed in detail. Two other hierarchic methods, one based on centroid method and the other based on graph theoretical approach, are also discussed.

2.1.1 Single-link Clustering

Single-link clustering is also known as nearest neighbor method. In this method clusters are generated on the basis that the similarity between two clusters is considered as the similarity between the most similar document pair, one of which is in each cluster. Classifying documents using this method satisfy a property that each document in a cluster is more similar to at least one document of that cluster than it is to any other document in any other cluster.
The single-link clustering method can be best described by the following example. Consider a collection of five documents (d1, d2, d3, d4, d5) to be clustered. Assuming a document to be a set of key words, pairwise similarity between documents can be found using one of the similarity measures. After computing these values a document-document similarity matrix can be constructed as shown in Table 2.1.

At stage one of this classification process, two documents, d1 and d3, which have the highest similarity value (0.8) are grouped into one cluster. Then the similarity matrix is reconstructed by replacing the rows and columns of d1 and d3 with \{d1d3\}. The similarity values of this new cluster \{d1d3\} with all the other documents (d2, d4 and d5) is calculated.
Table 2.4: similarity matrix after d1d2d3-d4 are clustered

<table>
<thead>
<tr>
<th></th>
<th>d1d2</th>
<th>d2d3</th>
<th>d4</th>
<th>d5</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1d2</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>d2d3</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>d4</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>d5</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

as the maximum similarity between them. That is, the similarity values are given by:

\[
\max\{(d1d3, d2)\} = \max\{d1d2, d2d3\} = \max\{0.2, 0.7\} = 0.7
\]
\[
\max\{(d1d3, d4)\} = \max\{d1d4, d3d4\} = \max\{0.5, 0.3\} = 0.5
\]
\[
\max\{(d1d3, d5)\} = \max\{d1d5, d3d5\} = \max\{0.3, 0.1\} = 0.3
\]

This gives rise to a new similarity matrix which is shown in Table 2.2. At stage two, the cluster \{d1d3\} is grouped with the document d2 since it has the maximum similarity value (0.7) in the newly constructed similarity matrix (Table 2.2). The new cluster generated at stage two is \{d1d2d3\}. Again the similarity matrix is rebuilt by calculating the similarities between this new cluster and the other documents (d4 and d5). The new similarities are calculated as:

\[
\max\{(d1d2d3, d4)\} = \max\{(d1d3, d4), (d2, d4)\} = \max\{0.5, 0.4\} = 0.5
\]
\[
\max\{(d1d2d3, d5)\} = \max\{(d1d3, d5), (d2, d5)\} = \max\{0.3, 0.2\} = 0.3
\]

The new similarity matrix is shown in Table 2.3. At stage three, the cluster \{d1d2d3\} is grouped with the document d4 as it has the maximum entry of similarity value (0.5) in Table 2.3. Similarly the new similarities between the cluster \{d1d2d3d4\} and d5 is calculated and the new similarity matrix is built as shown in Table 2.4. Since there are only two elements in this similarity matrix, \{d1d2d3d4\} and d5, these are grouped into one cluster in stage four. Thus we have the final cluster obtained by single-link clustering method as shown in Figure 2.1. In this figure the various fusions at each stage are clearly represented.
output of this single-link clustering method is a tree like structure where each node in this tree represents a cluster. If the similarity value at each stage is considered as the threshold then the clusters formed so far are the clusters obtained for that particular threshold value.
2.1.2 Complete-link Clustering

Complete-link clustering is also known as furthest neighbor method and it operates in a way exactly the opposite of single-link method. Using this method documents are grouped on the basis of the similarity between the least similar pair of documents from two clusters. A particular document in a cluster is more similar to the least similar document in that cluster than to least similar document in any other cluster.

The complete-link clustering can be described by using the same similarity matrix as given in Table 2.1. At stage one documents d1 and d3 are grouped first into a cluster since they have the highest similarity value (0.8) in the matrix. The rows corresponding to the documents d1 and d3 are replaced with one single row, dld3. The similarity matrix is rebuilt by computing the similarity values between the cluster {dld3} and all other documents d2, d4 and d5. In a single-link clustering method the similarity between the cluster and other
document is computed as the maximum similarity between them but here in a complete-link method the minimum similarity is considered. It is given as:

\[
\begin{align*}
\min\{(d1d3, d2)\} &= \min\{d1d2, d2d3\} = \min\{0.2, 0.7\} = 0.2 \\
\min\{(d1d3, d4)\} &= \min\{d1d4, d3d4\} = \min\{0.5, 0.3\} = 0.3 \\
\min\{(d1d3, d5)\} &= \min\{d1d5, d3d5\} = \min\{0.3, 0.1\} = 0.1
\end{align*}
\]

and these values constitute the new similarity matrix as given in Table 2.5. At stage two, documents d2 and d4 are grouped since they have the maximum similarity value (0.4) in the newly constructed similarity matrix (Table 2.5). Again the similarity matrix is rebuilt by calculating the similarities between the clusters \{d1d3\}, \{d2d4\} and the document d5. These values are calculated as:

\[
\begin{align*}
\min\{(d1d3, d2d4)\} &= \min\{(d1d3, d2), (d1d3, d4)\} = \min\{0.2, 0.3\} = 0.2 \\
\min\{(d2d4, d5)\} &= \min\{d2d5, d4d5\} = \min\{0.2, 0.1\} = 0.1
\end{align*}
\]

The new similarity matrix is shown in Table 2.6. At stage three, the clusters \{d1d3\} and \{d2d4\} are grouped together to form a new cluster \{d1d2d3d4\} since they have the maximum entry similarity value in Table 2.6. Similarly the new similarities between the cluster \{d1d2d3d4\} and d5 is calculated and the new similarity matrix is built as shown in Table 2.7. Since there are only two elements in this similarity matrix, \{d1d2d3d4\} and d5, these are grouped into one cluster in stage four. Thus we have the final cluster obtained by the complete-link clustering method as shown in Figure 2.2. In this figure the various clusters formed at different similarity values (at different stages) are depicted. The output for this method is also a tree like structure where each node represents a cluster. If at each stage the similarity value is considered as the threshold then we can think of the clusters gen-
erated up to that stage are the different clusters obtained for that particular threshold value.

![Complete-link clustering process](image)

**Figure 2.2: Complete-link clustering process**

### 2.1.3 Centroid method

A *cluster representative* is a document which represents and identifies the documents in that cluster. The cluster representative is also known as *cluster profile* or *centroid*. Given the
vector representation for a document, a centroid for a document cluster can be calculated as the vector sum or the average of all the documents in that cluster. Practically this centroid will be near to every document in that cluster. In particular, if three documents are represented as

\[
doc1 = \{0.5, 0.4, 0.3\} \\
doc2 = \{0.3, 0.6, 0.1\} \\
doc3 = \{0.1, 0.5, 0.2\}
\]

then the centroid can be computed as:

\[
\text{centroid} = \left\{ \frac{0.5+0.3+0.1}{3}, \frac{0.4+0.6+0.5}{3}, \frac{0.3+0.1+0.2}{3} \right\} \\
\text{centroid} = \{0.3, 0.5, 0.2\}
\]

Figure 2.3: A clustered document organization

A typical clustered document space is shown in Figure 2.3. In this figure each \(X\) represents an individual document and the circles bounding these \(X\)s are the clusters. The
centroid for each cluster is represented by a small filled circle. Also the centroid for the
document space is represented by a small filled square.

The process of generating clusters can be best described by an algorithm known as
single-pass algorithm [8]. The operation of this single-pass algorithm is given below:

1. The documents are processed serially.

2. The first document is associated with cluster one.

3. The next document is compared with cluster one and is added to it if the similarity
   value is sufficiently larger else another cluster is generated.

4. If there is more than one document in a recently generated cluster then compute the
   centroid for that cluster.

5. Get the next document and compare it with all the existing cluster centroids. Add
   this document to all those clusters with which it has sufficiently large similarity value.
   Otherwise generate a new cluster with this document as the cluster representative.

6. If a new document is added to the existing cluster then recompute the cluster centroid.

7. Repeat steps 5 and 6 until all the documents are processed.

Clustering using the single-pass algorithm proceeds in a bottom-up fashion. That is,
it considers one document at a time and tries to group them into clusters as it proceeds.
When the clusters are generated out of the documents the centroids for these clusters are
obtained by using this algorithm. If the centroids generated are large in number then this
classification scheme can be extended to another level where the objects to be clustered are
now the centroids. That is, we can compute the super centroids of the existing centroids. Similarly, it can also be extended to another level where we compute the hyper centroid of the existing super centroids. A typical document structure using this notion is shown in Figure 2.4. In this figure each individual document is represented as $X$. The centroids for these document clusters are represented by small filled circles. The super centroids of these centroids are represented by small filled squares. And finally, the hyper centroid of all the super centroids is represented as a filled diamond.

![Figure 2.4: A structure representing the notion of cluster profiles](image)

### 2.1.4 Graph theoretical Clustering

Graph theory can be used to classify documents into clusters. Given a collection of $n$ documents where each document is represented as a vector shown below:

$$doc_i = (t_1, t_2, t_3, \ldots, t_m)$$
then depending on the interdocument similarity, document clusters in the form of a graph representation can be constructed. Using the similarity values between documents, a document-document matrix can be constructed. In graph theory, a connected component is defined as a set of nodes which can be reached mutually by traversing along the edges of the graph. We can use this definition of a connected component and define two documents to be connected if the similarity value between them exceeds a predefined threshold value. Using these connected components document clusters can be generated.

There are various algorithms developed for generating clusters based on graph theory. One such algorithm is by Bierstone [1]. According to him, a maximal complete subgraph defines the clusters in a graphical form. A maximal complete subgraph can be defined as a subgraph in which every node is connected to every other node and is not contained in any other subgraph. A connected component and its maximal complete subgraph is shown in Figure 2.5. In this figure a document-document similarity matrix is shown. If a threshold value of 0.4 is applied to the matrix, then we get the connected components. That is, a document is connected to another document when it exceeds the threshold value of 0.4. The corresponding maximal complete subgraph is also shown in Figure 2.5. The Bierstone algorithm which is given below takes in as input the connected components and outputs the maximal complete subgraphs.

Bierstone Algorithm for Finding Maximal Complete Subgraphs

- Step 1. \( i = 0, j = \) number of nodes in the input data set.
- Step 2. \( j = j - 1. \)
• Step 3. If $M_j = 0$, go to Step 2; otherwise, continue to Step 4.

• Step 4. For each $p_k \in M_j$, set $i = i + 1$ and define the complete subgraph $C_i = \{p_j, p_k\}$.

• Step 5. $j = j - 1$.

• Step 6. If $j = 0$, all input sets $M_j$ have been processed and the set of arrays $C$ represents the nodal sets of all maximal complete subgraphs of the input data set; if $j \neq 0$ continue to Step 7.

• Step 7. If $M_j = 0$, go to Step 5 to get the next input array. Otherwise, set $W = M_j, L = i$ (the number of complete subgraphs produced so far), $n = 0$, and continue to Step 8.

• Step 8. $n = n + 1$.

• Step 9. If $n > L$, all complete subgraphs $C_k$ have been searched: go to Step 17; otherwise, continue to Step 10.

• Step 10. Define the complete subgraph $T = C_n \cap M_j$. If $T$ contains fewer than 2 nodes, go to Step 8; otherwise, delete from $W$ all nodes contained in $T \cap W$ and go to Step 11.

• Step 11. If $T = C_n$ go to Step 15.

• Step 12. If $T = M_j$, set $i = i + 1$ and define the complete subgraph $C_i = T \cup \{p_j\}$, and go to Step 5 to get the next input array; otherwise, continue to Step 13.

• Step 13. If $T$ is a subset of any complete subgraph $C_q(q = 1, \ldots, n - 1, n + 1, \ldots, i)$ that contains $p_j$, ignore this complete subgraph as it is already contained in $C_n$ and go to Step 8; otherwise, set $S = T \cup \{p_j\}$ and continue to Step 14.
• Step 14. If some complete subgraph \( C_q(q = L + 1, \ldots, i) \) is a subset of the complete subgraph \( S \), redefine the complete subgraph \( C_q = S \) and delete any \( C_q(r = q + 1, \ldots, i) \) which is also a subset of \( S \); otherwise set \( i = i + 1 \) and define the new complete subgraph \( C_i = S \). Go to Step 8.

• Step 15. Redefine the complete subgraph \( C_n \) as \( C_n = C_n \cup p_j \). Delete any \( C_q(q = n + 1, \ldots, i) \) that is a subset of the altered \( C_n \). Continue to Step 16.

• Step 16. If \( T = M_j \), go to Step 5 to get the next input array; otherwise, go to Step 8.

• Step 17. For each \( p_k \) remaining in \( W \), set \( i = i + 1 \) and create the new complete subgraph \( C_i = \{p_j, p_k\} \). Go to Step 5 to get the new input array.

In the above algorithm each document is represented a node and each node \( p_j(j = 1, \ldots, n) \) is assigned a unique number. For each unique node \( p_j \) there is a corresponding array \( M_j \) where the connected components are stored. It is given by

\[
M_j = \{p_k \mid \text{the pair } (p_j, p_k) \text{ represents an edge of the graph and } k > j\}
\]

A set of arrays represented by \( C_i \) is also used where complete subgraphs are built. The algorithm builds the complete subgraphs from the set of connected nodes denoted by \( \{p_j\} \cup M_j \) where \( M_j \) is not empty. These complete subgraphs can be combined with the collection of complete subgraphs \( C_n \) which have already been constructed. Finally, the algorithm outputs maximal complete subgraphs which are all the complete subgraphs of \( C_n \). This algorithm is considered to be the most efficient and fast of all the algorithms which are used to build clusters based on graph theory.
Document - Document Similarity Matrix

<table>
<thead>
<tr>
<th></th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>d5</th>
<th>d6</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>o</td>
<td>0.4</td>
<td>0.5</td>
<td>0.3</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td>d2</td>
<td>o</td>
<td>0.7</td>
<td>0.8</td>
<td>0.9</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>d3</td>
<td>o</td>
<td>0.6</td>
<td>0.7</td>
<td>0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d4</td>
<td>o</td>
<td>0.5</td>
<td>0.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d5</td>
<td>o</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>o</td>
</tr>
</tbody>
</table>

Connected Component

Maximal Complete Subgraph

Figure 2.5: Connected component and its maximal complete subgraph
Chapter 3

Experimental System

3.1 Experiment Description

The main aim of this experimental system is to do automatic concept classification of documents and to compare the results of this system with the manual classification system.

A test collection consisting of 54 documents, ranging over a wide variety of topics, is used for the experiment. A list of over 3200 keywords, taken from the Licensing Support System Thesaurus [10], served as a dictionary for this experiment. Additionally, two subject experts\(^1\) served as a source of input for this experiment.

The experiment mainly deals with the generation of document vectors, generation of concept vectors, computing the similarity between these vectors, classifying the documents under the concepts and comparing this automatic document classification with the manual classification for performance evaluation. In the following sections each of these topics are

\(^1\text{two geology students}\)
described in more detail.

3.1.1 Document Vector Generation

In a vector space a document is viewed as a collection of term vectors. The terms which describe the document content are known as keywords or lead-terms. In order to find a list of terms that represent a document vector, the natural language text of each document is analyzed for their occurrence. There are basically two types of terms - stop words and go words. Stop words such as 'and', 'not', 'of', 'but' have high frequency of occurrence in the documents. On the other hand, the go words that actually represent document content occur with varying frequencies in the document. In fact, the frequency of occurrence is used to assign weights to the keywords. The process of document vector generation can be best described as follows:

- the document text is read for content analysis and to find the occurrence of the 3200 lead-terms in the text.
- the stop words are eliminated by consulting a list of stop words2.
- each of the remaining go words are reduced to their word stems in order to calculate the correct frequency of occurrence of each term. Reducing the words to their word stems reduces the anomalies occurring due to the various forms that the word appears in the document. For example, the lead-term shaft and the term in the document shafts do not match exactly. The stemming algorithm is adapted form Chris Paice[7].

This algorithm is iterative in nature and uses a table of rules that specify what has

---

2 The list of stopwords is taken from GNU groff software distribution
to be done if the word ends in a particular form. Depending on the final letter of the suffix the rules are grouped into sections. This makes the search in table much easier because the rule can be accessed by looking at the final letter of the word. A typical rule in the table would look like “sei3y” which implies that if a word ends in “-ies” then replace the last three letters by “-y”. For example, supplies is changed to supply. The algorithm for the stemmer is given below:

1. inspect the final letter of the form; consider the relevant section and select the first rule; if no section corresponds to that letter then terminate.

2. if the final letters of the form do not match the reversed ending in the rule then goto 4; if the word is not intact goto 4; if the word does not satisfy the conditions for a set of predefined acceptable conditions then goto 4.

3. delete the right end of the form the number of characters specified in the rule; if the continuation string is “.” then terminate; otherwise goto 1.

4. move to the next rule in the table; if the section letter has changed then terminate; otherwise goto 2.

- the term frequency \((freq_i,j)\) of a keyword \(j\) in a document \(i\) is calculated.

- the document frequency \((docfreq_j)\) of each term \(j\) is calculated. That is, the number of documents in which it occurs. The process of calculating the term frequency in a document and in the whole collection can be described as follows:

  - define a block as the number of lines contained in the document for the \(freq_i,j\)
  - or as the number of lines of all the documents combined for the \(docfreq_j\).
– read each line from the document text, delete the stopwords and reduce the go words to their word stems using the stemming algorithm.

– for each word in the list of lead-terms, match it with the stemmed word. For all matches keep track of the count in a counter.

– if a particular matched stemmed word has reappeared in either the same line or in a different line increment the count in its counter.

– if the block size is declared as the number of lines in a document then output \( freq_{ij} \), which is the value contained in the counter of each matched stemmed word.

– if the block size is declared as the number of lines of all the documents in the collection then output \( docfreq_j \), which is the value contained in the counter for each matched stemmed word.

• using the collection weighting algorithm, weights are assigned to the keywords identifying each document. That is, weight \( wt_{ij} \) for a term \( j \) in a document \( i \) is calculated as

\[
wt_{ij} = freq_{ij} \cdot [\log(N) - \log(docfreq_j) + 1]
\]

• The weight values generated by using the above formula are between 0 and 1. The document vectors are binarized by using a threshold value. The threshold considered here is the average of all the nonzero weights of the terms. Anything above this threshold value is assigned a value of 1 and anything below is assigned a value of 0 in the document vector. Binarization of document vectors is done mainly because the concept vectors generated, which will be discussed in the next section, are also binary.
A typical document vector generated using the above procedure looks like

\[ \text{Doc}_i = (0, 1, 0, \ldots, 0, 1) \]

The length of each document vector is over 3200 because the list of lead-terms taken from [10] contains over 3200 terms. Total of 54 document vectors are generated for this test collection.

3.1.2 Concept Vector Generation

The list of lead-terms contained over 3200 terms and associated with each term, there is a broader term, narrower term, related term, used-for term and a category field under which the term falls. A typical list of lead-terms would look like:
<table>
<thead>
<tr>
<th><strong>Air Pollution</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BT</td>
<td>Pollution</td>
</tr>
<tr>
<td>NT</td>
<td>Air Emission</td>
</tr>
<tr>
<td></td>
<td>Air Pollution Control</td>
</tr>
<tr>
<td></td>
<td>Air Releases</td>
</tr>
<tr>
<td>RT</td>
<td>Air Quality</td>
</tr>
<tr>
<td></td>
<td>Clean Air Act</td>
</tr>
<tr>
<td></td>
<td>Gaseous Waste</td>
</tr>
<tr>
<td></td>
<td>Particulates</td>
</tr>
<tr>
<td></td>
<td>Plumes</td>
</tr>
<tr>
<td></td>
<td>Pollutants</td>
</tr>
<tr>
<td>CA</td>
<td>Atmosphere</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Air Quality</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>UF</td>
<td>Radiological Air Quality</td>
</tr>
<tr>
<td>BT</td>
<td>Air</td>
</tr>
<tr>
<td>RT</td>
<td>Air Pollution</td>
</tr>
<tr>
<td></td>
<td>Clean Air Act</td>
</tr>
<tr>
<td>CA</td>
<td>Atmosphere</td>
</tr>
</tbody>
</table>

The acronyms BT, NT, RT, UF and CA stand for broader term, narrower term, related term, used-for term and a category respectively. The category field (CA) is the predefined concept into which the lead-term is categorized. There are about 16 concepts under which all the 3200 lead-terms would appear. These 16 concepts are given below:
Concept files are built by accumulating all the lead-terms that appear under each concept. Sixteen such files are built by pulling out the terms from the category field and grouping them into a concept file. Each of the concept file built in this fashion had a varying number of lead-terms associated with it.
The process of generating the concept vectors may be best described as follows: for each of the lead term in the list of lead-terms a search is made for its presence or absence in the concept file. A zero or one is placed in the concept vector depending on whether the lead term is absent or present in the concept file. By repeating the search process for all the terms in the list we obtain a binary vector containing 0's and 1's. The size of this vector being equal to the number of lead terms in the list. That is, the size of the concept vector generated is over 3200. A typical concept vector would look like

\[ Con_i = (0, 1, 0, \ldots, 1, 0) \]

There are total of 16 concept vectors generated for the 16 predefined concepts.

### 3.1.3 Selection of A Similarity Coefficient

A *similarity coefficient* is a resemblance coefficient for which the larger the value, the more similar the two objects being compared are. A *dissimilarity coefficient* is also a resemblance coefficient for which the smaller the value, the more similar the two objects are. The objects here in this case are document and concept vectors.

A dissimilarity coefficient is selected for this experiment. If the dissimilarity is considered as the distance between objects to be clustered then it satisfies the Euclidean properties such as the distance between two objects should be greater than zero, distance between the object and itself should be equal to zero and distance between object \( A \) and object \( B \) should be the same as the distance between object \( B \) and object \( A \). The dissimilarity coefficient used in the experiment satisfies all the properties mentioned above and is closely related to the Dice coefficient and is monotone with respect to \((1 - \text{Jaccard coefficient})\). If
the concept vector is represented as $Con_i = (x_1, x_2, x_3, \ldots, x_n)$ and the document vector as $Doc_i = (y_1, y_2, y_3, \ldots, y_n)$ then the dissimilarity coefficient ($DC$) for $Con_i$ and $Doc_i$ can be computed as

$$DC = \frac{\sum x_i(1-x_i) + \sum y_i(1-x_i)}{\sum x_i + \sum y_i}$$

The dissimilarity coefficient value obtained by using the above formula can be used when the vectors considered are binary and can be converted into a similarity coefficient ($SC$) as

$$SC = \frac{1}{1+DC}$$

Using the above formula for similarity coefficient, the similarity values are calculated between each document vector and the 16 concept vectors.

### 3.1.4 Classification Phase

In this section the two classification schemes - automatic classification and manual classification are described. This phase is the most important part of the experiment because the performance of the system depends on this phase. Automatic classification is an important notion by which we can group logically related documents together so that they can be very efficiently retrieved.

**Automatic Concept Classification of Documents**

The process of automatic concept classification of documents is described in this section. From the previous section we have the similarity coefficient values between each document and the 16 concepts. In this classification scheme, the concepts are divided into three groups - strong, related and weak groups. The three concept groups are generated for each document based on the similarity value between them. A strong concept group identifies all
those concepts, out of the 16 available concepts, which are strongly related to a particular document. Similarly the related and weak concepts groups identify the related and weak concepts for that document. The process of categorizing the 16 concepts into 3 groups can be best described by the following algorithm:

**Algorithm to classify the concepts into STRONG, RELATED, and WEAK groups**

1. For each document sort the 16 values of similarity coefficients obtained by comparing it with the 16 concepts in descending order.
2. Find the pairwise difference between each adjacent sorted similarity values.
3. Pick two highest difference values.
   
   (a) Group the concepts above the first difference value into strong concepts.
   
   (b) Group the concepts between the first and second difference values into related concepts and
   
   (c) the rest of the concepts which are below the second difference value fall into weak concepts.

The algorithm described above categorizes the 16 concepts into 3 groups for each document. All the strong concepts will occur in one cluster, the related in one cluster and the weak in another cluster. By doing this classification it makes possible to start with a specific items in a particular subject and to find related items in neighboring subject fields.
Consider for example the process of classifying the 16 concepts into three groups for a particular document having a document identification (docid) as 1972. If the similarity values obtained between this document and the 16 concepts are given as:

<table>
<thead>
<tr>
<th>Concept</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>atmosphere</td>
<td>0.500000</td>
</tr>
<tr>
<td>biology</td>
<td>0.500000</td>
</tr>
<tr>
<td>engineering</td>
<td>0.505917</td>
</tr>
<tr>
<td>equipment</td>
<td>0.502212</td>
</tr>
<tr>
<td>geography</td>
<td>0.500000</td>
</tr>
<tr>
<td>geology</td>
<td>0.502525</td>
</tr>
<tr>
<td>government</td>
<td>0.500000</td>
</tr>
<tr>
<td>hydrology</td>
<td>0.517685</td>
</tr>
<tr>
<td>management</td>
<td>0.511905</td>
</tr>
<tr>
<td>materials</td>
<td>0.507273</td>
</tr>
<tr>
<td>modeling</td>
<td>0.528090</td>
</tr>
<tr>
<td>processes properties</td>
<td>0.502000</td>
</tr>
<tr>
<td>safety</td>
<td>0.500000</td>
</tr>
<tr>
<td>socio-economic factors</td>
<td>0.502174</td>
</tr>
<tr>
<td>transportation</td>
<td>0.500000</td>
</tr>
<tr>
<td>waste</td>
<td>0.505455</td>
</tr>
</tbody>
</table>

then the classification algorithm would first sort in descending order the similarity values of the document with the concepts. That is, when the values are sorted it would look like:
After sorting the similarity values, the classification algorithm computes the differences between the adjacent similarity values. It then picks the two highest difference values. Here, in this case, the two highest difference values are between the concepts (modeling, hydrology) and (hydrology, management). And the difference values are 0.010405 and 0.005780 respectively. Since the first highest difference is between modeling and hydrology, the algorithm categorizes the concept modeling as the strong concept for the document 1972. And since
hydrology is the only concept between the two highest difference values, it is categorized as the related concept for that document. Finally, all the concepts lying below the second highest difference value are categorized as the weak concepts group. A typical clustered concept list for the document (docid:1972) is given below:

<table>
<thead>
<tr>
<th>Strongest</th>
<th>Related</th>
<th>Weak</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>modeling</td>
<td>hydrology</td>
<td>management</td>
<td>0.511905</td>
</tr>
<tr>
<td></td>
<td></td>
<td>materials</td>
<td>0.507273</td>
</tr>
<tr>
<td></td>
<td></td>
<td>engineering</td>
<td>0.505917</td>
</tr>
<tr>
<td></td>
<td></td>
<td>waste</td>
<td>0.505455</td>
</tr>
<tr>
<td></td>
<td></td>
<td>geology</td>
<td>0.502525</td>
</tr>
<tr>
<td></td>
<td></td>
<td>equipment</td>
<td>0.502212</td>
</tr>
<tr>
<td></td>
<td></td>
<td>socio-economic factors</td>
<td>0.502174</td>
</tr>
<tr>
<td></td>
<td></td>
<td>processes properties</td>
<td>0.502000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>biology</td>
<td>0.500000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>geography</td>
<td>0.500000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>safety</td>
<td>0.500000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>government</td>
<td>0.500000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>transportation</td>
<td>0.500000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>atmosphere</td>
<td>0.500000</td>
</tr>
</tbody>
</table>

The classification algorithm has picked up modeling as the strong concept for the particular document 1972. It is this strong concept which is of main concern to us for the evaluation.
purpose. That is, it can be used to judge the appropriateness of the automatic classification algorithm.

Manual Classification of Documents

In this section the manual classification scheme that is done by the subject experts is described. The two experts worked on the collection of 54 documents and did the indexing job manually. Each document here is analyzed for the document content and is assigned to the predefined concepts based on their knowledge in that subject area. In this way they have classified all the 54 documents under the 16 concepts manually. A typical manual classification is given below:

<table>
<thead>
<tr>
<th>docid:1972</th>
<th>hydrology modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>docid:2047</td>
<td>geology modeling</td>
</tr>
<tr>
<td>docid:2055</td>
<td>modeling waste</td>
</tr>
</tbody>
</table>

3.1.5 Performance Evaluation

In this section, the performance of the automatic concept based document classification system is compared with the manual classification of documents done by the subject experts.
There are generally two parameters that are used in information retrieval to evaluate the performance of any retrieval system. They are *precision* and *recall*. Recall can be defined as the proportion of relevant material retrieved. On the other hand, precision is defined as the proportion of retrieved material that is relevant. Consider for example, if the document collection can be divided into relevant and nonrelevant items as shown in Figure 3.1 then the recall (R) and precision (P) are given by:

\[ R = \frac{\text{number of documents retrieved and relevant}}{\text{total relevant in collection}} \]

\[ P = \frac{\text{number of documents retrieved and relevant}}{\text{total retrieved}} \]

In this experiment, recall and precision are calculated for the automatic concept based document classification system by comparing it with the manual classification system. Given below are a few examples how recall and precision are calculated in this case.
Example 1: If for a particular document (docid:1813)
manually chosen concepts are: hydrology and modeling
automatically chosen concepts: hydrology and modeling
then
\[ \text{recall} = 100\% \]
\[ \text{precision} = 100\% \]

Example 2: If for a particular document (docid:1834)
manually chosen concepts are: hydrology and modeling
automatically chosen concept: hydrology
then
\[ \text{recall} = 50\% \]
\[ \text{precision} = 100\% \]

Example 3: If for a particular document (docid:1841)
manually chosen concepts are: geology and modeling
automatically chosen concepts: biology, geology, hydrology and modeling
then
\[ \text{recall} = 100\% \]
\[ \text{precision} = 50\% \]

Example 4: If for a particular document (docid:1853)
manually chosen concepts are: engineering and safety

automatically chosen concepts: safety and waste

then

\[ \text{recall} = 50\% \]

\[ \text{precision} = 50\% \]

The precision and recall for all the 54 documents in the collection are calculated and the average of these values is computed. The threshold value used to binarize the document vector has an important impact on the values of precision and recall. It was found that the threshold value had an effect on the similarity values generated between concept and document vectors. These similarity values produced affected the clustering process and hence varying values of precision and recall are generated. For different values of threshold, the values of (precision, recall) varied from (80%, 30%) to (30%, 95%). Finally after finding a perfect balance between these values the results obtained are given below:

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>68%</td>
<td>68%</td>
</tr>
</tbody>
</table>

Validation using Graph Theoretical Techniques

Graph Theoretical technique described in Chapter 2 has been used to check for the validity of the results produced by the automatic concept based document classification system. For each concept, the list of all documents having that concept as the strong concept is built. In other words, for each strong concept document clusters are built. Now,
using the original document vectors, document clusters are generated by applying graph theoretical clustering technique. In particular, the algorithm developed by Minker given in [1] is used to cluster documents. The document clusters of strong concepts and those generated by graph theoretical techniques are compared. It has been found that almost in all cases, that is, all concepts, the clusters generated were found to be the same. Thus, it can be stated that the experiment of automatic concept based document classification is meaningful.
Chapter 4

Conclusions

4.1 Conclusions and Future Work

This thesis mainly deals with the automatic concept based document classification. An experimental system is built that does this classification automatically. In summary, document vectors are generated for the 54 documents in the collection by processing them serially, indexing them and stemming them by using the stemming algorithm. The stemmed words are then assigned weights by using the collection weighting algorithm. The document vectors are then binarized by using the sum of all non-zero weights as the threshold. That is, anything above the threshold is assigned a value of 1 and anything below it is assigned a value of 0. A total of 54 document vectors are generated using the above process. A list of 16 concept files is built by pulling out the terms under the CA field of the LSS Thesaurus and grouping them under each concept. The concept vectors are generated by using a dictionary look-up procedure. That is, the terms under each concept are looked for in the dictionary of terms and their presence or absence is indicated by a 1 or 0 in the vector. The similarity coefficient between the document and concept vectors is found. The similarity
values are then sorted in a descending order and then for each document the 16 concepts are grouped into strong, related and weak groups in the classification phase. The performance of this automatic classification system is measured against the manual classification done by the subject experts and the values of precision and recall are calculated. It has been found that the values of both precision and recall are 68%. These values are fairly good for an ideal automatic classification system.

The existing automatic concept based document classification system can be further improved by using the techniques of natural language processing and by building knowledge bases in particular subject areas. A knowledge base can specify the relation between the entities and can also contain a set of inference rules designed to extend the available knowledge by supplying new facts and relations from already available information. By incorporating learning into this system an expert model can be built which may further improve the values of precision and recall. Also if a way can be found to assign weights to the terms in a concept vector then again the values of precision and recall would increase. This is left as a future work.
Bibliography


