

CONCEPT BASED RETRIEVAL AND INFORMATION FILTERING

by

Ali H. Alsaffar



A DISSERTATION

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfillment of Requirements

For the Degree of Doctor of Philosophy

Major: Computer Science

Under the Supervision of Professor Jitender S. Deogun

Lincoln, Nebraska

January, 2001

DISSERTATION TITLE

Concept Based Retrieval and Information Filtering

BY

Ali H. Alsaffar

SUPERVISORY COMMITTEE:

APPROVED

DATE

Signature

Dr. Jitender S. Deogun

Typed Name

1/15/2001

Signature

Dr. Sharad Seth

Typed Name

1/16/2001

Signature

Dr. Ashok Samal

Typed Name

1/16/2001

Signature

Dr. Stephen Goddard

Typed Name

1/16/2001

Signature

Dr. Ram Narayanan

Typed Name

1/16/2001

Signature

Typed Name

CONCEPT BASED RETRIEVAL AND INFORMATION FILTERING

Ali H. Alsaffar, Ph.D.
University of Nebraska, 2001

Advisor: Jitender S. Deogun

Information filtering has become an important component of modern information systems due to significant increase in its applications. The major goals of a successful Information Filtering System include: to close the representation gap between documents and user profiles, *and* to classify/categorize documents efficiently as they arrive into the system. In this thesis, we develop a conceptual query model that is close to the user's need and investigate an information filtering method for text categorization/classification.

The central idea of our conceptual model is captured in a rule based query model. The proposed approach involves preprocessing of the rule base to generate Minimal Term Sets (MTSs) that speed up the retrieval process. Furthermore, we extend our model into two directions when document terms are non-binary. First, we incorporate the p -Norm model into the process of evaluating MTSs. Second, we adopt the Generalized Vector Space model (GVSM) in which the term-term association is well established.

For text classification/categorization, we investigate a steepest descent induction

algorithm combined with a two-level preference relation on user ranking. The performance of the proposed algorithm is evaluated experimentally. The experiments are conducted using Reuters-21578 data collection. Finally, we demonstrate effectiveness of proposed method by comparing experimental results to other inductive methods.

ACKNOWLEDGMENTS

I would like to dedicate my gratitude to many individuals who have made their contributions, directly or indirectly, to this thesis. I would like to express my sincerest thanks and appreciations to my advisor, Professor Dr. Jitender Singh Deogun, who has contributed very generously of his time, expertise and guidance not only through my academic career but also through my life as a personal friend. I will be indebted to him for his help and understanding. Special thanks dedicated to Dr. Hayri Sever from Hacettepe University, Turkey. He was a visiting faculty member at the Department of Computer Science and Engineering at the University of Nebraska-Lincoln in the academic year 1998/1999. I learned extensively from working closely with him. His guidance and support helped me tremendously in my dissertation. I am indebted to him.

I am grateful to the members of my reading committee, Dr. Ashok Samal and Dr. Stephen Goddard for very careful reading of the manuscript and improving the readability of this thesis. Many thanks are also go to Professor Dr. Sharad Seth and Professor Dr. Ram Narayanan for serving on my Ph.D. supervisory committee.

I wish to express my gratitude to the University of Bahrain for scholarship and the support provided through the duration of this project. Also, thanks to the Department of Computer Science & Engineering and the University of Nebraska-Lincoln for giving me the opportunity to study for the Ph.D. degree and also for the teaching

assistantship.

My sincere appreciations to my parents for their inspiration, understanding, and love that bestowed on me. My success has been assured by their prayers and support throughout my life. I am very grateful to my uncle Sayed Hasan Al-Noaimi, to my brothers Dr. Saeed, Jameel, Zuhair, Kareem, Mohammed, Fadhel, and Dr. Younis. I am thankful to all my friends especially Dr. Aqeel Ahmed, Dr. Mohammed Al-Sammak and his wife Dr. Suad Traif for their support. Finally, I want to convey my heartiest thanks to my wife, Sana Al-Noaimi, for her unconditional love, for her sacrifices, for her patience, and for her prayers.

Contents

Abstract	i
Acknowledgments	iii
1 Introduction	1
1.1 Information Filtering, Information Retrieval, and Classification	2
1.2 Problem Formulation	4
1.3 Objectives of Dissertation Research	10
1.4 Research Contributions	14
1.5 Dissertation Organization	16
2 Literature Survey	17
3 Minimal Term Sets	24
3.1 Introduction	25
3.2 AND/OR Goal Tree	29
3.3 Minimal Term Sets	36
3.3.1 Binary Weights and Simple Retrieval Functions	37

3.3.2	Binary Weights and Generalized Retrieval Functions	41
3.3.3	Sharing Of Rule Bases	44
3.4	Conclusion	45
4	Retrieval with Extended Boolean Models and Document Term	
	Weights	46
4.1	Introduction	46
4.2	Incorporating Document Term Weights into AND/OR Goal Tree . . .	49
4.3	Incorporating Document Term Weights into MTS Model	55
4.4	Conclusion	58
5	On Modeling of Concept Based Retrieval in Generalized Vector	
	Spaces	59
5.1	Introduction	60
5.2	Precision/Recall Effectiveness Measures	61
5.3	Review of Rule-base Trees and MTSs	64
5.4	Review of GVSM	69
5.5	Modeling RUBRIC concepts with GVSM	74
5.5.1	Mapping MTSs to Query Vectors in GVSM	75
5.5.2	A Complete Example	78
5.6	Experiment Design and Results	81
5.7	Conclusion	83

6	Optimal Queries in Information Filtering	85
6.1	Introduction	86
6.2	Preliminaries	87
6.3	Effectiveness Measures	93
6.4	The Experiment	95
6.4.1	Reuters-21578 Data Set and Text Representation	95
6.4.2	Training	98
6.4.3	Results	102
6.5	Conclusion	105
7	Summary and Future Work	107
7.1	Concluding Remarks	107
7.2	Future Work	109
	Bibliography	111

Chapter 1

Introduction

With the rapid growth of the internet and related technologies, tremendous efforts have been focused on techniques and methodologies to handle and process information efficiently. For example, *electronic commerce* (EC), which is a rapidly growing area related to the Internet, must be supported by online capabilities, such as secure transactions, practical payment methods, and intelligent search engines.

Information environments to support applications like EC are usually large, open, and heterogeneous. Recent approaches have introduced intelligent agents to facilitate working with these environments [20]. The intelligent agents are integral parts of online environments that provide mechanisms for advertising, searching, presenting, managing, and updating information. These agents are new software technologies that act as user surrogates for automating many challenges on the World Wide Web [24]. Basically, agents act on behalf of the users in order to carry out routine tasks. Some of these tasks are locating and accessing information from various on-line information sources, resolving inconsistencies in the retrieved information, scanning newspaper sites for articles relating to a company's business, collecting information about a

given stock, scanning and filtering, and searching online news related to a company's clients and competitors.

Hypertext and Information Retrieval (IR) systems are two different types of information systems developed in response to different aspects of information needs of users. Browsing structured text is, for example, easy in hypertext environments, while IR systems generally ignore such structure and view a document as an atomic entity. Thus, the two systems differ in representing, organizing and retrieving documents, and expressing user needs. On one hand, documents in hypertext systems are described by markup languages like SGML or HTML (or XML, more recently). On the other hand, in IR systems flat (or unstructured) documents are indexed and retrieved in their entirety.

1.1 Information Filtering, Information Retrieval, and Classification

Information Filtering (IF) is akin to Information Retrieval (IR) as well as Classification. Our hypothesis is that IF is in fact a type of classification process that has some similarities and differences with IR.

Information filtering and Information Retrieval systems. *Information Seeking* is a term that describes any process by which users can obtain information from automated information systems. Two major types of information seeking is IR and IF. IF differs from IR in the sense that the former monitors documents as they enter the system (newly arrived documents) and selects the ones that match the user

query (known as a *profile* in Selective Dissemination of Information [23]). Information retrieval systems, on the other hand, accepts user needs (requests) as a query and provide the user with a list of documents ordered (ranked) based on the similarity to the user query. Thus, IF seeks relevant/non-relevant decision rather than a ranked output as in IR [8].

Information Filtering Systems share many of the features of Information Retrieval. However, there are some differences. The major differences between IF and IR are [8]:

- (1) IF involves selection or rejection of incoming data (dynamic datastream), either being broadcast by remote sources such as newswire services, or sent directly by other sources such as e-mails, while IR is concerned with the selection of text from a relatively static database.
- (2) Filtering involves the distribution of the filtered text to groups or individuals while IR is concerned with the collection and organization of text.
- (3) Where filtering is typically concerned with repeated uses of the system with a long-term goal or interest, IR is typically concerned with single uses of the system with a one-time goal or interest.

Information Filtering and Classification. Since an IF system makes decision based on relevance and non-relevance of documents it can be considered as a classifier. Classification is the assignment of documents to relevant and non-relevant classes based on similarity between the document and the user query. However, an IF system

is dynamic in the sense that user profiles are updated periodically and, consequently, classification is performed online. Classification has been widely studied in computer science, statistics, statistical pattern recognition [16, 43], and presents a useful tool in IR [25, 28, 41].

There are a variety of text-based information systems other than IF and IR. Text Routing [41] involves sending relevant incoming data to individuals or groups (i.e. documents are to be assigned to one of two categories, relevant and non-relevant.) This process is identical to filtering. Categorization systems [7, 25, 26, 28] are designed to attach one or more predefined categories to incoming objects. In text categorization systems, the categories are static as compared to filtering profiles which are dynamic. Extraction systems [15] emphasize the extraction of facts or common features from the text objects. Selective Dissemination of Information (SDI) [23] is identical to most information filtering applications.

1.2 Problem Formulation

Information Filtering and Information Retrieval are generally divided into three sub-tasks: *Collecting* is the process of gathering information from producers; *Selecting* is the process of handling the filtering or retrieval operations; and *Displaying* is the process of presenting the results to the user. Figure 1.1 depicts the basic framework that is common to IF and IR. Our current focus is on information filtering, therefore, we want to review the information filtering model in detail as shown in Figure 1.2. It

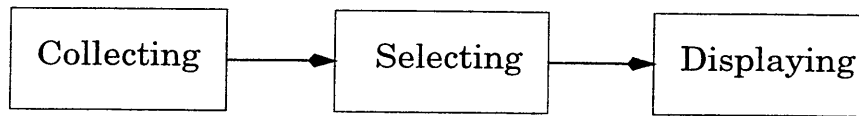


Figure 1.1: Basic framework for IR and IF.

may be noted that the success of the filtering process depends on the representation of documents and profiles. Consequently, the goal of a filtering system is to close the representation gap between documents and profiles.

Concept based retrieval: One of the major problems in searching on the Web is that commercial search engines do not honor user preferences properly. One of the reasons for this is that there still exist a considerable gap between the terminologies used in defining queries and that used in document representation. This is due to the fact that Web catalogs for individual vendors employ different structures, names, formats, and abstractions for products. A number of approaches have been developed in an effort to close this gap, namely the extension of queries by use of a thesaurus [37], inference models that cluster either index terms or documents [12, 35, 36, 44], or feedback techniques as described in the next chapter [38].

Our objective in this thesis is to enhance query representation by developing a conceptual query model that is close to the users' need and is efficient for query evaluation. This leads us to a model of IR that combines concept level retrieval with descriptor level retrieval. Therefore, we consider a *concept-level* retrieval model on top of *descriptive-level* retrieval. Retrieval of documents based on keywords and descriptor matching is called descriptor-level retrieval and retrieval of documents

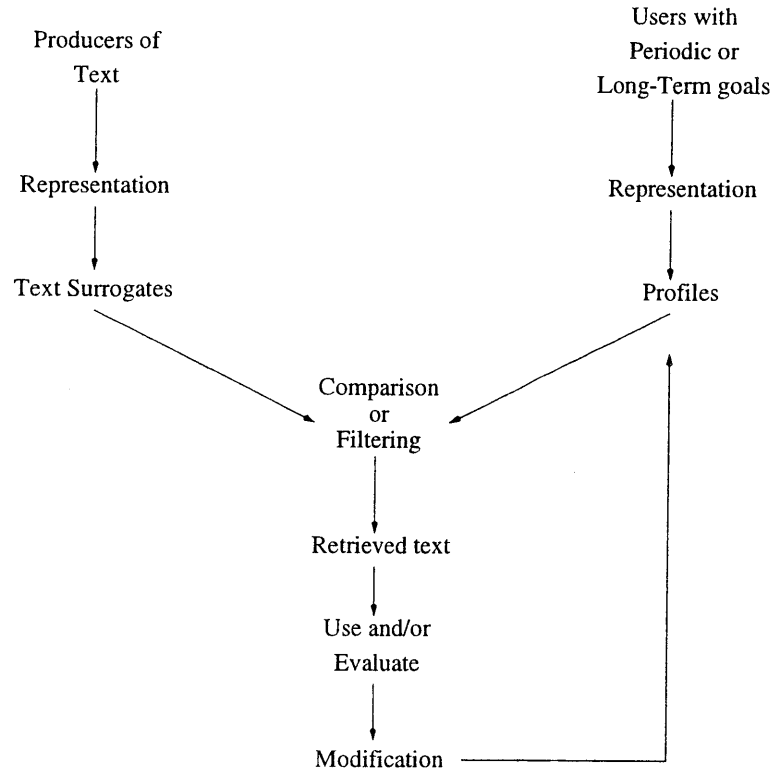


Figure 1.2: Information Filtering Model.

based on query topics is called concept-level retrieval. We adopt the *Rule Based Retrieval of Information by Computer* (RUBRIC) model [29] on top of the descriptive-level retrieval. Query topics are specified by rules and the output is obtained by evaluating an AND/OR Goal Tree of these rules. (Figure 1.3 shows the rules for the query topic Desktop, which will be explained in detail in Chapter 3). An AND/OR Goal tree is a tree that is composed of inner nodes, representing query topics of these rules, and outer nodes (leaves), representing text references of these rules. A weight is also assigned on each edge between nodes. These weights are obtained from the rules. For an example of an AND/OR Goal tree, see Figure 5.2 in Chapter 5.

```

PROCESSOR & RAM & CACHE & HARD_DRIVE & VIDEO => DESKTOP
(PENTIUM | AMD) & "MMX"                      => PROCESSOR
["P2/233",0.8] | "P2/266"                     => PENTIUM
["K6/233",0.6] | ["K6/266", 0.9]              => AMD
["32MB",0.5] | "64MB"                         => RAM
"512KB" | "Cache"                             => CACHE
"6.4GB" & "EIDE"                              => HARD_DRIVE
("ATI" | ["AGP",0.7]) & "4MB/3D"              => VIDEO

```

Figure 1.3: A Rule Base for a Desktop Concept.

The AND/OR Goal Tree algorithm (presented in Chapter 3) generates a Prolog-like code for a given query topic. The input of the algorithm is a set of facts corresponding to the inverted index file and a set of rules for query topics. The code generated is used for evaluation such that a list of documents along with their Retrieval Status Values (RSV) is obtained as the retrieval output. RSV of a document is defined as a quantitative value reflecting the degree of similarity between a document and a query. We show that the time complexity of AND/OR evaluation is exponential in the number of the conjunctions in the DNF expression associated with the query topic [4].

Information Filtering Process: In information retrieval, the linear separation refers to an acceptable retrieval, where all relevant documents are ranked higher than all non-relevant documents, and the retrieval function that provides an acceptable retrieval is called a *linear discriminant function*. If the retrieval output returned by the information system in response to a user query is not satisfactory to the user, the system may launch an interactive session in which the current retrieval output

is used as a training set. The user judges some of the documents as either relevant or non-relevant; the system constructs a new query using the current query and the user's judgments and a new retrieval output is brought to the user's attention. This interactive session continues until an acceptable retrieval is obtained; The resulting query is called an *optimal query*.

In text categorization, we have neither a retrieval output nor a user query. Instead, we have a number of categories, and the document collection for each category is partitioned into training and test cases. Now, the classification problem becomes a process of obtaining, for each category, an optimal query from the training set and evaluating this query on the test set.

Exploiting the historical knowledge generated by optimal queries reduces response time of the system to a user query or the time required to obtain an optimal query. Zhang et al [52] presented an object-oriented modeling of the history of optimal queries, where the retrieval scheme assumed a concept hierarchy of documents and a rule subsystem. Raghavan and Sever [32] conjectured experimentally that the process of query formulation converges faster when past optimal queries are reused as initial queries. They proposed a *query base* to store the history of such optimal global knowledge.

A *preference relation* may be defined as a binary relation that determines which document is more useful to a user based on his/her reasoning of preference [51]. Let $\succeq$ be a preference relation defined on a set of documents, Δ . For any documents

$D, D' \in \Delta, D \succeq D'$ is interpreted as: D is preferred (or considered equally good) to D' by the user. It is easy to see that the user preference yields a weak order, such that, for all $D, D', D'' \in \Delta$ the following hold:

I. $D \succeq D'$ or $D' \succeq D$, and

II. $D \succeq D'$ and $D' \succeq D'' \implies D \succeq D''$

If $D \succeq D'$ and $D' \succeq D$ hold we write $D \sim D'$, which means D and D' are equally good. If $D \succeq D'$ and $D' \sim D$ does not hold then we write $D \succ D'$ which means D is strictly better than D' .

In this thesis we concentrate on the preference model for information filtering. An optimal query is developed by formulating and optimizing based on a two-level classification scheme. This scheme assumes that relevant documents are preferred over non-relevant documents. Hence,

$$D \succ D' \iff (D \in rel, D' \in nrel)$$

$$D \sim D' \iff (D, D' \in rel) \text{ or } (D, D' \in nrel).$$

After such relation is obtained between each pair of relevant and non-relevant documents, the goal is to compute an optimal query that produces a ranking output such that relevant documents are ranked higher than non-relevant.

1.3 Objectives of Dissertation Research

We propose to develop a methodology for designing information filtering systems.

The proposed framework consists of two building blocks:

- I. Conceptual Retrieval, and
- II. Text categorization.

The proposed conceptual retrieval framework is developed on top of a descriptive retrieval model. The text categorization includes a framework for the information filtering that utilizes the preference relation discussed in Section 1.2.

We introduce the concept of a *Minimal Term Set* (MTS) in the context of RUBRIC system which is also applicable to any rule-based system. MTS will replace the Prolog-like code produced by the AND/OR algorithm for rule evaluations (AND/OR algorithm is presented in Chapter 3). We show in Chapter 3 that the time complexity of evaluating rules based on MTS model is polynomial. The objective is to develop an algorithm that generates MTSs from rules presented in Disjunctive Normal Form (DNF). Rules that are represented in DNF are stratified based on the dependency relation. If a query topic QT_1 depends on another query topic QT_2 , then QT_2 is placed in a stratum lower than QT_1 . As a result, we apply forward chaining in evaluating the rule set such that the rules in a lower stratum are evaluated before the rules in a higher stratum. Figure 1.4 shows the DNF of the rules shown in Figure 1.3. So, our overall objectives for the conceptual framework are summarized as:

```

Stratum 0:
  AMD      = {{["K6/233", 0.6]}, {"K6/266", 0.9]}}
  CACHE    = {{["512KB", 1]}, {"Cache", 1]}}
  HARD_DRIVE = {{["6.4GB", 1], ["EIDE", 1]}}
  PENTIUM   = {{["P2/233", 0.8]}, {"P2/266", 1]}}
  RAM       = {{["32MB", 0.5]}, {"64MB", 1]}}
  VIDEO     = {{["ATI", 1], ["4MB/3D", 1]},
               {"AGP", 0.7], ["4MB/3D", 1]}}

Stratum 1:
  PROCESSOR = {{[PENTIUM, 1], ["MMX", 1]},
               [AMD, 1], ["MMX", 1]}}

Stratum 2:
  DESKTOP   = {{[PROCESSOR, 1], [RAM, 1], [CACHE, 1],
               [HARD_DRIVE, 1], [VIDEO, 1]}}

```

Figure 1.4: Disjunctive Normal Forms of Rules – defined in Figure 1.3

1. Develop an algorithm that generates MTSs from rules presented in Disjunctive Normal Form (DNF.)
2. Show that both AND/OR goal tree and MTSs evaluation result in the same retrieval output if term weights are binary.
3. Show that MTSs evaluation is better than AND/OR Goal tree.
4. Develop an algorithm that retrieves documents based on MTS model in polynomial time. It is assumed that term weights are Boolean and documents are represented by vectors.

For the information retrieval system, an optimal user query is constructed and stored in the user profile. The system monitors documents arriving at any time, if a match is found between a document and a user profile this document is saved

and sent to the user. The user inspects the documents and marks them as relevant or non-relevant (two-level of preference). A newly created query is, then, obtained from the current query that discriminates the preferred documents from the others. This process is continued until the query is no longer needed (i.e. the user profile is removed from the system.)

For query formulation, we adopt the *Steepest Descent Algorithm* (SDA). The reason for choosing this algorithm is three fold. First, it was shown theoretically in [47] that Rocchios' method for relevance feedback in vector space model was only a first-order approximation and was not necessarily a solution vector. In this case, a higher-order approximation relevance feedback as compared to Rocchios' relevance feedback is needed. Second, the SDA approach has not been experimentally evaluated in IF context. Therefore, the performance of SDA for IF is unknown. Our purpose is to investigate SDA's merits and/or demerits. Third, it has been established that using past optimal queries as initial queries input to the algorithm results in efficient convergence [32]. Thus, we believe that SDA is a natural choice for query formulation.

We propose an experimental setup using the Reuters-21578 text categorization test collection. This collection is a popular choice for text categorization research and is available publicly¹. The documents of the collection are Reuters newswire stories that are divided into training and test sets. Each document is assigned manually by a human expert to topics that it belongs to.

¹<http://www.research.att.com/~lewis/reuters21578.html>

We summarize the objectives of our experimental evaluation as follows:

1. Generate multi-class classifiers using the steepest descent algorithm for each category in the Reuters-21578 corpus. Obtaining the training set for each category is carried out by the following steps:
 - (a) Pre-process the corpus, performing parsing and tokenizing the text portion of the title and the body of training and unused documents, to produce a dictionary of single words excluding numbers.
 - (b) Use a *stop list* to eliminate functional words from the dictionary.
 - (c) Employ stemming to the words in the dictionary to reduce each remaining word to a word-stem form.
 - (d) Compute the term-to-document frequency matrices for the training and test documents.
 - (e) Compute document vectors for each category. These vectors make up the training set for each category.
2. Evaluate and report the performance of the classifiers. We apply average precision/recall and Breakeven point as effectiveness measures.
3. Analyze the steepest descent algorithm by evaluating the classifiers for different sizes of training set. Empirically, estimate the best training size.
4. Analyze the steepest descent algorithm on different values of normalized recall R_{norm} .
5. Report the effect of negative examples in multi-class classification.

1.4 Research Contributions

Our objective in this thesis is to develop an efficient information filtering system model. The major goals of this model are: closing the representation gap between the documents and the profiles, and filtering (classifying) documents efficiently as they arrive to the system. As a result, an information filtering system is efficient if the above two goals are achieved successfully.

Our first contribution is developing a conceptual retrieval model for the information filtering system. The objective of this model is to close the terminologies used for defining queries and those for representing documents. We propose the *Minimal term Set* (MTS) model, which involves preprocessing of a rule base system. MTS model was first published in a paper submitted to the *Foundation of Intelligent Systems: Eleventh International Symposium on Methodologies for Intelligent System*, ISMIS'99 [4]. In this paper, we adopted RUBRIC (*RUle Based Retrieval of Information by Computer*) model, which was a conceptual-level retrieval that was built on top of a descriptive-level retrieval [29]. We have provided an algorithm for computing the minimal term sets and derived the associated computational complexity. We showed that MTSs evaluation was polynomial and more efficient compared to query evaluation in RUBRIC.

Furthermore, we developed strategies for enhancing retrieval output by incorporating document term weights into the computation of document-query similarity measures for MTS model. Accordingly, the concept of MTS was enhanced and ex-

tended into two directions. In the first direction, we incorporated p -Norm model, which is an extended Boolean model, into MTS. In this model, the parameter p , which is used in the *AND* and *OR* formulas, is set to ∞ when generating an equivalent DNF and set to intermediate value for evaluation. We published this study in the *Journal of Intelligent Information Systems* [6]. In the second direction, we adopted the GVSM, *Generalized Vector Space Model*, in which the term-term association was integrated in the retrieval computation. We defined a formal mapping of MTSs to query vectors in GVSM. This study and its associated experiments were published and presented in the *Foundation of Intelligent Systems: The 12th International Symposium on Methodologies for Intelligent System*, ISMIS'00 [22].

In an attempt to improve the process of information filtering, we proposed a framework for the filtering problem based on *Steepest Descent Algorithm* SDA, which is an induction method, combined with a two-level preference relation on user ranking. The proposed method was experimentally evaluated using Reuters-21578 text categorization test collection and the results were published in a paper submitted to the *Foundation of Intelligent Systems: The 12th International Symposium on Methodologies for Intelligent System*, ISMIS'00 [5]. Our experiment in this paper was compared with five different induction methods that were recently experimented on Reuters-21578. In conclusion, it performed very well.

1.5 Dissertation Organization

The organization of this dissertation is as follows. In Chapter 2, we present a literature survey of concept based retrieval and information filtering and related disciplines. The proposed conceptual model is presented in Chapter 3 where the notion of Minimal Term Sets is introduced. The algorithms for generating and evaluating MTSs are also presented in Chapter 3. In Chapters 4 and 5, the concept of MTS is extended to incorporate advanced models. In Chapter 4, MTS model is extended by incorporating p -Norm model into the evaluation of MTSs to handle the case where the weights of both documents and query terms are non-binary. In Chapter 5, we extend MTS model by adopting Generalized Vector Space Model (GVSM) in which term-term association is well established. In Chapter 6, an Information Filtering method based on steepest descent algorithm is investigated, and the performance of the proposed algorithm is experimentally evaluated. Further, we demonstrate the effectiveness of SDA by comparing experimental results to other inductive methods. Finally, concluding remarks are presented in Chapter 7. The work in this thesis is presented in [4, 5, 6, 22].

Chapter 2

Literature Survey

Information Retrieval has been studied extensively for more than three decades. Useful surveys of the IR literature can be found in [18, 23, 36]. These books present a good discussions on topics like capabilities of IR, cataloging and indexing, data structures, clustering, user search techniques, text search techniques, and evaluation. The major distinctions between IR and IF are discussed in [8]. IR and IF have been described as two sides of the same coin to emphasize that Information Filtering shares many properties with Information Retrieval.

It may be noted that most of the research relevant to IR is also relevant to IF. For example, Yan and Molina [50] applied indexing under the Vector Space Model (VSM) [36] to information filtering. Documents and user profiles were represented as vectors of dimension m , where m is the total number of terms used to identify content. Terms were identified after stemming and deleting words that belong to a *stop list*. Term weights were computed by an *inverse document frequency* formula and similarity between profiles and documents were obtained by a *cosine similarity measure* [36]. Given a profile P and a relevance threshold θ , a document D is relevant

to P if $\text{sim}(D, P) > \theta$, where sim is the similarity measure. Similarly, Callan [11] has developed a filtering system named InRoute from parts of the INQUERY retrospective retrieval system. InRoute was based on the inference network model [8] of information retrieval.

Relevance feedback is a technique to enhance user queries based on a user's judgment of documents as relevant or non-relevant. The current query is modified by increasing the weight of terms in relevant documents and decreasing the weight of terms that are in non-relevant documents. Pioneering work on relevance feedback was conducted by Rocchio who advanced the relevance feedback formula known as the *Rocchio formula*. Later, two relevance feedback methodologies were presented in [38]. The first, was relevance feedback based on the vector space retrieval model, and the second, was relevance feedback based on the probabilistic retrieval model. In vector space feedback, three methods were identified, namely, *Ide dec-hi*, *Ide regular*, and *Standard Rocchio*. Let a document D and a query Q be represented in t -dimensions as $D = (d_1, \dots, d_t)$ and $Q = (q_1, \dots, q_t)$, where d_i and q_i represent the i^{th} term in D and Q respectively. Also, let r be the number of retrieved relevant documents and n be the number of retrieved non-relevant documents. Then the formulas for the three methods are:

1. Ide dec-hi

All identified relevant documents but only one retrieved non-relevant document D_k are used for feedback. D_k is selected as the top-most non-relevant document

retrieved.

$$Q_{new} = Q_{old} + \sum_{i=1}^r D_i - D_k \quad (2.1)$$

2. Ide regular

All previously retrieved relevant and non-relevant documents are used for feedback.

$$Q_{new} = Q_{old} + \sum_{i=1}^r D_i - \sum_{i=1}^n D_i \quad (2.2)$$

3. Standard Rocchio

It is similar to Ide regular except that the terms in previously retrieved relevant and non-relevant documents are adjusted by two constants β and γ respectively.

These constants are in the range 0 to 1 so that $\beta + \gamma = 1$.

$$Q_{new} = Q_{old} + \beta \sum_{i=1}^r D_i - \gamma \sum_{i=1}^n D_i \quad (2.3)$$

On the one hand, Hull [21] represented the initial query for the routing problem for a given set of relevant and non-relevant documents by the mean of only relevant documents without including any non-relevant documents in the computation. He discarded non-relevant documents in the computation claiming that no improvements have been observed in the collection that was used, which was a reason that it may not be applicable to other collections.

On the other hand, Salton and Buckley [38] (as well as Wong and Yao [47]) represented the initial query in the application of relevance feed back by the mean of

both the relevant and non-relevant documents. They claimed that the existence of non-relevant documents in the feedback process produced more desirable results than considering none of them. They experimented on 72 different relevance feedback methods and their evaluation of these experiments was accompanied by a ranking order. The best method exhibiting the highest recall-precision was designated a rank order 1, and the worst process was designated a rank of 72. Surprisingly, the best overall relevance feedback method, with an average rank of 2.2 for five collections, was the “Ide dec hi” method. The method that came next, with an average rank of 9.2, was the “Ide regular.” Finally, the third method was the “Rocchio” method, with an average rank order of 16.

An alternative feedback technique is the *Incremental Relevance Feedback* [1]. In this technique the initial result of a query is a single document that is supposedly most relevant to the query. This document is presented to the user in an interactive session and the user indicates whether it is relevant or not. The user feedback is then used for query reformulation in order to compute the next supposedly most relevant document. This interactive session continues until the query is adjusted towards relevant documents. A similar study is conducted by Allan [3] in information filtering, where the document collection is not static and the setting is non-interactive. The incremental feedback is experimented on the TREC routing task by breaking the documents into training and test sets. The queries are improved based upon the training set and the results are checked against the test set.

In IR/IF, *feature selection* is the problem of reducing the number of features that are necessary and sufficient to describe target concept(s). Feature selection is also known as Dimensionality Reduction. A subset of the features are selected that represent the most important features in the feature space that is used in the learning process.

Schutz et. al [41] used the high score of the χ^2 measure to select a set of existing features to use. The test on the most important features was applied to a table (contingency table) that contained the number of relevant N_{r+} and non-relevant N_{n+} documents in which the term occurs, and the number of relevant N_{r-} and non-relevant N_{n-} documents in which the term does not occur as follows:

$$\chi^2 = \frac{N(N_{r+}N_{n-} - N_{r-}N_{n+})^2}{(N_{r+} + N_{r-})(N_{n+} + N_{n-})(N_{r+} + N_{n+})(N_{r-} + N_{n-})}$$

Another method used for dimensionality reduction is creating a new much smaller set of features that original features are mapped into. This method is adopted by [14] and is known as Latent Semantic Indexing (LSI). The process is to describe terms and documents as linear combinations of orthogonal (uncorrelated) indexing variables. This description can be obtained as a matrix constructed from the Singular Value Decomposition (SVD) of the term-document matrix. Let X be the term-document frequency matrix then it is decomposed by SVD into $X=TSD'$, where T and D have orthogonal columns and S is diagonal, and all elements of S (feature factors) are positive and ordered in decreasing magnitude. Therefore, the least important factors can easily be removed by truncating the matrices T , D , and S .

Classification is the process of making decision on class assignment to input patterns of cases. The decision is usually generated from training patterns by means of *machine learning* algorithms. Data for each case consists of a pattern of observations (features) and the corresponding correct classification. The objective of the learning algorithm is to customize the classifier by finding a general way of relating any particular pattern of observations to one of the specified classes.

In pattern recognition principles [16], two techniques are known, namely, *supervised* and *unsupervised* learning. In most cases, representative patterns from each class under consideration are available. In these situations, *supervised* pattern recognition techniques are applicable. In a supervised learning environment, the system is “taught” to recognize patterns by means of various adaptive schemes. The essentials of this approach are a set of *training* patterns of known classification and the implementation of an appropriate learning procedure.

In some applications, only a set of training patterns of unknown classification may be available. In these situations, *unsupervised* pattern recognition techniques are applicable. As mentioned above, supervised pattern recognition is characterized by the fact that the correct classification of every training pattern is known. In the unsupervised case, however, one is faced with the problem of actually learning the pattern classes present in the given data. This problem is also known as *learning without a teacher*.

The optimal classification rule is the Bayes’ rule [16]. Let the feature vector $\mathbf{x}$ be

a d -component and C_i is class i (or category i). Let $P(C_i)$ be the priori probability of class C_i and $P(\mathbf{x}|C_i)$ is the conditional probability of $\mathbf{x}$ given it is in class C_i . Then decide C_i if $\mathbf{g}_i(\mathbf{x}) > \mathbf{g}_j(\mathbf{x})$ for all $j \neq i$, where $\mathbf{g}_i(\mathbf{x}) = P(\mathbf{x}|C_i)P(C_i)$. Although the Bayes' rule is the optimal classification rule, its direct use is too difficult to comprehend. Furthermore, the rule has been used for classification when measurements come from a normal distribution.

Classification by learning is widely used in machine learning. Good examples can be found in books by Mitchell [30] and Weiss et al [45]. Some of these examples are, Decision Trees, Artificial Neural Networks, Bayesian learning, k -Nearest Neighbor, inductive rules, and Genetic Algorithms. Classification by learning is also used in Information Systems for text classification. Lewis [26] used a probabilistic approach for evaluation of the effect of phrasal and clustered representations on text categorization. Schütz et al [41] presented and compared three different classification algorithms; linear discriminant analysis, logistic regression, and neural networks. The three classifiers were tested using the Tipster collection and the TREC-2 and TREC-3 routing tasks. Finally, Apté et al [7] validated their text categorization algorithm based on decision rules using the Reuters-22173 data collection.

Chapter 3

Minimal Term Sets

There is considerable interest in bridging the terminological gap that exists between the way users prefer to specify their information needs and the way queries are expressed in terms of keywords or text expressions that occur in documents. One of the approaches proposed for bridging this gap is based on technologies for expert systems. The central idea of such an approach was introduced in the context of a system called Rule Based Retrieval of Information by Computer (RUBRIC) [29]. In RUBRIC, user query topics (or concepts) are captured in a rule base represented by an AND/OR tree. The evaluation of the AND/OR tree is essentially based on minimum and maximum weights of query terms for conjunctions and disjunctions, respectively. The time to generate the retrieval output of an AND/OR tree for a given query topic is exponential in m , where m is the maximum number of conjunctions in the DNF expression associated with the query topic. In this chapter, we propose a new approach for computing the retrieval output. The proposed approach involves preprocessing of the rule base to generate Minimal Term Sets (MTSs) that speed up the retrieval process. The computational complexity of the on-line query evaluation following the

preprocessing is polynomial in m . We show that the computation of MTSs allows a user to choose query topics that best suit their needs and to use retrieval functions that yield a more refined and controlled retrieval output than is possible with the AND/OR tree when document terms are binary.

3.1 Introduction

One of the core issues in retrieval systems, as well as search engines on the Internet, is to address user preferences properly. Some retrieval engines provide advanced query interfaces to approximate user preferences by allowing weighted terms and flexibility in the specification of display format for the retrieval output. There is, however, still a considerable gap between the terminologies used for defining queries and those for representing documents. There are a number of approaches available for bridging the terminological gap between users and authors of indexed documents, namely the extension of queries by the use of thesaurus [37], the use of inference models that cluster either index terms or documents [12, 35, 36, 44], or the application of feedback techniques that may be used in conjunction with the previous two approaches [38]. We adopt the RUBRIC (RUle Based Retrieval of Information by Computer) model on top of a descriptor-level retrieval such that the proposed retrieval engine supports concept based retrieval with only a moderate increase of complexity in defining user queries as well as in responding to these queries [29].

RUBRIC is really a preference management system rather than an expert system.

The system's knowledge base, in an expert system, defines what is known about some field of inquiry (e.g., infectious diseases, or geology) in a readily usable form analogous to that used by human experts. Although the knowledge is never complete and perhaps not agreed upon by all experts, there exists some underlying theory or physical model that everyone concerned believes in. In the case of information retrieval, as in other areas of personal preference, such as politics or matters of style, there is no definite "right" answer. Accordingly, RUBRIC is built for capturing and evaluating human preferences. This is the main reason for adopting RUBRIC in our architecture. Preference management systems are likely to play a significant role in retrieval systems of the future, as artificial intelligence research leads to better ways of tackling the problem of complex, multi-attribute decision making.

Our design philosophy is based on the premise that conceptual-level retrieval can be built on top of and is considered complimentary to descriptive-level retrieval in which a user may express his needs through a query topic (or a concept). We use rules involving combinations of descriptors or text references to specify a query topic. A rule is simply a definition of a query topic (or a decision concept) in terms of its components; that is, subqueries (or subconcepts) that are joined by a logical 'AND' operator. The logical 'OR' operator is implicitly employed in evaluating alternate definitions of the same topic. In other words, the expression on the left hand side of a rule is called a pattern in the sense that the existence of a pattern implies existence of a decision concept within a document. As an example, a set of rules for the Desktop

```

PROCESSOR & RAM & CACHE & HARD_DRIVE & VIDEO => DESKTOP
(PENTIUM | AMD) & "MMX"                      => PROCESSOR
["P2/233",0.8] | "P2/266"                     => PENTIUM
["K6/233",0.6] | ["K6/266", 0.9]              => AMD
["32MB",0.5] | "64MB"                        => RAM
"512KB" | "Cache"                            => CACHE
"6.4GB" & "EIDE"                             => HARD_DRIVE
("ATI" | ["AGP",0.7]) & ["4MB/3D",0.9]      => VIDEO

```

Figure 3.1: A Rule Base for a Desktop Concept.

query is shown in Figure 3.1. In the informal syntactical representation of a rule, the derived terms (or subtopics) are represented in capital letters and explicit terms (or text references) thought to be present in documents are represented in double quotation. Both “AND” and “OR” operators have the same priority level and the association is from left to right. These rules can be suppressed by using parentheses. The weight assignment of query topics and explicit terms is explained in Section 3.2.

A set of rules for a query topic constitutes a goal tree involving AND/OR arcs. Furthermore, the calculus to combine weights is to use minimum and maximum for AND and OR arcs, and product for propagating the weights via implication nodes.

Incorporating production rules in expressing the information needs of users preserves the query structure inherent in a Boolean retrieval model; that is, the conjunction and disjunction of query topics correspond to term phrases and term synonyms. At the same time, the rule-based retrieval model closely resembles the fuzzy-set retrieval model by ranking documents with respect to the lowest or highest weights of query terms for conjunction and disjunction, respectively.

The concept-level module can be implemented on top of a standard descriptor-level retrieval engine. The standard approach for a concept-level module to interact with the retrieval engine is to send queries that return, for each document, the subset of the explicit terms in the leaf nodes appearing in the document. This information is then used to evaluate the document against the rule base. Since this approach, which we call AND/OR goal tree generation/evaluation, is slow, we propose an alternative approach for interaction between a concept-level module and the retrieval engine. In our approach, a rule base for a query topic is converted to a disjunctive normal form (DNF) made up of conjunctions of explicit terms, called minimal term set (MTS) [4]. Minimal here has the usual meaning. If some terms in an MTS do not appear in a document then the retrieval status value (RSV) of the document changes (reduces) to some other value; if additional terms appear in the document (than what is in an MTS), then RSV is not affected. Such conversion of the desktop concept is shown in Figure 3.2. Each conjunction in MTS query is treated as an independent query and submitted to the retrieval system. We believe that the availability of MTSs allows a user to choose query topics that best suit their needs from an existing set of query topics

This chapter is organized as follows. In next section, we describe and analyze the complexity of the algorithm for the generation of an AND/OR goal tree and its evaluation. In Section 3.3, we develop two algorithms by assuming that document terms are binary and query terms are weighted; one for generation of MTSs, and

```
[{"P2/233", "MMX", "64MB", "512K", "6.4GB", "EIDE", "ATI", "4MB/3D"}, 0.8]
[{"P2/233", "MMX", "64MB", "Cache", "6.4GB", "EIDE", "ATI", "4MB/3D"}, 0.8]
[{"P2/233", "MMX", "64MB", "512K", "6.4GB", "EIDE", "AGP", "4MB/3D"}, 0.7]
[{"P2/233", "MMX", "64MB", "Cache", "6.4GB", "EIDE", "AGP", "4MB/3D"}, 0.7]
[{"P2/233", "MMX", "32MB", "512K", "6.4GB", "EIDE", "ATI", "4MB/3D"}, 0.5]
[{"P2/233", "MMX", "32MB", "512K", "6.4GB", "EIDE", "AGP", "4MB/3D"}, 0.5]
[{"P2/233", "MMX", "32MB", "Cache", "6.4GB", "EIDE", "ATI", "4MB/3D"}, 0.5]
[{"P2/233", "MMX", "32MB", "Cache", "6.4GB", "EIDE", "AGP", "4MB/3D"}, 0.5]
```

Figure 3.2: Minimal Term Sets of the P2/233 Desktop Subconcept.

the second for evaluation of MTSs. Both generation and evaluation take strictly into account the calculus of the rule-based retrieval model. Furthermore, partial evaluation of conjunctions is possible so that the user (or information searcher) can limit the size of the retrieval output. Partial evaluation is facilitated by use of normalized retrieval functions such as the cosine function. In the final section, we provide the conclusion of this chapter.

3.2 AND/OR Goal Tree

Let $W(D_i, T_j) = w_{ij}$ be a weight assignment function, where D_i is the i^{th} document in a document collection, T_j is the j^{th} term name in a set of terms used for indexing documents, and w_{ij} is the significance of j^{th} term in i^{th} document. An auxiliary structure realizing the weight assignment function is called an index file or an inverted-index file, if weights are accessible by term names. In the Prolog database we represent the inverted-index file by a set of facts described as follows. The fact $T_i(D_j, w_{ji})$ exists in Prolog database, if the presence of term T_i in document D_j has a significance of w_{ji} .

The algorithm *Generate_AND/OR* produces a Prolog-like code for a given query topic such that the retrieval output, a list of documents and their retrieval status values (RSV), is obtained as a result of processing AND/OR goal tree. It takes two input files: A set of facts corresponding to the inverted index file, and a set of rules for a query topic. The output of the algorithm is simply a Prolog-like code that is used for evaluating an AND/OR goal tree for a given query topic. The critical steps of the algorithm are given as follows.

Step 1. First we combine the rules with the same head using the “or” operator. Then for each rule, we create its disjunctive normal form, where the query topic appears as a set of conjunctions of one or more terms, each conjunction being connected to the next by an “or” operator, as shown below.

$$QueryTopic = \{CF_1, CF_2, \dots, CF_i, \dots, CF_m\},$$

where “QueryTopic” is the head of a rule, and CF_i is the conjunction of its term conjuncts, which can be either *structure* or *functor*. If a subquery or text word is paired with a user-supplied weight, we call it structure; otherwise we name it as functor. Since the default weight for a functor is one, an explicit weight of one is assigned when constructing a disjunction of conjunctions representing a QueryTopic. In Figure 3.3, we show the intermediate form of the rule set given in Figure 3.1. Our notation in assigning weights to terms in the body of a rule differs from the notation of RUBRIC in which a single weight is associated with the head of a rule, but notice that both become semantically equivalent expressions once an appropriate calculus

```

Stratum 0:
  AMD      = {{["K6/233", 0.6]}, {"K6/266", 0.9]}}
  CACHE    = {{["512KB", 1]}, {"Cache", 1]}}
  HARD_DRIVE = {{["6.4GB", 1], ["EIDE", 1]}}
  PENTIUM  = {{["P2/233", 0.8]}, {"P2/266", 1]}}
  RAM      = {{["32MB", 0.5]}, {"64MB", 1]}}
  VIDEO    = {{["ATI", 0.8], ["4MB/3D", 0.9]},
              {"AGP", 0.7], ["4MB/3D", 0.9]}}

Stratum 1:
  PROCESSOR = {{[PENTIUM, 1], ["MMX", 1]},
              [AMD, 1], ["MMX", 1]}}

Stratum 2:
  DESKTOP   = {{[PROCESSOR, 1], [RAM, 1], [CACHE, 1],
              [HARD_DRIVE, 1], [VIDEO, 1]}}

```

Figure 3.3: Disjunctive Normal Forms of rules defined in Figure 3.1.

for the combination of weights and propagation of intermediate results is specified.

Step 2. Let QT denote the rule “QueryTopic.” We say the rule QT_i is dependent on the rule QT_j if there exists a reference in the body of QT_i to QT_j . This stratification defines a partial order on the rule set (or DNFs.) Furthermore, we define a total order on the rule set by requiring $QT_j \leq QT_i$, if QT_j is in a lower stratum than QT_i or QT_j and QT_i are in the same stratum. We evaluate the rule set by forward chaining such that the rules in lower stratum are evaluated before processing the rules in higher level. As seen in Figure 3.3, we have three classes of the rules for the Desktop Concept and all rules in a stratum are in lexicographical order.

Step 3. Assume that the rules in regard to a query topic are ordered as described in Step 2. Starting from the first rule, Step 3 will be repeated until each rule in the totally ordered rule set is considered. Let CF_{km} be the m^{th} conjunction in the k^{th} rule.

Let $CF_{km} = \{[T_1, w_1], \dots, [T_n, w_n]\}$, where the pair $[T_i, w_i]$ is i^{th} term with weight w_i . We verbally say that a document represented by $ID\#$ and RSV is in the predicate set $CF_{km}(ID\#, RSV)$ if it is satisfied by this predicate. A similar interpretation is valid for $T_i(ID\#, RSV)$, where the term T_i is either a subordinate query topic or a text reference.

The generation of a prolog-like rule indicates that if a document contains all terms T_i , $1 \leq i \leq n$, then it is in CF_{km} with retrieval status value computed as follows. T_i has two roles in evaluating retrieval status value of a document in CF_{km} : one is its significance in a document as a document term, denoted by RSV_i , and the other is its significance in a user query as a search term, denoted by w_i . These two significance factors are multiplied to compute overall effect of a term and the resulting value is denoted by N_i . This would be enough in determining the retrieval status value of a document, if we had only one term in CF_{km} . In case we have multiple terms in defining CF_{km} , we take the minimum of all N 's which means that the combined effect of all terms in determining the retrieval status value of a document in CF_{km} is given by the minimum N over all terms.

Each conjunction in a given query topic is treated as an independent subordinate query topic such that it offers an alternate path in arriving at the query topic. In case we have multiple conjunctions, the overlapping cases (i.e., a document might be present in more than one predicate set) must be considered to avoid duplicated documents in the retrieval output of QT_k . For m conjunctions $C(m, i)$ gives the num-

ber of combinations for which a document satisfies i conjunctions, i.e., the number of overlapping cases with size i is $C(m, i)$. Therefore, we need to consider as many as $\sum_{i=1}^m C(m, i) = 2^m - 1$ cases to avoid duplications. We encode each region using a binary string of size m which is in the set generated by the regular expression $\{0, 1\}^m - \{0^m\}$, and for each region r we generate the corresponding QT_k^r . The RSV of the document with $ID\#$ in the predicate set $QT_k^r(ID\#, RSV)$ equals the maximum of the RSV s of the document satisfying alternative definitions in the region r for QT_k . For $m = 3$, we have, for example, seven regions and the encoding value $(101)_2 = 5_{10}$ indicates a predicate QT_k^5 whose evaluation contains the documents satisfying only the first and third conjunctions.

We have described an algorithm that generates a Prolog-like code for a given rule set. Considering the time complexity of the *Generate_AND/OR* algorithm, we see that Step 3 is the most time consuming step. Let k , m and n be the number of subordinate query topics plus the query topic, the maximum number of conjunctions in a (subordinate) query topic, and the maximum number of terms in a conjunction, respectively. Then Step 3 takes at most $O(km(n + 2^m))$. This is because (1) we have k DNFs, and for each DNF we have at most m conjunctions, and finally we need to specify at most n term predicates for each conjunction; and (2) the evaluation of a DNF without duplication requires us to consider 2^m cases, each with m conjunctions. From this explanation it is easy to deduce that the time complexity of AND/OR evaluation is bounded by $O(km(n + 2^m)J)$, where J is computational effort to compute

a predicate set.

Example 3.1 Let a Desktop concept be defined as in Figure 3.1. The DNFs of the rules and their evaluation orders are given in Figure 3.3. Now, consider the VIDEO subconcept in stratum zero. The produced Prolog-like code for this subconcept is as follows.

```
CF00(ID#,RSV):- F_ATI(ID#,RSV1),
                F_4MB/3D(ID#,RSV2),
                N1  = RSV1 * 0.8,
                N2  = RSV2 * 0.9,
                RSV = min([N1,N2]).

CF01(ID#,RSV):- F_AGP(ID#,RSV1),
                F_4MB/3D(ID#,RSV2),
                N1  = RSV1 * 0.7,
                N2  = RSV2 * 0.9,
                RSV = min([N1,N2]).

VIDEO(ID#,RSV):- CF00(ID#,RSV),
                 not(CF01(ID#,_)).

VIDEO(ID#,RSV):- CF01(ID#,RSV),
                 not(CF00(ID\#,_)).

VIDEO(ID#,RSV):- CF00(ID#,RSV1),
                 CF01(ID#,RSV2),
                 RSV = max([RSV1,RSV2]).
```

Notice that to compute the predicate VIDEO we only need F_ATI, F_4MB/3D, and

F_AGP text references¹. Assume that the weights of these text references are as follows:

¹The “F_” prefix in front of each text reference is just syntactical sugar which is used to put a text reference into a proper fact form with respect to Prolog notation.

$$F_ATI(ID\#,RSV) = \{(d_2, 0.25), (d_3, 0.32), (d_5, 0.53), (d_8, 0.48), (d_{11}, 0.31), \\ (d_{15}, 0.47)\}$$

$$F_4MB/3D(ID\#,RSV) = \{(d_2, 0.27), (d_3, 0.46), (d_4, 0.16), (d_7, 0.39), \\ (d_8, 0.23), (d_9, 0.16), (d_{10}, 0.17), (d_{11}, 0.3), \\ (d_{13}, 0.19), (d_{15}, 0.22)\}$$

$$F_AGP(ID\#,RSV) = \{(d_2, 0.4), (d_4, 0.24), (d_5, 0.37), (d_6, 0.44), \\ (d_9, 0.7), (d_{10}, 0.76), (d_{11}, 0.22), (d_{14}, 0.22)\}$$

then the predicate sets of both CF00 and CF01 are

$$CF00(ID\#,RSV) = \{(d_2, 0.20), (d_3, 0.26), (d_8, 0.21), (d_{11}, 0.25), (d_{15}, 0.2)\}$$

$$CF01(ID\#,RSV) = \{(d_2, 0.24), (d_4, 0.14), (d_9, 0.14), (d_{10}, 0.15), (d_{11}, 0.15)\}.$$

Document d_2 is in the predicate set CF00 because it is present in both facts F_ATI and $F_4MB/3D$. The multiplication of the significance of the term ATI in d_2 , which is 0.25, by the search term weight of ATI , which is 0.8, yields its possible propagation value, which is $N1 = 0.2$. Similarly, given that the term $4MB/3D$ is present in document d_2 , the possible propagation value of document d_2 , denoted by $N2$, equals to $0.27 \times 0.9 = 0.24$. Since minimum of these two values is 0.2, the RSV of d_2 in the predicate set CF00 equals to 0.2, which we call the real propagated value. This process is repeated for computing RSVs of all documents in CF00 and CF01, respectively. Finally, The set of documents in VIDEO concept is obtained as follows.

1. The set of documents satisfying the conjunction CF00, but not CF01.

$$\text{VIDEO}(\text{ID\#}, \text{RSV}) = \{(d_3, 0.26), (d_8, 0.21), (d_{15}, 0.2)\}$$

2. The set of documents satisfying the conjunction CF01, but not CF00.

$$\text{VIDEO}(\text{ID\#}, \text{RSV}) = \{(d_4, 0.14), (d_9, 0.14), (d_{10}, 0.15)\}$$

3. The set of documents satisfying both conjunctions CF00 and CF01.

$$\text{VIDEO}(\text{ID\#}, \text{RSV}) = \{(d_2, 0.24), (d_{11}, 0.25)\}$$

The union of the above sets yields the set of documents in VIDEO subconcept. Notice that whenever a document satisfies a conjunction of a DNF, it must be in the set of the DNF because of the fact that disjunction never eliminates a document once it is in one of its conjunctions. In case a document satisfies more than one conjunction of a DNF predicate with different RSVs, the maximum of these RSV values becomes the RSV of the document in the DNF predicate set. In this example, document d_2 satisfies both conjunctions CF00 and CF01 with RSV values 0.20 and 0.24, respectively, then its new weight in VIDEO predicate set becomes $\max(0.20, 0.24) = 0.24$. ■

3.3 Minimal Term Sets

Two points need addressing in the evaluation of an AND/OR goal tree. First, the algorithm *Generate_AND/OR* can handle vector based models (VSMs). Actually, nothing in the algorithm needs to be changed to accommodate real valued weights of document terms. There is, however, an aspect that is not consistent with the premises of the VSM: the query processing in AND/OR evaluation makes it impossible for a document to be in the retrieval output, if it does not satisfy any conjunction of the

rule set in DNF form. This sounds like a Boolean query model rather than a VSM. The second point is the exponential nature of AND/OR evaluation, which depends on m in a (subordinate) query topic. It is reasonable to expect that the average value of m may be small. In this section, we address these two points from the perspective of evaluation of MTSs.

3.3.1 Binary Weights and Simple Retrieval Functions

The minimal term set is the result of the process of expressing a query topic in terms of its text references (or index terms) as shown in Figure 3.2. The essence of our discussion here is to show that both AND/OR goal tree and MTSs evaluation end up with the same retrieval output, provided that term weights are Boolean.

Generate_MTS. The input of the algorithm is just a syntactically correct and acyclic query topic definition. The two steps of the *Generate_AND/OR* algorithm, converting rules into their DNFs and defining a total order on the rules with respect to dependency relationship, are repeated in the *Generate_MTS* algorithm. The third step is sketched in Algorithm 1.

Algorithm 1 *Third Step of MTSs Generation.*

1. /* Let the total order on DNFs be $\{QT_1, QT_2, \dots, QT_k\}$ */
2. **for** $l = 1$ **to** $k - 1$ **do**
3. let $QT_l = \{CF_{l1}, CF_{l2}, \dots, CF_{lm}\};$
4. **for** $u = 1$ **to** m **do**
5. let $CF_{lu} = \{[T_1, W_1], [T_2, W_2], \dots, [T_i, W_i], \dots, [T_n, W_n]\},$
6. where each T_i is a text word;
7. find the minimum weight, say mw , of terms in $CF_{lu};$

8. **for** each QT_d depending on QT_l , where $d > l$, **do**
9. let $QT_d = \{CF_{d1}, CF_{d2}, \dots, CF_{dy}\};$
10. let $CF_{dv} = \{[T_1, W_1], [T_2, W_2], \dots, [T_j, W_j], \dots, [T_x, W_x]\}$
11. such that $v \leq y$ and $T_j = QT_l$;
12. **for** each such CF_{dv} **do**
13. **if** $u < m$ **then**
14. Create CF_{new} as a duplicate of CF_{dv}
15. $QT_d = QT_d \cup \{CF_{new}\}$
16. **end if**
17. $CF_{dv} = CF_{dv} - \{[T_j, W_j]\};$
18. **for** each $[T_i, W_i]$ **in** CF_{lu} **do**
19. $CF_{dv} = CF_{dv} \cup \{[T_i, mw * W_j]\}$
20. Convert each conjunct
21. $CF_{ki} = \{[T_1, W_1], [T_2, W_2], \dots, [T_n, W_n]\}$ of QT_k to
22. $CF'_{ki} = [\{T_1, T_2, \dots, T_n\}, \min(W_1, W_2, \dots, W_n)]$, and then
23. Sort the conjunctions in descending order of their search weights.

The time complexity of this algorithm in the worst case is in $O(k^2m^n)$, where: k is the number of rules (or DNFs); m is the maximum number of conjunctions; and n is the maximum number of terms in a conjunction.

Theorem 3.3.1 *Let QT be a query topic that is syntactically correct and acyclic. Let $QT_{AND/OR}(ID, RSV)$ and $QT_{MTS}(ID, RSV)$ be the set of pairs of documents and their retrieval status values, when evaluated by AND/OR goal tree and MTSs, respectively. Assume that term weights are binary. Then,*

$$QT_{AND/OR}(ID, RSV) = QT_{MTS}(ID, RSV)$$

if the evaluation model of MTS is realized in two phases:

1. For each conjunction of text references, use Boolean query processing and then associate the conjunction weight with each document in the retrieval output; and
2. When all conjunctions are exhausted, eliminate duplicated documents by keeping the one with maximum weight.

Proof. It is clear that processing of a Prolog-Like rule generated by RUBRIC system is similar to processing a query in a Fuzzy-Like retrieval model. In other words, a predicate term $T_i(ID\#, RSV_i)$ is considered as a member function that returns RSV_i as the truth degree of term T_i in document $ID\#$. Furthermore, if, for example, a rule consists of two predicates, namely, T_i and T_j , a document satisfying this rule will have a RSV that equals the minimum of the two weighted RSV's. So, consider the following rule for a conjunction.

$$P(ID\#, RSV) : - T_1(ID\#, RSV_1), T_2(ID\#, RSV_2), \dots, T_n(ID\#, RSV_n),$$

$$N_1 = RSV_1 * w_1,$$

$$N_2 = RSV_2 * w_2,$$

$$\vdots$$

$$N_n = RSV_n * w_n$$

$$RSV = \min(N_1, N_2, \dots, N_n).$$

According to the hypothesis of the theorem, document term weights are binary, it follows that RSV_i of a document in a predicate set $T_i(ID\#, RSV_i)$ is evaluated to 1. Thus, for a document satisfying the above rule, $N_i = w_i$ and $RSV = \min_i \{w_i\}, 1 \leq$

$i \leq n$. Therefore, *on the one hand*, the rule evaluation depends only on query term weights, and *on the other hand*, as given in Algorithm 1, in Step 3 of *Gen_MTS* algorithm, a conjunction weight is generated by computing the minimum of only query term weights in the conjunction, and the result follows. $\square$

Remark 1. Evaluation of the MTSs is simply a table lookup operation. If the MTSs evaluation is realized in two phases as described in Theorem 3.3.1, the time complexity of the first phase of MTSs evaluation is of $O(m * [(n - 1) * J])$, where J is the computational effort to evaluate term lists in an inverted index file and to intersect two term lists. The time complexity of the second phase is bounded by that of the sorting algorithm for documents in the retrieval output.

Remark 2. Let $CF_i = \{[T_1, \dots, T_m], c_i\}$ and $CF_j = \{[T_1, \dots, T_n], c_j\}$, where CF_i and CF_j , $i \neq j$, are conjunctions in a given MTS form. Furthermore let us introduce the following useful functions.

1. $t(CF) =_{def.} \{T_1, \dots, T_m\}$ is a function that returns a set of terms included in a given conjunction CF in MTS form.
2. $tt(D) =_{def.} \{T_1, \dots, T_n\}$ is a function that returns a set of terms representing a given document.
3. $\gamma(CF) =_{def.} c$ is a function that returns the constant value associated with a given conjunction CF in MTS form.

For a given conjunction and document, a Boolean retrieval function based on term matching can be defined by the following characteristic function.

$$\beta(CF, D) = \begin{cases} 1 & \text{if } t(CF) \subseteq tt(D); \\ 0 & \text{otherwise.} \end{cases} \quad (3.1)$$

Let $\alpha(CF, D) =_{def.} \beta(CF, D) * \gamma(CF)$ be a retrieval function of the MTS evaluation.

It is easy to see that, if $t(CF_i), t(CF_j) \subseteq tt(D)$ and $\gamma(CF_i) \geq \gamma(CF_j)$,

$$\alpha(CF_i, D) \geq \alpha(CF_j, D).$$

Remark 2 allows us to impose a limit on the number of documents in the retrieval output, provided that the conjunctions in MTS form are sorted in descending order of search weights. We guarantee that further evaluation of subsequent conjunctions would not retrieve a document with higher retrieval value than the documents in the retrieval output. This is a useful property, i.e., partial evaluation of MTS conjunctions, that reduces the time complexity of MTSs evaluation. For example, the number of conjunctions for Desktop concept in Figure 3.1 is 32; that is, a full-shot evaluation of the MTSs requires to run the retrieval function for each conjunction over the inverted index file. Instead, as a result of Remark 2, whenever we get the top n distinct documents, we stop retrieving other documents.

3.3.2 Binary Weights and Generalized Retrieval Functions

A conjunction in the form of MTS consists of conjunctions of explicit terms and its interpretation for a given document requires that terms of the document subsume explicit terms of the conjunction, but this restriction is quite strong for information

retrieval systems, even if each conjunction presents an alternative specification of a query topic. We may, therefore, want to consider other retrieval functions under the assumption we view a conjunction as a set of terms, for example

$$\beta'(CF, D) = \frac{|t(CF) \cap tt(D)|}{|t(CF) \cup tt(D)|}, \quad (3.2)$$

is a cosine function of Boolean terms. With this retrieval function, the hypothesis of Remark 2 does not necessarily hold. However, we still have the ability to identify a subset of documents in the evaluation set of the CF_i such that their RSVs are at least equal to the maximum RSV that can be obtained by evaluating the CF_{i+1} – note that $\gamma(CF_i) \geq \gamma(CF_{i+1})$. This is due to the fact that the cosine retrieval function has the maximum value of one.

Let us introduce the MTSs evaluation as shown in Algorithm 2. *QueryTopic* is a set of conjunctions sorted in descending order of their search weights, and n is the maximum size of retrieval output. Assume that $\gamma(null) =_{def} 0$. Furthermore, the *TempList* is an ordered set of pairs of documents and their RSVs. This descending order of RSVs is assumed to be kept during the application of union operation. Projection of a relation $R(A_1, A_2, \dots, A_n)$ onto some attributes $X \subseteq \{A_1, A_2, \dots, A_n\}$ is denoted by $\pi_X R(A_1, A_2, \dots, A_n)$.

Algorithm 2 *MTS_Evaluation(QueryTopic, n)*

1. $m \leftarrow |QueryTopic|$; // Number of Term Lists
2. $RetrievalOutput \leftarrow \emptyset$;
3. $TempList \leftarrow \emptyset$;

```

4.   $MaxRetVal \leftarrow 1$ ;
5.   $MaxTempRSV \leftarrow 0$ ;
6.  for  $i = 1$  to  $m$  do
7.     $MaxRSV \leftarrow MaxRetVal * \gamma(CF_{i+1})$ 
8.    for each document  $D$  of the collection do
9.       $RSV \leftarrow \beta(CF_i, D) * \gamma(CF_i)$ ;
10.     if  $RSV > MaxRSV$  and  $RSV > MaxTempRSV$  then
11.        $RetrievalOutput \leftarrow RetrievalOutput \cup \{(D, RSV)\}$ ,
12.       provided that  $D \notin \pi_D RetrievalOutput(D, RSV)$ ;
13.     else
14.        $TempList \leftarrow TempList \cup \{(D, RSV)\}$ ;
15.       if  $RSV > MaxTempRSV$  then  $MaxTempRSV \leftarrow RSV$ ;
16.     for each  $(TempD, TempRSV) \in TempList$ 
17.       such that  $TempRSV > MaxRSV$  do
18.        $TempList \leftarrow TempList - \{(TempD, TempRSV)\}$ ;
19.        $RetrievalOutput \leftarrow RetrievalOutput \cup \{(TempD, TempRSV)\}$ ,
20.       provided that  $TempD \notin \pi_D RetrievalOutput(D, RSV)$ 
21.      $MaxTempRSV \leftarrow \max(RSVs \in TempList)$ ;
22.     if  $|RetrievalOutput| \geq n$  then exit the loop;
23.   for each  $(D, RSV) \in TempList$  do
24.     if  $|RetrievalOutput| \geq n$  then exit the loop
25.      $RetrievalOutput \leftarrow RetrievalOutput \cup \{(D, RSV)\}$ ,
26.     provided that  $D \notin \pi_D RetrievalOutput(D, RSV)$ ;
27.   Sort  $RetrievalOutput$  in descending order of RSVs;
28.   return  $RetrievalOutput$ ;

```

Algorithm 2 is general enough to work with other retrieval functions as long as they have upper bound values. The essence of this algorithm is to append a pair of document and its RSV to the list of retrieval output if and only if it is guaranteed that the RSV of that document is at least as high as the RSVs of all remaining documents

that will be appended to the retrieval output.

3.3.3 Sharing Of Rule Bases

Assuming that a rule itself corresponds to a concept, we organize concepts into a concept hierarchy (specifically AND/OR goal tree) which actually yields a user view. This is because, when a user defines a concept and s(he) wants it to be persistent, we keep them in her/his profile. In other words, to honor users' preferences, it is natural to allow them to define their own views of concepts which may not necessarily coincide with others' views.

It is, however, feasible to provide means to users to share local knowledge bases. To do this, there is no need to integrate local views. Assume that each user incorporates the sharing process by allowing some or all her/his persistent concepts to be exported (or to be opened to the external world). Furthermore, assume that for each concept we keep its intensional definition, e.g., the MTSs in Figure 3.2 constitutes the intension of the P2/233 Desktop concept. If a new user wants to browse a concept list, he/she is asked to roughly define the concept in terms of index terms, which are called text references in RUBRIC, using a query interface. The user's definition is matched against the MTSs of persistent and exported concepts and the resulting set in the form of rules is returned to the user.

This scheme of course requires one-to-one correspondence between a rule and its MTSs in the rule base.

3.4 Conclusion

In this chapter, we developed a new approach for computing retrieval output in the context of a rule-based retrieval system. The proposed approach involves the preprocessing of the rule base to generate minimal term sets that enhance the retrieval process. We provide an algorithm for computing the minimal term sets and derive the associated computational complexity. We also show that the computation of the retrieval status values of documents in a collection is very efficient, when the method based on MTS is adopted. Besides the advantage of efficient processing, the availability of MTSs can enable users to select query topics best suited to their needs among those available in the system. We developed an approach to generalize the RUBRIC model to obtain a refined retrieval output, under the assumption that MTSs are available.

Chapter 4

Retrieval with Extended Boolean Models and Document Term Weights

4.1 Introduction

In this chapter, we develop strategies for enhancing retrieval output by incorporating document term weights into the computation of document-query similarity measures for both AND/OR Goal tree and MTSs, based on the p -Norm model [39]. In the following we refer to query topics as *abstract query terms* and explicit terms (or text references) as *concrete query terms*.

The evaluation of an AND/OR Goal tree is a rule-based retrieval model, for a given query topic, based on recursive application of the following calculus.

1. Select the minimum query term weights in a conjunction.
2. Select the maximum query term weights in a disjunction.

In addition, for evaluating an abstract query term, the following additional calculus is needed.

3. Calculate the product of a user defined weight and the propagated value obtained by Step 1 and 2.

In this rule-based retrieval model, document term weights are assumed to be binary. In Section 3.3, we established the notion of MTS and proved that the evaluation of the AND/OR goal tree and MTSs come up with the same document ranking. One problem with this kind of computation calculus is that it ranks documents depending only on the lowest or highest weights of query terms for conjunctions and disjunctions, respectively, which suffers, unfortunately, from a lack of discrimination among documents nearly to the same extent as the conventional retrieval systems do. Even though in [29] it was not stated explicitly, the AND/OR goal tree in Section 3.2 was defined in a way that the document term weights can be readily processed. This was because we defined query term weights instead of implication factor (as was done in [29]). Now, in this section, we attempt to explore the case where document term weights are incorporated into the computation of similarity between a query and a document. In this case, we need to find a finer computation of retrieval status values than that of the rule-based retrieval model. We consider the extended Boolean retrieval model developed by Salton et al. [39] as our computational model and show how query-document similarity is computed using both AND/OR Goal tree and MTS model, respectively.

p -Norm Model. We present a brief description of the extended Boolean retrieval model. The Boolean model is extended using a distance function of L_p vector norm

reflecting the extent of query-to-document similarity in an n -dimensional vector space. In the extended model, the queries and documents are represented as follows. Let some document D in a given collection be denoted by $(d_1, d_2, \dots, d_n)$, where D is a p -Norm vector with weights of $W(D, T_i) = d_i$ corresponding to document terms T_i , $1 \leq i \leq n$. We assume that the generalized AND and OR queries are expressed, respectively, as follows

$$Q_{AND_p} = (T_1, w_1) \text{ AND}_p (T_2, w_2) \text{ AND}_p \cdots \text{ AND}_p (T_n, w_n), \quad (4.1)$$

$$Q_{OR_p} = (T_1, w_1) \text{ OR}_p (T_2, w_2) \text{ OR}_p \cdots \text{ OR}_p (T_n, w_n) \quad (4.2)$$

where the coefficient p ranges between 1 to ∞ and indicates the degree of strictness, with 1 being the least strict and ∞ being the most strict.

Now, the query-document similarity is calculated using the formula:

$$\text{SIM}(Q_{OR_p}, D) = \sqrt[p]{\frac{w_1^p d_1^p + \cdots + w_n^p d_n^p}{w_1^p + \cdots + w_n^p}} \quad (4.3)$$

$$\text{SIM}(Q_{AND_p}, D) = 1 - \sqrt[p]{\frac{w_1^p (1 - d_1)^p + \cdots + w_n^p (1 - d_n)^p}{w_1^p + \cdots + w_n^p}} \quad (4.4)$$

It is shown in [39] that for $p = \infty$, equations (4.3) and (4.4) are reduced to:

Query evaluation in the Boolean retrieval model if document and query term weights are binary indicating presence or absence of terms; or

Query evaluation in the fuzzy-set retrieval model if document terms are weighted but query terms are still binary.

Furthermore, by setting p to 1 we obtain the normalized coordinate-level matching for query evaluation in the vector-processing retrieval model. The best intermediate value for p is determined empirically for a collection but typically is in the range $2 \leq p \leq 5$. It should be noted that for the intermediate values of p , finer retrieval status values are obtained at the expense of a small loss in query structure; that is, smaller p values make some term phrases less distinguishable.

4.2 Incorporating Document Term Weights into AND/OR Goal Tree

In this section we describe a strategy to incorporate document term weights into an AND/OR Goal tree. Our approach consists of adopting the skeleton of the algorithm for generation of the AND/OR Goal tree presented in Section 3.2. It is easy to see that the skeleton of the algorithm is still applicable when term weights are incorporated. Moreover, several steps in the algorithm can be used without any change. For example, Steps 1 and 2 of the algorithm (i.e., conversion of a query topic, or a set of query rules, to its DNF, and imposing a total order on the conjunctions in the stratum) can be imported without change. However, Step 3 involving the generation of the Prolog code for each rule must be changed for the case of the AND/OR Goal tree with respect to the p -Norm model. The changes required in this step are presented

in the following paragraph.

Without loss of generality, let us call equations (4.3) and (4.4) as OR_p and AND_p formulas, respectively. Consider a query in the form of an AND/OR goal tree. We define an evaluation scheme for such a query as follows:

1. Instead of the operator *min*, apply the AND_p formula for the conjunctions, and
2. Instead of the operator *max*, apply the OR_p formula for the disjunctions.

We introduce a new term weight, *computed document term weight*, for an abstract document term weight because the relative weight of an abstract term in a given document can be approximated by the value obtained from the evaluation of the goal tree rooted at the abstract term that involves a finite applications of OR_p or AND_p formulas.

We assume that a query topic is converted into its DNF. This not only simplifies the representation of the dependency relation between rules but also makes the OR_p and AND_p formulas simple to implement. It is, however, appropriate to note that the associative and distributive properties of Boolean operators do not hold any more in the extended Boolean retrieval model [39]. We can at most utilize the quasi-associativity property of AND_p and OR_p operators, which is not sufficient by itself to generate a DNF that is equivalent to AND/OR goal tree representation for a given query topic. Hence, we assume that p is set to ∞ when generating DNF of AND/OR goal tree and then, p is assumed to be set to an intermediate value. The first assumption renders a query structure equivalent to that of the conventional Boolean

retrieval models, and the second assumption allows us to mimic the query evaluation in vector-processing retrieval models. This provides query evaluation similar to that of the SIRE system in which a strict Boolean match is first used and then retrieved documents are ranked in descending order by the sum of the matching query and document terms [31].

Example 4.1 Let us consider again the Desktop concept defined in Figure 1.3. The Prolog-like code for VIDEO subconcept in stratum zero is produced as follows. The variable RSV_i should be interpreted (a) as a document term weight if the term is a concrete term, or (b) as computed document term weight if the term is an abstract term. Assume that (a) the p value is set to 2 and (b) the query (or search) weight of a conjunction is simply set to 1¹.

```
CF00(ID#,RSV):- F_ATI(ID#,RSV1),
                 F_4MB/3D(ID,RSV2),
                 D1 = 1-RSV1, D2=1-RSV2,
                 W1 = 0.8, W2 = 0.9,
                 square(D1 * W1,SP1), square(D2 * W2,SP2),
                 square(W1,W1P), square(W2,W2P),
                 RSV = sqrt((SP1 + SP2) / (W1P + W2P)).

CF01(ID#,RSV):- F_AGP(ID#,RSV1),
                 F_4MB/3D(ID#,RSV2),
                 D1 = 1-RSV1, D2 = 1-RSV2,
                 W1 = 0.7, W2 = 0.9,
                 square(D1 * W1,SP1), square(D2 * W2,SP2),
```

¹An alternative assumption would be to take the average of query weights of conjuncts, which is advised in [39].

```

square(W1,W1P), square(W2,W2P),
RSV =sqrt((SP1 + SP2) / (W1P + W2P)).
VIDEO(ID#,RSV):- CF00(ID#,RSV1),
not(CF01(ID#,_)),
RSV = sqrt((RSV1 * RSV1)/2).
VIDEO(ID#,RSV):- CF01(ID#,RSV1),
not(CF00(ID#,_)),
RSV = sqrt((RSV1 * RSV1)/2).
VIDEO(ID#,RSV):- CF00(ID#,RSV1),
CF01(ID#,RSV2),
RSV = sqrt(((RSV1 * RSV1)+(RSV2 * RSV2))/2).

```

Assume that document weights of text references in VIDEO predicate, namely F_ATI, F_4MB, and F_AGP, are as follows.

```

F_ATI(ID#,RSV)    = {(d2, 0.56), (d3, 0.32), (d5, 0.53), (d8, 0.48),
                     (d11, 0.31), (d15, 0.47)}
F_4MB/3D(ID#,RSV) = {(d2, 0.27), (d3, 0.46), (d4, 0.16), (d7, 0.39),
                     (d8, 0.23), (d9, 0.16), (d10, 0.17), (d11, 0.3),
                     (d13, 0.19), (d15, 0.22)}
F_AGP(ID#,RSV)    = {(d2, 0.4), (d4, 0.24), (d5, 0.37), (d6, 0.44),
                     (d9, 0.7), (d10, 0.76), (d11, 0.22), (d14, 0.22)}

```

then the predicate sets of both CF00 and CF01 are

```

CF00(ID#,RSV) = {(d2, 0.38), (d3, 0.37), (d8, 0.33), (d11, 0.30), (d15, 0.32)}
CF01(ID#,RSV) = {(d2, 0.32), (d4, 0.19), (d9, 0.31), (d10, 0.33), (d11, 0.27)}.

```

Document d_{11} is in the predicate set CF00 because it is present in both facts F_ATI and F_4MB/3D. The retrieval status value of d_{11} is computed by applying AND_p on both F_ATI and F_4MB/3D as specified in CF00. Since the significance of ATI and 4MB/3D in d_{11} and their corresponding search term weights are 0.56, 0.27, 0.8, and 0.9, respectively, then the retrieval status value of d_{11} yields to 0.3, using AND_p formula. This process is repeated for computing RSVs of all documents in CF00 and CF01, respectively.

Finally, The set of documents in VIDEO concept is obtained as follows.

1. The set of documents satisfying the conjunction CF00, but not CF01.

$$\text{VIDEO}(\text{ID\#}, \text{RSV}) = \{(d_3, 0.26), (d_8, 0.21), (d_{15}, 0.23)\}$$

2. The set of documents satisfying the conjunction CF01, but not CF00.

$$\text{VIDEO}(\text{ID\#}, \text{RSV}) = \{(d_4, 0.13), (d_9, 0.22), (d_{10}, 0.23)\}$$

3. The set of documents satisfying both conjunctions CF00 and CF01.

$$\text{VIDEO}(\text{ID\#}, \text{RSV}) = \{(d_2, 0.35), (d_{11}, 0.29)\}$$

Notice that a document satisfying a predicate of a conjunction must also be in the DNF predicate set. Therefore, in order to compute the RSV of a document in the DNF predicate, its RSV is considered a computed document term weight and its search term weight is set to 1. For example, d_{11} satisfies both CF00 and CF01 predicates with RSVs (as computed document term weights) 0.3 and 0.27, respectively, then its new weight in VIDEO predicate set is equal to 0.29 which is computed by

the OR_p formula. Finally, the union of the above sets yields the set of documents in VIDEO\verb subconcept. The ranking of all documents in VIDEO(ID#,RSV) predicate will be in order of RSV as $\{(d_2,0.35), (d_{11},0.27), (d_3,0.26), (d_{10},0.23), (d_{15},0.23), (d_9,0.22), (d_8,0.21), (d_4,0.13)\}$. It is clear that documents d_2 and d_{11} have the highest ranking values due to the plausible fact that they satisfy a larger number of conjunctions of VIDEO subconcept compared to other documents. In other words, they belong to both CF00 and CF01 predicate sets. However, d_2 is ranked higher than d_{11} because ATI term has higher weight in d_2 than in d_{11} . ■

For intermediate values of p , even though term weights are involved, the query processing in the evaluation of the extended AND/OR Goal tree is still Boolean. In other words, documents having a partial list of the terms of conjunctions are completely prohibited from consideration. For instance, document d_5 in Example 3.1 is not included in the predicate set of CF00 because it contains the ATI term but not 4MB/3D. Thus, the benefit of AND_p and OR_p formulas are not fully utilized, i.e. when an AND_p formula is applied, a document that contains a partial list of the terms is not completely rejected but rather will have a lesser RSV, and when OR_p is used a document containing all terms will have higher RSV than that containing a partial list of the terms. Therefore, for intermediate values of p it is better to consider a single MTS as an alternative to AND/OR Goal tree where AND_p and OR_p formulas are fully utilized into the computation of document-query similarity.

4.3 Incorporating Document Term Weights into MTS Model

In this section, we develop a strategy for incorporating document term weights into MTS model. Clearly, when documents with term weights are considered, the MTSs generation given in Section 3.3 is no longer valid due to the fact that the weights of conjunctions are computed under the assumption that the corresponding query terms are present in the documents, and then this assumption is verified during retrieval of documents². Here, in adoption of the principles of extended Boolean retrieval to the MTS approach in a rule-based retrieval, we adhere to two basic postulates asserted in Section 4.2. That is, in generation of MTSs, we assume that the p value is, first, set to infinity so that we are able to utilize association and distribution properties of Boolean operators. Furthermore, contrary to the case for an AND/OR goal tree, we do not take document term weights into account when MTSs for a given query topic is generated. Second, during evaluation of MTSs, we use an intermediate value of p such that the query structure suffers a small loss. This also implies that the treatment of each conjunction as an independent subordinate query topic is not possible here due to the fact that the retrieval status value of a conjunction in the DNF of a query topic does not necessarily determine the resulting retrieval status value of that DNF.

Therefore, the disjunction of all conjunctions must be evaluated to rank documents.

²A strict verification is provided in Theorem 3.3.1 and then, the partial evaluation of MTSs (e.g., use of cosine retrieval function) is discussed to allow an information searcher to limit the size of retrieval output.

Under the restrictions given above, we define an evaluation scheme based on AND_p and OR_p formulas for the MTSs of a query topic as follows:

1. Assume that a query topic is converted to its MTSs representation (i.e., disjunction of conjunctions of concrete query terms) as described in Section 3.3.
2. Consider the weight of a conjunction, obtained as a result of MTS generation, as the extent of its relative importance in the query topic.
3. Preserve each concrete query term weight and apply the AND_p formula to the corresponding conjunction.
4. Apply the OR_p formula over conjunctions to get retrieval status value of a document for the query topic.

The algorithm for computing query-document similarity between a document D and an MTS query Q using AND_p and OR_p formulas is given in Algorithm 3. Here, first, the distance between each conjunction and the document D is computed. This distance is interpreted as the RSV of the document D when matched against a particular conjunction. As a matter of fact, since each conjunction contributes to an alternate path (a subordinate query leading to the query topic), the AND_p value of a conjunction determines the degree of containment of all terms of this conjunction in D ; that is, it indicates the computed document weight of corresponding conjunction. Therefore, after all these ‘computed document weights’ are determined, the relative

distance of the document to the query topic is computed using OR_p formula. Finally, the result of OR_p will be treated as the retrieval status value of D .

Algorithm 3 *Extended MTSs evaluation.*

1. $/* D = \{(T_1, d_{T_1}), (T_2, d_{T_2}), \dots, (T_k, d_{T_k})\}, */$
2. $/* Q = \{C_1, C_2, \dots, C_n\}. */$
3. $/* C_j = \{[(T_1, q_{T_1}), (T_2, q_{T_2}), \dots, (T_m, q_{T_m})], w_{c_j}\} */$
4. **Evaluate-MTS-Query**(D, Q)
5. Let $V = \{v_1, v_2, \dots, v_n\}$, where $v_j = w_{c_j}$;
6. $/*$ Compute AND_p for each conjunction C_j $*/$
7. **for** $j = 1$ **to** n **do**
8. $AND_p^{c_j} = 1 - \sqrt[p]{\frac{q_{T_1}^p(1-d_{T_1})^p + \dots + q_{T_m}^p(1-d_{T_m})^p}{q_{T_1}^p + \dots + q_{T_m}^p}}$
9. $/*$ Compute OR_p for Q $*/$
10. $RSV = \sqrt[p]{\frac{v_1^p(AND_p^{c_1})^p + \dots + v_n^p(AND_p^{c_n})^p}{v_1^p + \dots + v_n^p}}$
11. **return** RSV;

The advantage of the extended MTSs evaluation is that the user preferences are still honored in a sense that the management of user profiles or local knowledge is easier and more effective than dealing with the management of a single and global knowledge tree. In other words, users are allowed to associate the rules of concepts with their corresponding intensional definitions (MTSs) and to make this association persistent for future retrieval. This way, query-to-query similarity [32] is utilized in MTSs of persistent query topics when a new information searcher expresses his/her information need in terms of keywords connected with Boolean operators to browse

the relevant and persistent query topics, which are displayed as a set of rules. Notice that these displayed query topics are ranked with respect to their degree of closeness to the information need of the user. Furthermore, the MTS module can still be implemented on top of the descriptor-level retrieval system that maintains weights of descriptors in its auxiliary file(s). The advantage of partial evaluation embedded in the original MTS query for controlled ranking is, however, diminished in the extended MTS query evaluation. This is due to the fact that a document is not guaranteed to have a lesser RSV when evaluated by subsequent conjunctions.

4.4 Conclusion

In this chapter, extensions to AND/OR Goal tree and MTS model are developed by incorporating document term weights into the computation of document-query similarity. We adopted p -Norm model as our retrieval model with the parameter p set to ∞ when generating an equivalent DNF and set to an intermediate value for evaluation. The advantage of the extended MTSs evaluation is that user preferences can be honored in an efficient and effective way.

Chapter 5

On Modeling of Concept Based Retrieval in Generalized Vector Spaces

As mentioned earlier, one of the essential issues in information retrieval is to bridge the gap between the way users specify their information needs and the way queries are expressed. In RUBRIC, the retrieval output is determined by Boolean evaluation of the AND/OR tree, and can be enhanced by generating MTSs. However, since the Boolean evaluation ignores the term-term association unless it is explicitly represented in the tree, the terminological gap between users' queries and their information needs still remains. To solve this problem, we adopt the generalized vector space model in which the term-term association is well established, and thus extending the RUBRIC model. Experiments are performed on some variations of the extended RUBRIC model, and the results are compared to the original RUBRIC model based on recall-precision.

5.1 Introduction

Many intelligent retrieval approaches have been studied to meet the users preferences properly [9, 13, 29, 33]. However, there always exists a terminological gap between the way queries are defined and the way documents are represented. Users and authors of documents may have different knowledge and background. Different users may use different names, formats, and structures for the same targets. One of the approaches proposed in the literature for this purpose involves the use of production rules to capture user-query concepts (or topics). The central idea of such an approach was introduced in the context of RUBRIC [29] (see Chapter 3 for more details). In RUBRIC, the retrieval output is determined by Boolean evaluation of the AND/OR tree and can be enhanced by evaluating Minimal Term Sets (MTSs) [4, 6]. However, since the Boolean evaluation ignores the term-term association unless it is explicitly represented in the tree, the terminological gap between users queries and their information needs may still be significant. The original RUBRIC model uses MAX and MIN functions to compute the Retrieval Status Value (RSV) for each MTS. That is, it assigns the highest RSV to a document no matter how many MTSs coexist in the document. This useful information is just ignored. More accurate retrieval results can be obtained by using this information in the computation of RSVs. Additionally, the interpretation of the conjunctions of MTSs requires a given document that contains explicit terms of the conjunction, but this is also a strong provision for information retrieval systems, even if each conjunction presents an alternative specification of a

query topic.

To solve this problem, we adopt the Generalized Vector Space Model (GVSM) [49, 46]. The basic idea of a GVSM is to specify a vector space with a set of vectors as its basis (these vectors are called the atomic expressions for the minterms and they are supposed to be mutually linearly independent). Each term and thus each document can be represented as a vector in the space. In addition, a query corresponding to the user's preference is also represented as a vector. By computing the similarity between the query vector and the document vector, the RSV for a document corresponding to the query is obtained. The main advantage of GVSM is that the term-term relationships are also accounted for, therefore, it can give a fairly accurate retrieval output even if some terms in the query do not appear explicitly in a document. These term-term associations are computed as an integral part of the automatic indexing process. However, the difficulty of finding a query that is an accurate description of a user's need and easy to understand is still a challenge for GVSM. Therefore, in this chapter, we propose a strategy to integrate the ideas of concept based retrieval in RUBRIC with GVSM.

5.2 Precision/Recall Effectiveness Measures

The two major measures commonly associated with information systems are precision and Recall. These measures were used in Information Systems for many years. In information retrieval the term *relevant* document is used to represent a document

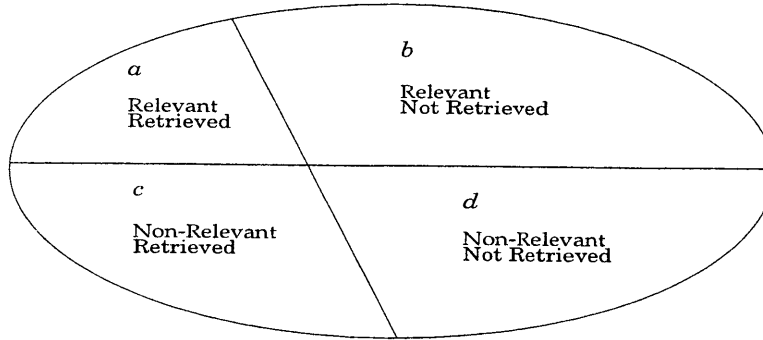


Figure 5.1: Effects of Search on Total Document Space

containing the needed information. When a user decides to issue a search looking for information on a topic, the total database is logically divided into four parts, consisting of relevant retrieved (*a*), relevant not retrieved (*b*), non-relevant retrieved (*c*), and non-relevant not retrieved (*d*). This situation is illustrated in Figure 5.1.

- *Recall* is the ratio of the number of relevant documents retrieved to the total number of relevant. In Figure 5.1,

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

- *Precision* is the ratio of the number of relevant documents retrieved to the total number of documents retrieved. In Figure 5.1,

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

Ideally, you would like to achieve both high recall and high precision. In reality, you must strike a compromise. Indexing terms that are specific yields higher precision at the expense of recall. Indexing terms that are broad yields higher recall at the

cost of precision. For this reason, an IR system's effectiveness is measured by the precision parameter at various recall levels known as Precision-Recall graph, which is explained in the next paragraph.

The user is usually presented with all retrieved documents as a ranked list. The user examines this ranked list starting from the top document. In this situation, the recall and precision measures vary as the user proceeds with his examination of the list. As an example, consider the following set of relevant documents to a query $q_i = \{d_3, d_5, d_9, d_{25}, d_{39}, d_{44}, d_{56}, d_{71}, d_{89}, d_{123}\}$. And assume the following ranked list is returned by the system for the query q_i

- | | | |
|----------------------|----------------------|-------------------|
| 1. $d_{123} \bullet$ | 6. $d_9 \bullet$ | 11. d_{38} |
| 2. d_{84} | 7. d_{511} | 12. d_{48} |
| 3. $d_{56} \bullet$ | 8. d_{129} | 13. d_{250} |
| 4. d_6 | 9. d_{187} | 14. d_{113} |
| 5. d_8 | 10. $d_{25} \bullet$ | 15. $d_3 \bullet$ |

The relevant documents are marked with a bullet after the document number. The system returned five relevant documents and 10 non-relevant documents. The first document, d_{123} , which is labelled relevant, corresponds to 10% (1/10) of all relevant documents. Thus we say that we have a precision of 100% at 10% recall. The next relevant document, d_{56} , which is the third document in the ranked list, corresponds to $(2/3) = 66.67\%$ precision at 20% recall. This computation is repeated and a precision versus recall curve is generated. The above ranked list did not contain all the ten relevant documents. Therefore, partial of the curve is presented. In general, to generate a precision versus recall curve, all relevant documents are required to appear in the ranked list so that precision is computed at different levels of recall in

the range 0%–100%.

In the above example, the precision and recall figures are for a single query. Usually, however, retrieval systems are evaluated by running them for several distinct queries. In this case, for each query a distinct precision versus recall curve is generated. To evaluate the retrieval performance of the system over all queries, we average the precision figures at each level of recall, and the result is *average* precision/recall curve. This average precision/recall curve of the evaluated system is used as the system's performance when compared with other retrieval systems.

5.3 Review of Rule-base Trees and MTSs

For the purpose of consistency with the material in this chapter, we present a brief summary with an example of a rule-base tree evaluation. Refer to Chapter 3 for more details.

In RUBRIC, concepts of interest are formulated using a two-step top-down refinement strategy. The first step is to express a given request as a single concept. The next step is to refine the initial concept by decomposing it into a set of component parts that are related by AND/OR logical operators. A component may take the form of a new concept defined at a different level of abstraction, a text expression, or a single index term. In each case, a weight is assigned to the individual concept-component pairs that are formed during the decomposition process. The assigned weight represents the user belief in the degree to which a given component

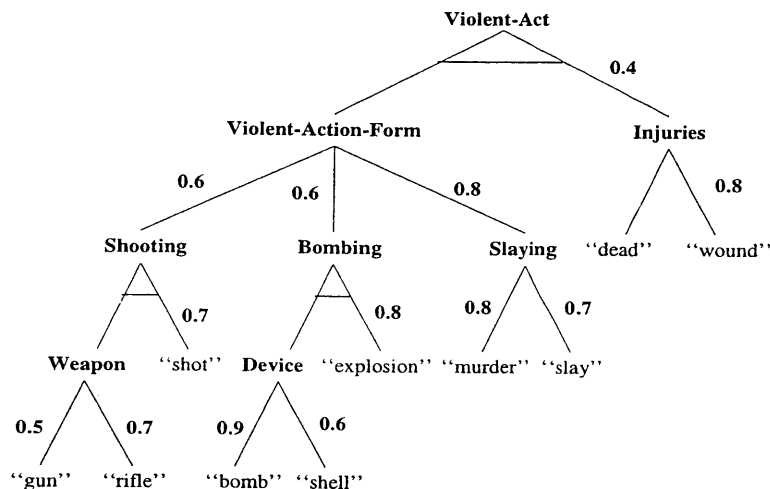


Figure 5.2: Rule-base tree for the concept **Violent-Act**. The leaf nodes are index terms and the internal nodes are concepts. The weights, other than the default weights, are displayed along the edges connecting concepts and components. The default weight is 1.0

characterizes the related concept.

However, there is no static analysis of a rule-base tree, prior to its application against a document database [4]. Therefore, each time the rule-base tree is applied to a document, the whole process must be repeated. It is, therefore, inefficient and the evaluation complexity is exponential in m , where m is the number of nodes in the tree. Accordingly, to enhance the computation, generation of MTSs is proposed [4]. A minimal term set consists of index terms that are necessary to guarantee that the retrieval status value of a concept is larger than 0. In other words, if any of the index terms in the MTS is removed, the RSV becomes 0.

To illustrate the evaluation of an AND/OR tree, we detail an example of the **Violent-Act** concept.

Example 5.1 In Figure 5.2, we show a rule-base tree for the concept **Violent-Act**,

```

(Violent-Action-Form,1.0) & (Injuries,0.4)      ==> Violent-act
(Shooting,0.6) | (Bombing,0.6) | (Slaying,0.8) ==> Violent-Action-Form
(Weapon,1.0) & ("shot",0.7)                    ==> Shooting
(Device,1.0) & ("explosion",0.8)                ==> Bombing
("gun",0.5) | ("rifle",0.7)                    ==> Weapon
("bomb",0.9) | ("shell",0.6)                   ==> Device
("murder",0.8) | ("slay",0.7)                  ==> Slaying
("dead",1.0) | ("wound",0.8)                   ==> Injuries

```

Figure 5.3: Production rules for the concept **Violent-act**.

where the leaf nodes are index terms enclosed by double quotations and the internal nodes are concepts. The weights are displayed along the edges connecting concepts and components. If no weight is specified by the user, a default weight 1.0 is assumed and is not displayed in the tree. The concept **Violent-Act** is first decomposed into two component concepts, **Violent-Action-Form** and **Injuries**, which are related to **Violent-Act** by AND. The AND operator is distinguished from an OR operator by drawing a horizontal line between its branches.

Alternatively, the concept **Violent-act** can be represented as a set of production rules as shown in Figure 5.3. The “&” and “|” operators are used to denote the “AND” and “OR” operators, respectively.

The Retrieval Status Value (RSV) of a document to the concept **Violent-Act** is computed by a bottom-up strategy. For example, if a document contains the words “gun”, “shot”, “bomb”, “slay”, and “dead” and no other words in the rule-base example are referred to, then the index term nodes “gun”, “shot”, “bomb”, “slay”, and “dead”, will receive a weight of 1.0 and all other index term nodes will receive

a weight of 0. Since the concept **Weapon** is composed of two components “gun”, and “rifle”, the weight for “gun” is 1.0 and “rifle” is 0. These components are joined by an OR operator, therefore, the relevancy to the concept **Weapon** is assigned to the *maximum* value of the products of the weights of its components and their corresponding weights connecting its components. Let $w(x)$ be the weight of x in a document and $e(x, y)$ be the weight of the edge between x and y . In this particular case, the result is

$$\begin{aligned} & \max(w(\text{“gun”}) \times e(\text{Weapon}, \text{“gun”}), w(\text{“rifle”}) \times e(\text{Weapon}, \text{“rifle”})) = \\ & \max(1.0 \times 0.5, 0 \times 0.7) = 0.5. \end{aligned}$$

By analogy, the weight for the concept **Shooting** is calculated in the same way. Since the operator for **Shooting** is AND, we use the *minimum* function instead of the maximum function. The result is

$$\begin{aligned} & \min(w(\text{Weapon}) \times e(\text{Shooting}, \text{Weapon}), w(\text{“shot”}) \times e(\text{Shooting}, \text{“shot”})) = \\ & \min(0.5 \times 1.0, 1.0 \times 0.7) = 0.5. \end{aligned}$$

This process is repeated until we get the relevancy (RSV) of the document to the concept **Violent-Act**, which is equal to 0.4. ■

A minimal term set consists of index terms that are necessary to make the retrieval status value of a concept larger than 0. That means if any of the index terms in the MTS is taken out, the RSV would be 0. The corresponding MTSs representation for

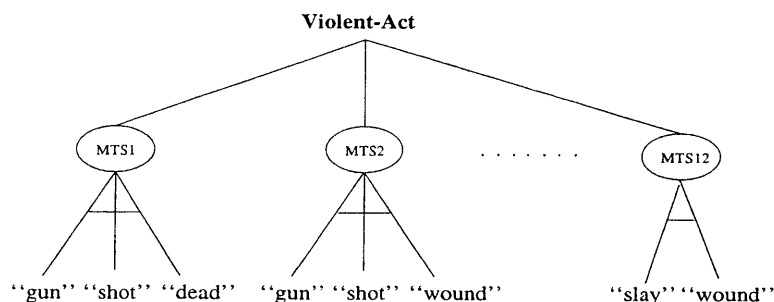


Figure 5.4: Simplified two-level tree for the concept **Violent-act**.

the concept **Violent-Act** is generated following the algorithm developed in Chapter 3. These MTSs along with associated conjunction weights are listed in Table 5.1. Actually, MTSs are all those possible minimal combination of index terms that could produce an RSV larger than 0. They simplify the rule-base tree to a two-level tree as shown in Figure 5.4.

In the process of MTSs evaluation, we can see that if a document satisfies more than one MTS, its RSV will be assigned to the highest RSV among the coexisting MTSs. For example, let us consider two documents: one of which contains 3 MTSs, namely, “MTS7”, “MTS9”, and “MTS12” (that is it contains all and only the index terms in the 3 MTSs); the other document contains only index terms in “MTS4”. Intuitively, since the first document contains more relevant terms, it is more likely to be talking about a violent act than the second one. But the RSV for the first document is 0.32, that is less than the RSV of the second document which is 0.4.

Label	MTS	Weights
MTS1	{“rifle”, “shot”, “dead”}	0.4
MTS2	{“bomb”, “explosion”, “dead”}	0.4
MTS3	{“murder”, “dead”}	0.4
MTS4	{“slay”, “dead”}	0.4
MTS5	{“shell”, “explosion”, “dead”}	0.36
MTS6	{“rifle”, “shot”, “wound”}	0.32
MTS7	{“bomb”, “explosion”, “wound”}	0.32
MTS8	{“shell”, “explosion”, “wound”}	0.32
MTS9	{“murder”, “wound”}	0.32
MTS10	{“slay”, “wound”}	0.32
MTS11	{“gun”, “shot”, “dead”}	0.3
MTS12	{“gun”, “shot”, “wound”}	0.3

Table 5.1: MTSs and RSVs for the concept **Violent-Act**

5.4 Review of GVSM

In information retrieval, it is common to model index terms and documents as vectors in a suitably defined vector space [40]. This approach is usually called Vector Space Model (VSM). In VSM, all the items of interest to the information retrieval system are modeled as elements of a vector space. Let $t_1, t_2, \dots, t_n$ be the terms used to index the documents in a collection. For each term, there is supposed to be a corresponding vector $\vec{t}_i$ in a vector space. These vectors $\{\vec{t}_i \mid i = 1, 2, \dots, n\}$ are considered as the generating set of the subspace and therefore all the items of interests can be represented as a linear combination of $\vec{t}_i$'s.

Let $d_1, d_2, \dots, d_m$ be the documents in a collection and q be a query. Then, we

can consider each d_α and q as vectors in the form of

$$\vec{d}_\alpha = \sum_{i=1}^n a_{\alpha i} \vec{t}_i \quad (5.1)$$

$$\vec{q} = \sum_{j=1}^n q_j \vec{t}_j \quad (5.2)$$

where $a_{\alpha i}$ and q_j are the components of $\vec{d}_\alpha$ and $\vec{q}$ along the direction of the terms $\vec{t}_i$ and $\vec{t}_j$, respectively. Then, the similarity of a document and a query could be acquired by computing the similarity of the document vector and the query vector in the vector space. Generally, the *cosine* similarity function is often used, that is,

$$\vec{d}_\alpha \cdot \vec{q} = \sum_{j=1}^n \sum_{i=1}^n a_{\alpha i} q_j \vec{t}_i \cdot \vec{t}_j \quad (5.3)$$

The documents in a collection can be ranked by the Retrieval Status Values (RSVs) given by similarity function values between the documents and the query. For this purpose we need to know $a_{\alpha i}$'s and $\vec{t}_i \cdot \vec{t}_j$. Note that sometimes we may or may not know the vector representation for $\vec{t}_i$ explicitly. Therefore, earlier researchers made the assumption that $\vec{t}_i$ and $\vec{t}_j$ are orthogonal such that $\vec{t}_i \cdot \vec{t}_j = 0$ if $i \neq j$. Unfortunately such an assumption does not hold in the real world. To solve this problem, the Generalized Vector Space Model (GVSM) is proposed.

The main idea of GVSM is to incorporate the vector representation of elements in a Boolean algebra into a vector space. In this mapping, terms are represented as a linear combination of vectors associated with the atomic expressions (or concepts) that are pairwise orthogonal. Let $t_1, t_2, \dots, t_n$ denote the terms that are used to index the documents in a collection. An *atomic* expression m_k , called a min-

term, in the n literals $t_1, t_2, \dots, t_n$ is a conjunction of the literals where each t_i appears exactly once and is either in complemented or uncomplemented form. That is, $m_k = x_1 \text{ AND } x_2 \text{ AND } \dots \text{ AND } x_n$, where $x_i = t_i$ or $\overline{t_i}$. It is well known that any Boolean expression in the literals $t_1, t_2, \dots, t_n$ can be uniquely expressed as a disjunction of min-terms. Therefore, since the number of all possible min-terms is 2^n and the conjunction of any two different min-terms is always zero (false), we can map the 2^n min-terms into the orthogonal basis of the vector space R^{2^n} as follows:

$$\begin{aligned}
 \vec{m}_1 &= (1, 0, 0, \dots, 0) \\
 \vec{m}_2 &= (0, 1, 0, \dots, 0) \\
 \vec{m}_3 &= (0, 0, 1, \dots, 0) \\
 &\vdots \\
 \vec{m}_{2^n} &= (0, 0, 0, \dots, 1)
 \end{aligned} \tag{5.4}$$

Since each t_i is itself an element of the Boolean algebra generated, t_i can be expressed in its disjunctive normal form:

$$t_i = m_{i_1} \text{ OR } m_{i_2} \text{ OR } \dots \text{ OR } m_{i_r} \tag{5.5}$$

where m_i 's are the min-terms in which t_i is uncomplemented. Let the set of min-terms in Eqn. (5.5) be denoted by $\{m\}^i$. We can now define basis vectors analogous to Eqn. (5.4) and the term t_i can be written in the vector notation as

$$\vec{t}_i = \sum_{m_k \in \{m\}^i} c_{i_k} \vec{m}_k, \quad \text{and} \quad c_{i_k} = \sum_{d_\alpha \in D_{m_k}} w_{\alpha i} \tag{5.6}$$

	t_1	t_2	t_3		
d_1	2	0	1	m_1	1
d_2	1	0	0	m_2	1
d_3	0	1	3	m_3	0
d_4	2	0	0	m_4	1
d_5	2	1	1	m_5	0
				m_6	1
				m_7	0
				m_8	0

(a)

	t_1	t_2	t_3	Basic Vectors
m_1	1	1	1	$\vec{m}_1 = (1, 0, 0, 0, 0, 0, 0, 0)$
m_2	1	1	0	$\vec{m}_2 = (0, 1, 0, 0, 0, 0, 0, 0)$
m_3	1	0	1	$\vec{m}_3 = (0, 0, 1, 0, 0, 0, 0, 0)$
m_4	1	0	0	$\vec{m}_4 = (0, 0, 0, 1, 0, 0, 0, 0)$
m_5	0	1	1	$\vec{m}_5 = (0, 0, 0, 0, 1, 0, 0, 0)$
m_6	0	1	0	$\vec{m}_6 = (0, 0, 0, 0, 0, 1, 0, 0)$
m_7	0	0	1	$\vec{m}_7 = (0, 0, 0, 0, 0, 0, 1, 0)$
m_8	0	0	0	$\vec{m}_8 = (0, 0, 0, 0, 0, 0, 0, 1)$

(b)

Table 5.2: (a) Document-term matrix. (b) Min-terms and basic vectors

where c_{ik} is in unnormalized form, $D_{m_k} = \{d_\alpha \mid d_\alpha \text{ contains all the non-negated terms in } m_k \text{ and also excludes all the negated index terms of } m_k\}$, and $w_{\alpha i}$ is the importance of each term t_i in document d_α . One possible computation of $w_{\alpha i}$ is from the term co-occurrence frequency as suggested by Wong et al. [46].

Example 5.2 Given a set of documents $D = \{d_1, d_2, d_3, d_4, d_5\}$ indexed by the set of terms $T = \{t_1, t_2, t_3\}$. The weights of each term in the documents are given by the matrix shown in Table 5.2(a). There are eight min-terms, generated by the literal t_1, t_2 , and t_3 , as shown in Table 5.2(b). In vector notation, these min-terms can be represented explicitly by the following set of orthonormal basis vectors.

In this example,

$$\begin{aligned}
t_1 &= m_1 \text{ OR } m_2 \text{ OR } m_3 \text{ OR } m_4 & \{m\}^1 &= \{m_1, m_3, m_4\} \\
t_2 &= m_1 \text{ OR } m_2 \text{ OR } m_5 \text{ OR } m_6 & \{m\}^2 &= \{m_1, m_5\} \\
t_3 &= m_1 \text{ OR } m_3 \text{ OR } m_5 \text{ OR } m_7 & \{m\}^3 &= \{m_1, m_3, m_5\}
\end{aligned}$$

Note that for t_1 , the minterms m_1, m_3 , and m_4 appear in some documents, m_2 is not contained in any document, we have, $\{m\}^1 = \{m_1, m_3, m_4\}$. Similarly for $\{m\}^2$ and $\{m\}^3$. Therefore, term vectors $\vec{t}_1, \vec{t}_2$, and $\vec{t}_3$ can be calculated by

$$\begin{aligned}
\vec{t}_1 &= \frac{2\vec{m}_1 + 2\vec{m}_3 + (2+1)\vec{m}_4}{[2^2 + 2^2 + 3^2]^{1/2}} = 0.485\vec{m}_1 + 0.485\vec{m}_3 + 0.728\vec{m}_4 \\
\vec{t}_2 &= \frac{1\vec{m}_1 + 1\vec{m}_5}{[1^2 + 1^2]^{1/2}} = 0.707\vec{m}_1 + 0.707\vec{m}_5 \\
\vec{t}_3 &= \frac{1\vec{m}_1 + 1\vec{m}_3 + 3\vec{m}_5}{[1^2 + 1^2 + 3^2]^{1/2}} = 0.302\vec{m}_1 + 0.302\vec{m}_3 + 0.906\vec{m}_5
\end{aligned}$$

By substituting into Eqn. (5.1), document vectors can be expressed by term vectors as follows.

$$\begin{aligned}
\vec{d}_1 &= 2\vec{t}_1 + \vec{t}_3 = 1.272\vec{m}_1 + 1.272\vec{m}_3 + 1.456\vec{m}_4 + 0.906\vec{m}_5 \\
\vec{d}_2 &= \vec{t}_1 = 0.485\vec{m}_1 + 0.485\vec{m}_3 + 0.728\vec{m}_4 \\
\vec{d}_3 &= \vec{t}_2 + 3\vec{t}_3 = 1.613\vec{m}_1 + 0.906\vec{m}_3 + 3.425\vec{m}_5 \\
\vec{d}_4 &= 2\vec{t}_1 = 0.97\vec{m}_1 + 0.97\vec{m}_3 + 1.456\vec{m}_4 \\
\vec{d}_5 &= 2\vec{t}_1 + \vec{t}_2 + 3\vec{t}_3 = 2.583\vec{m}_1 + 1.876\vec{m}_3 + 1.456\vec{m}_4 + 3.425\vec{m}_5
\end{aligned}$$

Suppose the query $\vec{q} = 2\vec{m}_1 + \vec{m}_3 + \vec{m}_5$, the similarity between $\vec{d}_1$ and $\vec{q}$ is

$$s_1 = \frac{\vec{d}_1 \cdot \vec{q}}{|\vec{d}_1| |\vec{q}|} = \frac{1.272 \times 2 + 1.272 \times 1 + 1.456 \times 0 + 0.906 \times 1}{[1.272^2 + 1.272^2 + 1.456^2 + 0.906^2]^{1/2} [2^2 + 1^2 + 1^2]^{1/2}} = 0.776$$

Similarly, $s_2 = (\vec{d}_2 \cdot \vec{q})/(|\vec{d}_2||\vec{q}|) = 0.594$, $s_3 = (\vec{d}_3 \cdot \vec{q})/(|\vec{d}_3||\vec{q}|) = 0.793$, $s_4 = (\vec{d}_4 \cdot \vec{q})/(|\vec{d}_4||\vec{q}|) = 0.594$, and $s_5 = (\vec{d}_5 \cdot \vec{q})/(|\vec{d}_5||\vec{q}|) = 0.872$. Based on the similarity values ($s_5 > s_3 > s_1 > s_2 \geq s_4$), d_5 will be retrieved first, d_3 second, and so on. ■

5.5 Modeling RUBRIC concepts with GVSM

In RUBRIC, the rule-base tree describes the relationships among concepts. Specification of a concept as a rule-base tree not only helps the user to describe his/her retrieval request more accurately, but also makes the resulting rule-base tree more understandable to other users. However, in order to retrieve a document, RUBRIC requires that the document contains all the index terms in one MTS. It is a strong provision for information retrieval systems, even if each conjunction presents an alternate specification of a query topic.

In an attempt to overcome this problem, in Chapter 4 we extend RUBRIC model in which such provision no longer exists [6]. Our model is based on the extended Boolean retrieval model. Each query term weight in an MTS conjunction is preserved and the extended AND_p operator is applied to the whole conjunction. Then, RSV is computed by applying the extended OR_p operator over all MTSs conjunctions.

In this chapter, we attempt to overcome the same problem described above (i.e., documents should contain all the index terms in an MTS) by considering term correlations as an alternate strategy using GVSM. In GVSM, however, term correlations are well established based on the co-occurrence frequency. GVSM itself does not

provide a way for describing a query at a conceptual level, while the RUBRIC model does. Consequently, it is a natural way to incorporate these two models together. The main issue here is to construct a query vector in GVSM for a concept defined in a rule base tree.

5.5.1 Mapping MTSs to Query Vectors in GVSM

In this section we describe a strategy for mapping MTSs into query vectors in GVSM. For this purpose, we consider *potential weights* of index terms to a given concept and *weighted MTS* for a concept c as an MTS in which each term t_i is assigned the potential weight of t_i to c .

Definition 1 (Absolute Weight) If we have a rule as

$$(t_1, w_1) \text{ } OP \text{ } (t_2, w_2) \text{ } OP \text{ } \cdots \text{ } OP \text{ } (t_n, w_n) \Rightarrow T,$$

where OP is either “|” or “ $\mathcal{E}$ ”, then we say that the *Absolute Weight* of t_i to T is $w_i, 1 \leq i \leq n$.

Definition 2 (Potential Weight) If we have two rules as

$$(t_1, wt_1) \text{ } OP \text{ } (t_2, wt_2) \text{ } OP \text{ } \cdots \text{ } OP \text{ } (t_n, wt_n) \Rightarrow T \text{ and}$$

$$(s_1, ws_1) \text{ } OP \text{ } (s_2, ws_2) \text{ } OP \text{ } \cdots \text{ } OP \text{ } (s_m, ws_m) \Rightarrow S.$$

where OP is “|” or “ $\mathcal{E}$ ”. Also, $s_j, 1 \leq j \leq m$ is the same topic as T (i.e. $s_j = T$), then we say that the *Potential Weight* of t_i to S is $(wt_i * ws_j), 1 \leq i \leq n$.

Definition 3 (Weighted MTS) A *Weighted MTS* for a concept T is an MTS in which each term t_i is assigned the Potential Weight of t_i to T .

Based on Definition 2, to compute the potential weight of an index term, we traverse the unique path in the rule-base tree from the index term up to the root of the given concept and multiply the weights on the path. For example, in Figure 5.2, the potential weight of index term “rifle” to concept **Violent-Act** will be $0.7 * 1.0 * 0.6 * 1.0 = 0.42$. Based on this potential weight, weighted MTS1 in Table 5.1 is computed as $\{(\text{“rifle”}, 0.42), (\text{“shot”}, 0.42), (\text{“dead”}, 0.4)\}$.

To complete the mapping of MTSs into query vectors in GVSM, it is necessary to define vector operations that correspond to AND and OR operators, respectively. In this regard, a model is proposed by Wong et al. [48] where the conjunction operator $\otimes$ and the disjunction operator $\oplus$ are introduced into GVSM that correspond to AND and OR operators, respectively. In this model, if two term vectors $\vec{t}_1$ and $\vec{t}_2$ are given as

$$\vec{t}_1 = \sum_{m_k \in \{m\}^1} c_{1k} \vec{m}_k \quad \text{and} \quad \vec{t}_2 = \sum_{m_k \in \{m\}^2} c_{2k} \vec{m}_k, \quad (5.7)$$

the disjunction operator $\oplus$ and the conjunctive operator $\otimes$ are respectively defined by

$$\vec{t}_1 \oplus \vec{t}_2 = \sum_{m_k \in \{m\}^1 \cup \{m\}^2} \max(c_{1k}, c_{2k}) \vec{m}_k \quad (5.8)$$

$$\vec{t}_1 \otimes \vec{t}_2 = \sum_{m_k \in \{m\}^1 \cap \{m\}^2} \max(c_{1k}, c_{2k}) \vec{m}_k \quad (5.9)$$

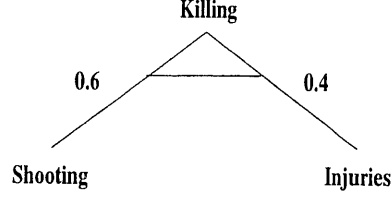


Figure 5.5: An example rule-base tree.

Although these definitions are intuitively acceptable, they are hardly general enough to adopt in our MTSs model. For example, when performing $\vec{t}_1 \oplus t_2$, we have to keep track of the set $\{m\}^1 \cup \{m\}^2$ for the resulting vector. Therefore, we introduce more general definitions for these operators as follows. For $\vec{t}_1 = c_{11}\vec{m}_1 + c_{12}\vec{m}_2 + \cdots + c_{1p}\vec{m}_p$ and $\vec{t}_2 = c_{21}\vec{m}_1 + c_{22}\vec{m}_2 + \cdots + c_{2p}\vec{m}_p$, where $p = 2^n$, we define

$$\vec{t}_1 \oplus \vec{t}_2 = \sum \max(c_{1k}, c_{2k})\vec{m}_k \quad (5.10)$$

$$\vec{t}_1 \otimes \vec{t}_2 = \sum \text{and-max}(c_{1k}, c_{2k})\vec{m}_k \quad (5.11)$$

where $\text{and-max}(x, y)$ is $\max(x, y)$ if x and y are not zero, and zero otherwise.

Having defined the operators $\oplus$ and $\otimes$, the next stage is to introduce a mapping, $h : \{C\} \rightarrow R^{2^n}$, which is used to transform the concepts in the rule tree into vectors.

- (a) $h(t_i, w_i) = w_i * t_i$
- (b) $h((c_1, w_1) \mid (c_2, w_2)) = h(c_1, w_1) \oplus h(c_2, w_2)$
- (c) $h((c_1, w_1) \& (c_2, w_2)) = h(c_1, w_1) \otimes h(c_2, w_2)$

Based on the mapping h defined above, all concepts in the rule tree can be transformed to a linear combination of basis vectors $\vec{m}$'s. It is easy to see that the defini-

	<i>wound</i>	<i>dead</i>	<i>shot</i>	<i>rifle</i>	<i>gun</i>
d_1	2	0	0	1	3
d_2	1	2	2	0	0
d_3	4	0	0	1	1
d_4	0	2	3	4	0
d_5	1	2	2	2	4
d_6	2	1	4	0	1

(a)

	<i>wound</i>	<i>dead</i>	<i>shot</i>	<i>rifle</i>	<i>gun</i>	<i>basis vector</i>
m_0	0	0	0	0	0	$\vec{m}_0$
m_1	0	0	0	0	1	$\vec{m}_1$
m_2	0	0	0	1	0	$\vec{m}_2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
m_{31}	1	1	1	1	1	$\vec{m}_{31}$

(b)

Figure 5.6: (a) Document-term matrix. (b) Min-terms and basic vectors

tions for operators $\oplus$ and $\oplus$ are compatible with the “AND” and “OR” operators in rule-base tree, which assign the document relevancy to a topic (concept) by choosing the and-max and max value of the product of the weights and the relevancy propagated from its children. So, we can calculate the MTSs for a concept in terms of $\vec{m}$ ’s by using the same way as in original rule-base tree in RUBRIC.

5.5.2 A Complete Example

In this subsection, we give an example showing how the extended RUBRIC model works. Suppose we have a rule-base tree as shown in Figure 5.5, where **Shooting**

and **Injuries** are the same concepts defined in Figure 5.2. The weighted MTSs for the concept **Killing** are listed as:

$$\{\text{"gun"}(0.5 * 1.0 * 0.6 = 0.3), \text{"shot"}(0.7 * 0.6 = 0.42), \text{"dead"}(1.0 * 0.4 = 0.4)\}$$

$$\{\text{"gun"}(0.5 * 1.0 * 0.6 = 0.3), \text{"shot"}(0.7 * 0.6 = 0.42), \text{"wound"}(0.8 * 0.4 = 0.32)\}$$

$$\{\text{"rifle"}(0.7 * 1.0 * 0.6 = 0.42), \text{"shot"}(0.7 * 0.6 = 0.42), \text{"dead"}(1.0 * 0.4 = 0.4)\}$$

$$\{\text{"rifle"}(0.7 * 1.0 * 0.6 = 0.42), \text{"shot"}(0.7 * 0.6 = 0.42), \text{"wound"}(0.8 * 0.4 = 0.32)\}$$

Let us assume the Document-term matrix and the basis vectors are given as in Figure 5.6(a) and (b), respectively. Using Eqn. 5.6, the term vectors can be computed as:

$$wound = \vec{t}_1 = 0.92\vec{m}_{19} + 0.15\vec{m}_{28} + 0.30\vec{m}_{29} + 0.15\vec{m}_{31}$$

$$dead = \vec{t}_2 = 0.55\vec{m}_{14} + 0.55\vec{m}_{28} + 0.27\vec{m}_{29} + 0.55\vec{m}_{31}$$

$$shot = \vec{t}_3 = 0.52\vec{m}_{14} + 0.34\vec{m}_{28} + 0.69\vec{m}_{29} + 0.34\vec{m}_{31}$$

$$rifle = \vec{t}_4 = 0.81\vec{m}_{14} + 0.40\vec{m}_{19} + 0.40\vec{m}_{31}$$

$$gun = \vec{t}_5 = 0.69\vec{m}_{19} + 0.17\vec{m}_{29} + 0.69\vec{m}_{31}$$

The coefficients of $\vec{m}$'s are computed from the Document-Term frequency matrix. In particular, the coefficient of $\vec{m}_{19}$ in t_1 is computed as follows. Since the Boolean pattern corresponding to $\vec{m}_{19}$ is 10011, only documents d_1 and d_3 are relevant to this pattern. Therefore, $\vec{m}_{19}$'s unnormalized coefficient is $2 + 4 = 6$. Similarly, the unnormalized coefficients of $\vec{m}_{28}$, $\vec{m}_{29}$, and $\vec{m}_{31}$ are computed as 1, 2, and 1, respectively. Hence, the normalized coefficient of $\vec{m}_{19}$ is $6/\sqrt{6^2 + 1^2 + 2^2 + 1^2} = 0.92$.

Using these term vectors, we can compute document vectors using Eqn. 5.1 as follows:

$$\vec{d}_1 = 2\vec{t}_1 + \vec{t}_4 + 3\vec{t}_5 = 0.81\vec{m}_{14} + 4.31\vec{m}_{19} + 0.30\vec{m}_{28} + 1.11\vec{m}_{29} + 2.77\vec{m}_{31}$$

$$\vec{d}_2 = \vec{t}_1 + 2\vec{t}_2 + 2\vec{t}_3 = 2.14\vec{m}_{14} + 0.92\vec{m}_{19} + 1.94\vec{m}_{28} + 2.22\vec{m}_{29} + 1.93\vec{m}_{31}$$

$$\vec{d}_3 = 4\vec{t}_1 + \vec{t}_4 + \vec{t}_5 = 0.81\vec{m}_{14} + 4.77\vec{m}_{19} + 0.6\vec{m}_{28} + 1.37\vec{m}_{29} + 1.69\vec{m}_{31}$$

$$\vec{d}_4 = 2\vec{t}_2 + 3\vec{t}_3 + 4\vec{t}_4 = 5.9\vec{m}_{14} + 1.6\vec{m}_{19} + 2.12\vec{m}_{28} + 2.61\vec{m}_{29} + 3.72\vec{m}_{31}$$

$$\vec{d}_5 = \vec{t}_1 + 2\vec{t}_2 + 2\vec{t}_3 + 2\vec{t}_4 + 4\vec{t}_5 = 3.76\vec{m}_{14} + 4.48\vec{m}_{19} + 1.93\vec{m}_{28} + 2.90\vec{m}_{29} + 5.49\vec{m}_{31}$$

$$\vec{d}_6 = 2\vec{t}_1 + \vec{t}_2 + 4\vec{t}_3 + \vec{t}_5 = 2.63\vec{m}_{14} + 2.53\vec{m}_{19} + 2.21\vec{m}_{28} + 3.8\vec{m}_{29} + 2.9\vec{m}_{31}$$

For the concept **Killing**, if we choose $\text{MTS11} = \{(\text{"gun"}, 0.3), (\text{"shot"}, 0.42), (\text{"dead"}, 0.4)\}$ as a query, then we can construct the query vector as follows.

$$\begin{aligned} \vec{q} &= [0.3 * (0.69\vec{m}_{19} + 0.17\vec{m}_{29} + 0.69\vec{m}_{31})] \otimes \\ &\quad [0.42 * (0.52\vec{m}_{14} + 0.34\vec{m}_{28} + 0.69\vec{m}_{29} + 0.34\vec{m}_{31})] \otimes \\ &\quad [0.4 * (0.55\vec{m}_{14} + 0.55\vec{m}_{28} + 0.27\vec{m}_{29} + 0.55\vec{m}_{31})] = 0.29\vec{m}_{29} + 0.22\vec{m}_{31} \end{aligned}$$

And if we choose $\text{MTS1} = \{(\text{"rifle"}, 0.42), (\text{"shot"}, 0.42), (\text{"dead"}, 0.4)\}$ and $\text{MTS11} = \{(\text{"gun"}, 0.3), (\text{"shot"}, 0.42), (\text{"dead"}, 0.4)\}$ as a query, then the corresponding vector $\vec{q}$ is computed as follows. $\vec{q} = \vec{q}_1 \oplus \vec{q}_2$, where $\vec{q}_1$ and $\vec{q}_2$ are the query vectors

for MTS1 and MTS11, respectively.

$$\begin{aligned}
\vec{q}_1 &= \{[0.42 * (0.81\vec{m}_{14} + 0.40\vec{m}_{19} + 0.40\vec{m}_{31})] \otimes \\
&\quad [0.42 * (0.52\vec{m}_{14} + 0.34\vec{m}_{28} + 0.69\vec{m}_{29} + 0.34\vec{m}_{31})] \otimes \\
&\quad [0.4 * (0.55\vec{m}_{14} + 0.55\vec{m}_{28} + 0.27\vec{m}_{29} + 0.55\vec{m}_{31})]\} = 0.34\vec{m}_{14} + 0.22\vec{m}_{31} \\
\vec{q}_2 &= 0.29\vec{m}_{29} + 0.22\vec{m}_{31} \\
\vec{q}_1 \oplus \vec{q}_2 &= 0.34\vec{m}_{14} + 0.29\vec{m}_{29} + 0.22\vec{m}_{31}
\end{aligned}$$

5.6 Experiment Design and Results

We describe experiments to validate our approach. We want to show that our approach produces more appropriate rankings as compared to the RUBRIC approach. We choose *Google* search engine [19] as our retrieval engine. The user interface of Google is simple and it uses a conjunction of keywords as queries and returns a set of links to web pages. We use the rule base tree, as shown in Figure 5.2, for our experimental evaluation. We compute 12 weighted MTSs and construct a query for each MTS. For each query, we choose the top 20 links retrieved by Google. Since some links to the web pages might have been changed or moved after being indexed by Google, we must eliminate these links. Finally, we get a collection of 196 documents. The relevance judgments, for evaluation purposes, are determined by looking through each document and choosing those documents related to the concept **Violent-Act**. From the collected documents, we can construct the document-term matrix. Using the matrix, we can compute the term vectors and document vectors. With respect to

Recall	Precision			
	R_1	R_2	ER_1	ER_2
0.1	0.5882	0.5882	0.6250	0.7143
0.2	0.5882	0.5882	0.5714	0.6667
0.3	0.5686	0.5686	0.6170	0.7073
0.4	0.5672	0.5672	0.6032	0.6909
0.5	0.5875	0.5875	0.6184	0.6528
0.6	0.6154	0.6154	0.5895	0.6087
0.7	0.5603	0.5603	0.5328	0.5462
0.8	0.5175	0.5175	0.5175	0.5034
0.9	0.5390	0.5390	0.5155	0.5092
1.0	0.4742	0.4742	0.4769	0.4792

Table 5.3: Recall-Precision using the original RUBRIC approaches (R_1 and R_2) and the extended RUBRIC approaches (ER_1 and ER_2).

each MTS, RSV is computed for each document. Further, these RSVs are combined into a single RSV for each document satisfying more than one MTS. As a result, the user can be given the option of performing the disjunction operation over some or all the MTSs. Thus, our approach for ranking requires post-processing of results from Google.

In our experiment, we select all the MTSs and compute the performance in terms of recall-precision. We conduct experiments using two different disjunction operators, namely $+$ and $\oplus$. We call the former *Extended Rubric version 1* (ER_1) and the latter *Extended Rubric version 2* (ER_2).

In order to compare earlier RUBRIC approaches, we conduct an experiment using two versions of the original RUBRIC model, denoted by R_1 and R_2 , respectively.

In R_1 , we adopt the original idea of RUBRIC. That is, we use the max operator and the min operator for disjunction and conjunction, respectively. In R_2 , we use the same operator (that is, min operator) for conjunction and use a weighted operator for disjunction [29]. The weighted operator is defined as follows. Suppose that document d is satisfied by three weighted MTSs M_1, M_2 , and M_3 with weights w_1, w_2 , and w_3 , respectively. Let w_2 be the maximum value among w_1, w_2 , and w_3 . Then, RSV is given by $RSV = w_1 + 2w_2 + w_3$. The performance of R_1, R_2, ER_1 , and ER_2 are summarized in Table 5.3.

5.7 Conclusion

In this chapter, an extension to RUBRIC model is developed by incorporating Generalized Vector Space model in which the term-term association is considered. Experiments are performed on some variations of the extended RUBRIC model, and the results are compared to the original model based on recall-precision.

From the experiments, we can summarize the following conclusions.

1. The original RUBRIC approach (R_1) gives inaccurate RSVs although it beats ER_i at recall 0.6-0.9.
2. The variation of RUBRIC (R_2) does not improve the accuracy of RSVs.
3. The extended approaches (ER_1 and ER_2) show a better performance than the original RUBRIC approaches (R_1 and R_2) except at recall 0.6-0.9.

4. ER_2 shows a better performance than ER_1 .

Our approach can be extended in several ways. The term weight is not necessarily frequency information. For example, some existing search engines provide weights or scores for retrieved documents. In this case, if a single term is sent as the query to the search engine, the weight or ranking assigned to a document can be treated as the weight of the term in the document. There are other interesting issues related to our approach. For example, based on relevance feedback, we can adjust the weights both in the rule base tree and the document-term matrix. Furthermore, with the help of some knowledge bases, the user can construct the rule base tree more automatically.

Chapter 6

Optimal Queries in Information Filtering

The objective of an information filtering is to classify/categorize documents as they arrive into the system. In this chapter, we investigate an information filtering method based on Steepest Descent Induction Algorithm combined with a two-level preference relation on user ranking. The performance of the proposed algorithm is experimentally evaluated. The experiments are conducted using Reuters-21578 data collection. We investigate the system performance under various settings of parameters. A microaverage breakeven effectiveness measure is used for performance evaluation. The best size of negative data employed in the training set is empirically determined and the effect of R_{norm} factor on the learning process is evaluated. Finally, we demonstrate effectiveness of the proposed method by comparing experimental results to other inductive methods.

6.1 Introduction

Information filtering (IF) is the process that monitors documents as they enter the system and selects only those that match the user query (known as a *profile*). Information retrieval (IR), on the other hand, accepts user needs (requests) as a query and provides the user with a list of documents ordered (ranked) based on the similarity with the user query. Thus, IF makes relevance/non-relevance decisions rather than providing ranked output list as in IR. From this observation, it is easy to see that the filtering process is in fact a *classification* process.

Classification is the assignment of relevance and non-relevance judgments to documents based on similarity between the document and the user query. Classification has been widely studied in statistics and statistical pattern recognition [16], as well as in IR [41].

In IR, the term linear separation is used for retrieval strategies that partition relevant and non-relevant documents into distinct blocks. Linear separation can be simply implemented by constructing optimal queries that rank (list) relevant documents higher than non-relevant. By importing the same notion to IF, we define an optimal query to be a query that most likely classifies a new document as relevant or not relevant to the user. Thus, IF can be thought of as a process of defining an optimal user query.

In this chapter, we propose a framework for the filtering problem based on *Steepest Descent Algorithm* SDA [47], which is an induction learning method, combined with

a two-level preference relation on user ranking. The system is trained on pre-judged documents and an optimal query is formulated following the proposed framework. The reason for choosing SDA is two fold. First, it has been shown by Wong [47] that SDA is a higher order approximation, whereas Rocchios' method (Relevance Feedback in Vector Space Model) is only a first-order approximation and does not necessarily provide a solution vector. Second, to the best of our knowledge, no investigation has been reported for SDA in information filtering, especially when preference relation is involved.

This chapter is organized as follows. In Section 6.2 we give a brief summary of basic concepts followed by effectiveness measures in Section 6.3. Details of the experiment is presented in Section 6.4 and the conclusion is in Section 6.5.

6.2 Preliminaries

A typical information retrieval (IR) systems S can be defined as a 5-tuple, $S = (T, D, Q, V, f)$, where: T is a set of index terms, D is a set of documents, Q is a set of queries, V is a subset of real numbers, and $f : D \times Q \Rightarrow V$ is a retrieval function between a query and a document. Note that a retrieval function typically has the commutative property, i.e., $f(d, q) = f(q, d)$, where $d \in D$ and $q \in Q$. IR systems based on the vector processing model represent documents by vectors of term values of the form $d = (t_1, w_{d_1}; t_2, w_{d_2}; \dots; t_n, w_{d_n})$, where t_i is an index term in d (i.e. $t_i \in T \cap d$) and w_{d_i} is the weight of t_i that reflects relative importance of t_i in d .

Similarly, for a query $q \in Q$, it is represented as $q = (q_1, w_{q_1}; q_2, w_{q_2}, \dots; q_m, w_{q_m})$, where $q_i \in T$ is an index term in q (i.e. $q_i \in T \cap q$) and w_{q_i} is the weight of query term q_i that reflects relative importance of q_i in q .

Our objective is to formulate an optimal query, q_{opt} , that discriminates more preferred documents from less preferred documents. With this objective in mind, we define a preference relation $\succeq$ on a set of partially-ordered documents, Δ , in a retrieval output as follows. For $d, d' \in \Delta$, $d \succeq d'$ is interpreted as d is preferred to or is equally good as d' . It is assumed that the user's preference relation on Δ yields a weak order where the following conditions hold [47]:

$$d \succeq d' \text{ or } d' \succeq d.$$

$$d \succeq d' \text{ and } d' \succeq d'' \Rightarrow d \succeq d''.$$

The essential motivation is that q_{opt} provides an acceptable retrieval output; that is, for all $d, d' \in \Delta$, there exists $q_{opt} \in Q$ such that if $d \succeq d' \Rightarrow f(q_{opt}, d) > f(q_{opt}, d')$.

Given the *user ranking* as a preference relation defined on a set of documents, a system that produces a *system ranking* closer to the user ranking is better than a system that produces a ranking that is further away. To quantify this idea, a performance measure may be derived by using the distance between a user ranking and a system ranking. For such a metric we use the R_{norm} value [10] as defined in the following. Let $(D, \succeq)$ be a document space, where D is a finite set of documents and $\succeq$ be a preference relation as defined above. Let Ψ be some ranking of D given

by a retrieval system. Then R_{norm} is defined as

$$R_{norm}(\Psi) = \frac{1}{2} \left(1 + \frac{S^+ - S^-}{S_{max}^+} \right) \quad (6.1)$$

where S^+ is the number of document pairs where a preferred document is ranked higher than non-preferred document, S^- is the number of document pairs where a non-preferred document is ranked higher than preferred one, and S_{max}^+ is the maximal number of S^+ .

Example 6.1 Consider the ranking $\Psi = (rnrn \mid rnrnrnrn)$ for a two level preference relation, where r stands for a relevant document and n for a non-relevant document. Thus, $S_{max}^+ = 21$, since there are 3 relevant (r 's) and 7 non-relevant (n 's) documents in Ψ ; $S^+ = 10$, since there are ten ordered pairs of (r, n) in Ψ ; and $S^- = 2$, since there are two ordered pairs of (n, r) in Ψ . Hence, $R_{norm} = 0.69$. ■

We formulate q_{opt} inductively by SDA as described in [47]. Let $B = \{b = d - d' : d \succeq d'\}$ be the set of difference vectors in a retrieval output. To obtain q_{opt} from any query $q \in Q$, we solve the following linear inequalities

$$q^T b > 0 \quad (6.2)$$

for all $b \in B$. A steepest descent algorithm is used to find a solution vector q_{opt} for Eq. (6.2). We define total error, which is to be minimized, in our linear system as follows.

$$J(q) = \sum_{b \in \Gamma(q)} -q^T b, \quad (6.3)$$

where

$$\Gamma(q) = \{b = d - d' : q^T b \leq 0\} \subseteq B.$$

We define $J(q) = 0$ if $\Gamma(q) = \emptyset$. (Eq. (6.3) is called *perceptron criterion function* in pattern recognition discipline [16]). It has been shown in [47] that the function $J(q)$ provides a measure of the total error induced by the query vector q . Therefore, minimizing $J(q)$ is equivalent to minimizing the error. The gradient vector is obtained from the derivation of the perceptron criterion function by q (i.e., $\nabla J(q) = -\sum_{b \in \Gamma(q)} b$). It is assumed here that $f(q, d) = q^T d$, which is the dot product, and for $f(q, d) > f(q, d') \Rightarrow q^T d > q^T d' \Rightarrow q^T (d - d') > 0 \Rightarrow f(q, b) > 0$. The steps of the algorithm are as follows.

1. Choose a starting query vector q_0 ; let $k = 0$.
2. Let q_k be a query vector at the start of the $(k+1)th$ iteration; identify the following set of difference vectors:

$$\Gamma(q_k) = \{b = d - d' : d \succeq d' \text{ and } f(q_k, b) \leq 0\};$$

if $\Gamma(q_k) = \emptyset$, $q_{opt} = q_k$ is a solution vector and exit the loop, otherwise,

3. Let

$$q_{k+1} = q_k - \alpha_k \nabla J(q_k)$$

where: α_k is a positive number that sets the step size and assumed to be one

here; $\nabla J(q_k) = -\sum_{b \in \Gamma(q_k)} b$

4. $k = k + 1$; go back to Step (2).

Example 6.2 Consider a set of documents $\Delta = \{d_1, d_2, d_3, d_4\}$ with

$$d_1 = (1, 1, 1)$$

$$d_2 = (0, 1, 2)$$

$$d_3 = (3, 4, 1)$$

$$d_4 = (1, 1, 0)$$

Suppose a user preference relation on Δ is given by:

$$d_1 \succeq d_2$$

$$d_1 \succeq d_4$$

$$d_3 \succeq d_2$$

$$d_3 \succeq d_4$$

For this preference relation, one obtains the following set of difference vectors:

$$B = \{b_{12}, b_{14}, b_{32}, b_{34}\},$$

where

$$b_{12} = (1, 0, -1)$$

$$b_{14} = (0, 0, 1)$$

$$b_{32} = (3, 3, -1)$$

$$b_{34} = (2, 3, 1)$$

By choosing $q_0 = 0$, we have $\Gamma(q_0) = B$. Thus, in the first iteration the query vector q_1 is equal to:

$$\begin{aligned} q_1 &= q_0 + (b_{12} + b_{14} + b_{32} + b_{34}) \\ &= (6, 6, 0) \end{aligned}$$

Using q_1 as query vector, $\Gamma(q_1) = \{b_{14}\}$ is obtained. In the second iteration q_2 becomes:

$$q_2 = q_1 + b_{14} = (6, 6, 1).$$

For query vector q_2 , $\Gamma(q_2) = \emptyset$. Therefore, $q_{opt} = q_2$. It can easily be verified that the retrieval function with $q_{opt} = (6, 6, 1)$ for documents in Δ , namely $\{d_1, d_2, d_3, d_4\}$, satisfies the acceptable ranking provision. ■

Theoretically it is known that SDA terminates only if the set of retrieved documents is linearly separable. Therefore, a practical implementation of the algorithm should guarantee that the algorithm terminates whether or not the retrieved set is linearly separable. As a result, we use a pre-defined iteration number and R_{norm} measure for this purpose. The algorithm is terminated if the iteration number reaches the pre-defined limit or the R_{norm} value of the current query is higher than or equal to some pre-defined value. Within the algorithm loop we continually update the query, q_k , that yields the highest R_{norm} value in order to return q_k as an optimal query.

Predicted Label	Actual Label	
	relevant	non-relevant
Relevant	a	b
Non-relevant	c	d

Table 6.1: Measures of system effectiveness.

6.3 Effectiveness Measures

In order to measure the performance of a classifier,¹ we use text categorization effectiveness measures. There are a number of effectiveness measures employed in evaluating text categorization algorithms. Most of these measures are based on the *contingency table* model. Consider a system that is required to categorize n documents by a query, the result is an outcome of n binary (or two-level) decisions. From these decisions a dichotomous table is constructed as in Table 6.1. Each entry in the table specifies the number of documents with the specified outcome label. For example, a is the number of documents whose predicted and actual labels agree upon being relevant.

In our experiment, the performance is measured based on *precision* and *recall*. Precision is defined as the percentage of relevant documents correctly retrieved among the whole retrieved set, and recall is the percentage of relevant documents retrieved among the whole set of relevant documents stored in the system. For text categorization, $\text{precision} = a/(a + b)$ and $\text{recall} = a/(a + c)$.

¹In this chapter, a classifier is defined as a query. Therefore, we will use query and classifier interchangeably.

We train all queries using the training set of documents for each query then evaluate them using the test set in the corpus. Usually a single composite recall-precision graph is reported reflecting the average performance of all individual queries in the system. Two average effectiveness measures, widely used in the literature, are: *Macroaverage* and *Microaverage* [27]. In information retrieval, Macroaverage is preferred in evaluating query-driven retrieval, while in text categorization Microaverage is preferred. Consider a system with n documents and q queries. Then there are q dichotomous tables, similar to the one in Table 6.1, representing the outcomes of two-level decisions (relevant or nonrelevant) by the filtering system (predicted label) and the user/expert (actual label) when a query is evaluated against all n documents. Macroaverage computes precision and recall separately from the dichotomous tables for each query, and then computes the mean of these values. Microaverage, on the other hand, adds up the q dichotomous tables all together, and then precision and recall are computed.

For the purpose of plotting a single summary figure for recall versus precision values, an adjustable parameter is used to control assignment of documents to profiles (or categories in text categorization). In other words, recall and precision values at different parameter settings are computed to show trade-off between recall and precision. This single summary figure is then used to compute what is called the *breakeven* point, which is the point at which recall is approximately equal to precision [27]. It is possible to use linear interpolation to compute the breakeven point between recall

and precision points.

6.4 The Experiment

In this section, we describe the experimental set up in detail. First, we describe how the Reuters-21578 dataset is parsed and the vocabulary for indexing is constructed. The training method is then discussed and the experimental results are presented.

6.4.1 Reuters-21578 Data Set and Text Representation

To experimentally evaluate the proposed information filtering method we use Reuters-21578 text categorization test collection Distribution 1.0 as our corpus [34]. This collection consists of 21,578 documents which are Reuters newswire stories. The documents of this collection are divided into training and test sets. Each document has five category tags, namely, *EXCHANGES*, *ORGS*, *PEOPLE*, *PLACES*, and *TOPICS*. Each category tag consists of a number of topics that are used for document assignment. We restrict our study to only the *TOPICS* category, which contains 135 topics. To be specific, we used the Modified Apte split that has 9,603 training documents, 3,299 test documents, and 8,676 unused documents.

The type of documents (i.e., either *training*, *testing*, or *unused*) in Modified Apte split is determined as follows: each document in the collection starts with a <REUTER> tag element that has the attributes TOPICS, LEWISSPLIT, CGISPLIT, OLDID, and NEWID.

- If LEWISSPLIT field is equal to “TRAIN” and TOPICS is equal to “YES”

	Training	Test
With at least one topic	7,775	3,019
With no topic	1,828	280
Total	9,603	3,299

Table 6.2: Number of documents in the collection.

then the document is a training document.

- If LEWISSPLIT field is equal to “TEST” and TOPICS is equal to “YES” then the document is a test document.
- Otherwise it is an unused document.

In addition to <REUTER> tag, there are other tags that are used to structure a document. For example, the <TOPICS> and </TOPICS> tags enclose the list of *TOPICS* items, if any, for the document, where each is delimited by the tags <D> and </D>.

Among training documents, there are some documents that are not assigned to any topic. These documents are screened out of the training set, thus reducing the training set to 7,775 documents. There are 135 topics in the *TOPICS* category, with 118 of these topics occurring at least once in the training or test documents². Three of the 118 topics have been assigned to none of the training documents but assigned to only one document in the test set. Each of these three topics will lead to empty

²In the description of the Reuters-21578 read-me file it is stated that the number of topics with one or more occurrences in *TOPICS* category is 120, but we have found only 118. The missing two topics are assigned to unused documents.

rules for their classifiers since they do not have any documents in the training set. During the evaluation process, these documents will automatically add some errors to the system performance. We choose to experiment with these 118 topics despite the effect of these errors in the evaluation process. Table 6.2 shows some statistics about the number of documents in the collection.

We have produced a dictionary of single words excluding numbers as a result of pre-processing the corpus including parsing and tokenizing the text portion of the title as well as the body of both training and unused documents. We have used a universal list of 343 stop words to eliminate functional words from the dictionary [42]. The Porter stemmer algorithm was employed to reduce each of the remaining words to word-stems form [2]. Since any word occurring only a few times is statistically unreliable [36], the words occurring less than five times were eliminated. The remaining words were then sorted in descending order of frequency.

Our filtering framework is based on the Vector Space Model (VSM) in which documents and queries are represented as vectors of weighted terms. Let t_{ik} be an i^{th} term of document with identity k in a collection. One common function employed for computing a document term weight, say w_{ik} , is to multiply term frequency (indicating the frequency of the term in a document) by the inverse document frequency of that term, which can be formulated as $w_{ik} = t_{ik} \times \log(N/n_k)$, where N is the total number of documents in the collection and n_k is the number of documents containing t_{ik} [36]. We use a normalized version of this function (i.e., making magnitudes of term weights

in a document vectors equal to one).

$$w_{ik} = \frac{tf_{ik} \times \log(N/n_k)}{\sqrt{\sum_{j=1}^n [tf_{ij} * \log(N)/n_j]^2}}$$

A document is assigned to a topic by a particular classifier if the *cosine similarity* measure between this document and the query of this classifier is greater than or equal to an externally supplied *threshold* value — previously in Section 6.3, it is called an adjustable parameter. If a document $D_i = (t_{1i}, \dots, q_{ni})$ and a query $Q_j = (q_{1j}, \dots, q_{nj})$ are n -dimensional vectors, the similarity measure between D_i and Q_j is computed as:

$$sim(D_i, Q_j) = D_i \cdot Q_j = \sum_{k=1}^n t_{ki} q_{kj}$$

where D_i and Q_j are assumed normalized vectors.

6.4.2 Training

In contrast to information retrieval systems, in text categorization systems, we have neither a retrieval output nor a user query. Instead, we have a number of topics and for each topic the document collection is partitioned into training and test cases. The training set contains only positive examples of a topic. In this sense, the training set is not a counterpart of the retrieval output due to the fact that we do not have negative examples. We can, however, construct a training set for a topic that consists of positive and negative examples, under the plausible assumption that any document considered as a positive example for the other topics and not in the set of positive examples of the topic at hand is a candidate for being a negative example of this topic.

Name	Train	Test	Name	Train	Test
earn	2877	1087	ship	197	89
acq	1650	719	corn	182	56
money-fx	538	179	money-supply	140	34
grain	433	149	dlr	131	44
crude	389	189	sugar	126	36
trade	369	118	oilseed	124	47
interest	347	131	coffee	111	28
wheat	212	71	gnp	101	35

Table 6.3: Top 16 topics with more than 100 positive examples.

Notice that in *characterization* of a topic it is enough to extract common features of the positive examples, but in categorization it is necessary to find discriminative features where the problem becomes a non-exclusive multi-categorization. Hence, it is essential to set up a training set such that it consists of positive and negative examples. In the following we address this issue and then determine empirically the size and quality of the training size used for the experiment.

The maximum number of positive examples per topic is 2877 and the average is 84. The size and especially the quality of the training set is an important issue in generating an induction rule set. In an information routing study [41], the learning method was not applied to the full training set but rather to the set of documents in the local region for each query. The local region for a query was defined as the 2000 documents nearest to the query, where similarity was measured using the inner product score to the query expansion of the initial query. Also, in [7] the rules for text categorization were obtained by creating local dictionaries for each classification

topic. Only single words found in documents on the given topic were entered in the local dictionary.

In our experiment, the training set for each topic consists of all positive examples while the negative data is sampled from other topics. The reason for including the entire set of positive examples is that SDA is an enhanced version of a relevance feedback algorithm and thus a larger number of positive examples makes the algorithm produce more efficient induction rules. Additionally, the result published by Dumais et al. [17] for the Reuters-21578 data shows that with respect to microaveraged score of the SVM (Support Vector Machine) over multiple random samples of training sets for the top 10 categories with varying sample size, but keeping size of negative data the same, performance of the SVM was degraded from 92% to 72.6% while the size was reduced from the whole training set down to 1%. Another important finding reported in that study shows that performance of the SVM becomes somewhat unstable when a category has fewer than 5 positive training examples. Here we have investigated the size of training set from a different perspective and tried to estimate the best size for the negative data in proportion to positive ones, which is described in the remaining part of this section.

To estimate the best size for negative data in the training set, we have experimented with the top 16 topics of the Reuters-21578 data as shown in Table 6.3 in which training and testing sizes of positive data are listed. The reason for experimenting on these topics is that they have abundance of positive examples. We have

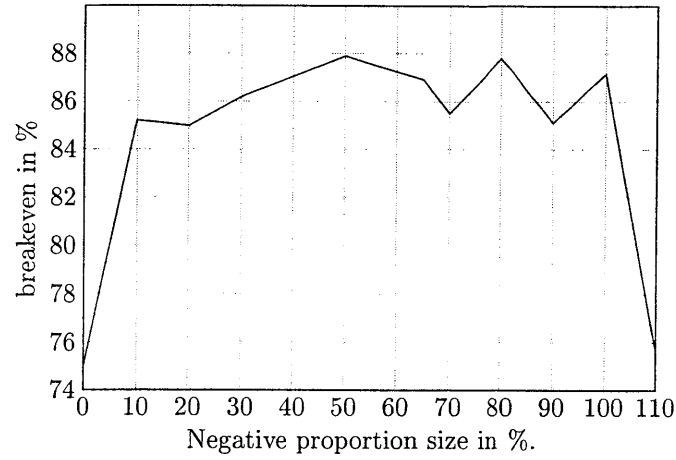


Figure 6.1: Performance of the top 16 topics at various negative-to-positive percentages

trained each of these topics as follows:

1. Consider all positive data,
2. take a $\phi\%$ of the positive data as the sample size of the negative data,
3. apply SDA on this training set and compute the breakeven point,
4. repeat the above steps but in Step 2 select a different negative data sample.

This process is continued until all negative data is exhausted. Figure 6.1 shows the performance for $\phi = 10, 20, \dots, 110$. The initial query of SDA is first set to the mean of the positive examples and for subsequent iterations the initial query is simply set to the query obtained in the previous iteration. We set R_{norm} value to its maximum value (i.e. 1.0) and assert on maximum of 150 iterations in case the value of R_{norm} is not reached.

As indicated in Figure 6.1, the quality of the induction is effectively enhanced as the proportion of negative data to positive data is increased. However, when it becomes more than the size of the positive data the performance is decreased. Besides, the maximum point is reached (i.e., the best performance in the absolute sense) when the proportion of negative data becomes 50% or 80%. It is worth stating that the performance is abruptly degraded when the size of negative data exceeds that of positive data. For the concern of obtaining the best quality of induction, we fixed the size of negative sample to 50% of the positive set. For example, for ‘grain’ topic in Table 6.3 we consider all the 433 training examples as positive data and 216 as negative data sampled from other topics.

6.4.3 Results

In our preliminary experiments in Subsection 6.4.2, we assumed that R_{norm} is set to its maximum value and assert on a maximum of 150 iterations. The choice of the R_{norm} value used in terminating criteria of the SDA algorithm is important in the learning process. This is because there is an application dependent trade off between quality of queries and processing overhead. A higher value may produce a better query on the expense of increasing the processing overhead, whereas a lower value may produce a less efficient query on the advantage of decreasing the processing overhead. Figure 6.2 shows the performance trade offs on various values of R_{norm} on the top 10 topics. As the value of R_{norm} decreases, the system performance also decreases until a point where the query produced for subsequent values of R_{norm} remains unchanged. This

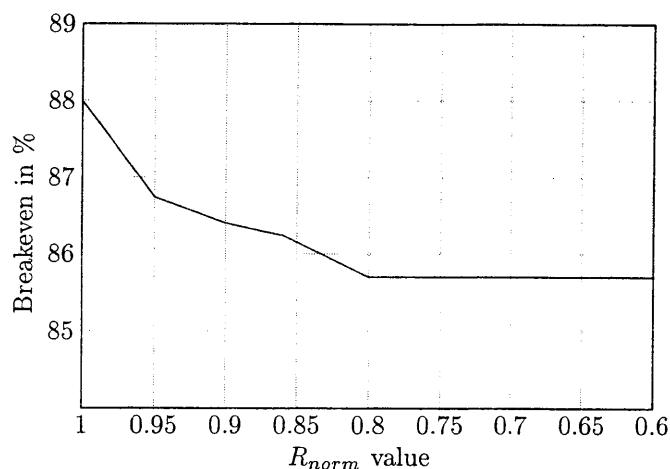


Figure 6.2: R_{norm} value versus breakeven on the top 10 topics.

is because on the first run, the algorithm produces a query that gives a ranking of a higher R_{norm} value regardless of the supplied R_{norm} value.

Finally, after all the necessary parameters are fixed, we train the system on all of the 118 topics. The average precision/recall graph of these topics is shown in Figure 6.3. The graph is magnified and plotted around the area where recall and precision are approximately equal (i.e. breakeven point.) From the graph, the breakeven point is approximately 81.28%.

As a comparative study, Table 6.4 presents results of SDA and other five inductive algorithms that were recently experimented on Reuters-21578 dataset [17]. *Findsim* method is a variant of Rocchio's method for relevance feedback. The weight of each term is the average (or centroid) of its weight in positive instances of the topic. We may compare SDA with this method since it is a first-order approximation. The list of names such as *NBayes*, *BayesNets*, *Trees*, and *SVM* in Table 6.4 stand for Naive

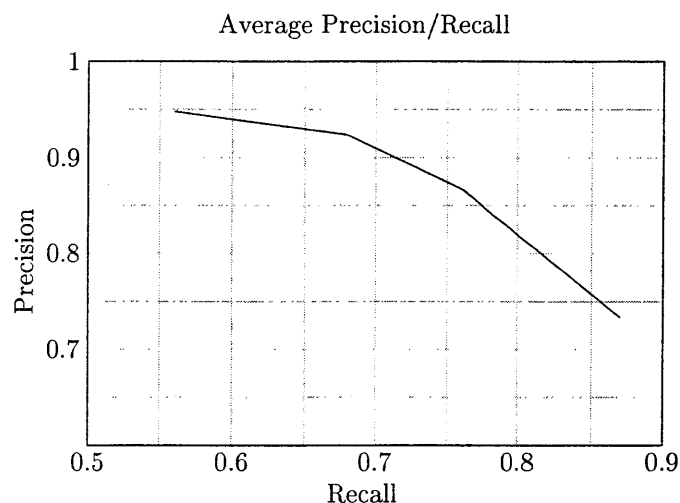


Figure 6.3: Average Precision/Recall on all 118 topics. Breakeven point is approximately 81.28%.

Bayes, Bayes Nets, Decision Trees, and Linear Support Vector Machines methods, respectively. For further details of these methods the reader is referred to [17].

SDA outperforms Findsim, NBayes, and BayesNet, and is almost as good as Decision Trees method. It is however outperformed by the Linear SVM method due to the fact that relevance feedback methods (including SDA) require as large a size of positive data as possible for drifting the query towards the solution region. Therefore, for topics with a small number of positive examples, which represent the majority of the topics in the Reuters-21578 data, the optimal query, which is close to the solution region is hard to find and the performance of the algorithm SDA becomes the same as the Findsim method on these topics. Nevertheless, on average it outperforms the Findsim method by a significant margin, which upholds the plausible fact that higher-order approximation methods, such as SDA, outperform counterpart

Topic	Findsim	NBayes	BayesNets	Trees	SDA	SVM
earn	92.9%	95.9%	95.8%	97.8%	98.5%	98.0%
acq	64.7%	87.8%	88.3%	89.7%	95.9%	93.6%
money-fx	46.7%	56.6%	58.8%	66.2%	78.4%	74.5%
grain	67.5%	78.8%	81.4%	85.0%	90.6%	94.6%
crude	70.1%	79.5%	79.6%	85.0%	86.5%	88.9%
trade	65.1%	63.9%	69.0%	72.5%	76.06%	75.9%
interest	63.4%	64.9%	71.3%	67.1%	77.29%	77.7%
wheat	68.9%	69.7%	82.7%	92.5%	82.2%	91.9%
ship	49.2%	85.4%	84.4%	74.2%	88.1%	85.6%
corn	48.2%	65.3%	76.4%	91.8%	82.36%	90.3%
Avg. Top 10	64.6%	81.5%	85.0%	88.4%	87.99%	92.0%
Avg All Cat.	61.7%	75.2%	80.0%	N/A	81.28%	87.0%

Table 6.4: Comparing results with other five inductive algorithms. Breakeven is computed on top 10 topics and on overall 118 topics.

first-order approximation methods, such as Findsim.

6.5 Conclusion

In this chapter, we study the performance of the steepest descent algorithm in information filtering systems. Information filtering resembles in most parts to classification/text categorization processes. We used the publicly available Reuters-21578 data collection, which is a well known dataset in text categorization and machine learning experiments. The size and quality of the training set is an important issue in generating induction rules. In our preliminary experiments on Reuters-21578 dataset, we find that as the proportion of negative to positive data is 50% or 80% the quality of induction is maximum. In a preliminary experiment on the R_{norm} factor and its

affect on the system performance, we observed that as R_{norm} is decreased the quality of the induced rule is decreased and hence the performance degrades. There is a well defined trade off between quality of induction and the computational efficiency. As a result, as R_{norm} decreases, SDA takes fewer iterations in producing the rule at the expense of possibly not having an optimal rule.

The average precision/recall of the system performance is computed over all 118 topics. The breakeven point is 81.28%. As SDA is compared to other inductive algorithms experimented on Reuters-21578, it outperforms Findsim, NBayes, and BayesNet methods. However, it competes with Decision Tree methods and is outperformed by the Linear SVM method.

Chapter 7

Summary and Future Work

In this chapter we present concluding remarks and discuss directions for future work. In Section 7.1, we briefly summarize the main results of this thesis. In Section 7.2, we present future research directions related to our work.

7.1 Concluding Remarks

The fundamental objective of an information filtering system is to efficiently classify/categorize documents arriving to the system. To achieve this goal an IF model must solve the problem of minimizing the terminological gap between queries represented by users and documents issued by authors. In this thesis we investigate the problem of developing an efficient information filtering system. We develop a conceptual query model that is close to the users' need and investigate an induction method for information filtering. The proposed conceptual model involves preprocessing of the rule base to generate Minimal Term Sets (MTSs). This model is then extended in two directions:

- incorporating p -Norm model into evaluation of MTSs, and

- integrating the Generalized Vector Space Model (GVSM) with MTSs.

We propose a new approach for bridging the terminological gap between users' query and authors' documents. This approach involves the preprocessing of the rule base to generate MTSs. In Chapter 3, we develop two algorithms, the first is *Generate_MTS* for generating MTSs queries and the second is *MTS_Evaluation(QueryTopic,n)* for evaluating MTS queries when generalized retrieval functions are involved. We compare our approach with an existing approach based on the AND/OR tree. We show that the computational complexity of our approach is polynomial whereas that of the AND/OR tree was exponential. Moreover, the MTSs model allows users to share rule-bases and thus facilitates efficient rewriting of queries.

On one hand, MTSs model is extended and enhanced by introducing document term weights with the extended Boolean Model. We adopt p -Norm model, as our computational model, into MTSs evaluation with the parameter p set to ∞ . Furthermore, we integrate the MTSs model with the generalized vector space model. In this model, term-term association is well established, and thus extending the RUBRIC model. Experiments are performed on some variations of the extended RUBRIC model, and the results are compared to the original RUBRIC model based on recall-precision.

We then develop an information filtering method based on Steepest Descent Induction Algorithm combined with a two-level preference relation on user ranking. The performance of the proposed algorithm is experimentally evaluated and com-

pared with other induction methods. The results show that the Steepest Descent Algorithm outperforms Findsim, NBayes, and BayesNet methods. However, its performance is comparable to that of Decision Tree method but it is outperformed by the Linear SVM method.

7.2 Future Work

One of the main problems in RUBRIC is the construction of production rules represented as AND/OR trees. Currently, these rules are manually constructed by experts. Therefore, one possible direction of research related to RUBRIC is to develop methods for automatically constructing rules from typical thesauri. The advantage of automating rule construction is to remove the user burden from defining concepts by themselves and to simplify the use of retrieval systems.

In Chapter 3, the algorithm *Generate_MTS* produces MTS conjunctions without considering storage limitations. The algorithm may be enhanced and extended to store and process only significant conjunctions rather than archiving all. A function may be devised that determines which conjunction needs to be archived.

It is obvious that information filtering is a dynamic process which keeps updating the query/classifier incrementally based on incoming documents. In the future, we intend to investigate the application of incremental learning to information filtering. Specifically, we want to determine how much information from past relevance judgments must be archived for a query in order to maintain a “high quality” of query

optimization while keeping space requirements as low as possible. It is also interesting to study what kind of eviction policy is useful when adding a new document to a full cache.

In Chapter 6, we considered a two-level preference relation in the induction process. However, the preference relation can be extended to a multi-level preference relation. One direction, we believe to be very interesting, is to investigate a multi-level preference relation assuming three level preferences (namely positive, boundary, and negative).

Bibliography

- [1] I.J. Aalbersberg, *Incremental relevance feedback*, Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1992, pp. 11–22.
- [2] Porter algorithm source code, <http://ils.unc.edu/keyes/java/porter/index.html>.
- [3] J. Allan, *Incremental relevance feedback for information filtering*, Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1996, pp. 270–278.
- [4] A. Alsaffar, J.S. Deogun, V.V. Raghavan, and H. Sever, *Concept based retrieval by minimal term sets*, Foundations of Intelligent Systems: Eleventh International Symposium on Methodologies for Intelligent Systems, ISMIS'99 proceedings (Warsaw, Poland) (Z. W. Ras and A. Skowron, eds.), Springer Verlag, June, 1999, pp. 114–122.
- [5] A. Alsaffar, J.S. Deogun, and H. Sever, *Optimal queries in information filtering*, Foundations of Intelligent Systems: The 12th International Symposium on Methodologies for Intelligent Systems, ISMIS'2000 proceedings, Zbigniew Ras,

- and Setsuo Ohsuga (eds.), *Lecture Notes in Artificial Intelligence* (Charlotte, North Carolina), vol. 1932, Springer Verlag, October, 2000, pp. 435–443.
- [6] A.H. Alsaffar, J.S. Deogun, V.V. Raghavan, , and H. Sever, *Enhancing concept-based retrieval based on minimal term sets*, *Journal of Intelligent Information Systems* **14** (March-June, 2000), no. 2/3, 155–173.
- [7] C. Apte, F. Damerau, and S. M. Weiss, *Automated learning of decision rules for text categorization*, *ACM Transactions on Information Systems* **12** (July, 1994), no. 3, 233–251.
- [8] N.J. Belkin and W.B. Croft, *Information filtering and information retrieval: Two sides of the same coin?*, *Communications of the ACM* **35** (1992), no. 12, 29–38.
- [9] K.D. Bollacker, S. Lawrence, and C.L. Giles, *CiteSeer: an autonomous web agent for automatic retrieval and identification of interesting publications*, *Proceedings of the Second Int'l Conference on Autonomous Agents*, ACM (New York, Minneapolis, St. Paul), May 1998, pp. 116–123.
- [10] P. Bollman and S.K.M. Wong, *Adaptive linear information retrieval models*, *Proceedings of the 10th International ACM SIGIR Conference on Research and Development in Information Retrieval* (New Orleans, LA), June 1987, pp. 157–163.

- [18] William B. Frakes and Ricardo Baeza-Yates, *Information retrieval: Data structures & Algorithms*, Prentice Hall, Englewood Cliffs, New Jersey 07632, 1992.
- [19] Google Search Engine, <http://www.google.com>.
- [20] M.N. Huhns and M.P. Singh, *Readings in agents*, Morgan Kaufman, San Francisco, California, 1998.
- [21] D. Hull, *Improving text retrieval for the routing problem using latent semantic indexing*, Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1994, pp. 282–289.
- [22] M. Kim, A. Alsaffar, J.S. Deogun, and V.V. Raghavan, *On modeling of concept based retrieval in generalized vector spaces*, Foundations of Intelligent Systems: The 12th International Symposium on Methodologies for Intelligent Systems, ISMIS'2000 proceedings, Zbigniew Ras, and Setsuo Ohsuga (eds.), Lecture Notes in Artificial Intelligence (Charlotte, North Carolina), vol. 1932, Springer Verlag, October, 2000, pp. 453–462.
- [23] G. Kowalski, *Information retrieval systems. Theory and Implementation*, Kluwer Academic, Norwell, Massachusetts, 1997.
- [24] B. Krulwich, *Automating the internet: Agents as user surrogates*, IEEE Trans. on Internet Computing **1** (1997), no. 4, 34–38.

- [25] D. D. Lewis, *Evaluating and optimizing autonomous text classification systems*, Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1995, pp. 246–254.
- [26] D.D. Lewis, *An evaluation of phrasal and clustered representations on text categorization task*, Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1992, pp. 37–50.
- [27] D.D. Lewis, *Evaluating text categorization*, Proceedings of Speech and Natural Language Workshop, February 1991, pp. 312–318.
- [28] D.D. Lewis and W.A. Gale, *A sequential algorithm for training text classifiers*, Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1994, pp. 11–22.
- [29] B.P. McCune, R.M. Tong, J.S. Dean, and D.G. Shapiro, *RUBRIC: A system for rule-based information retrieval*, IEEE Trans. on Software Engineering **11** (1985), no. 9, 939–944.
- [30] T.M. Mitchell, *Machine learning*, McGraw Hill, 1997.
- [31] T. Noreault, M. Koll, and M.J. McGill, *Automatic ranked output from boolean searches in SIRE*, Journal of the American Society in Information Science **32** (1981), no. 4, 275–279.

- [32] V.V. Raghavan and H. Sever, *On the reuse of past optimal queries*, Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1995, pp. 344–350.
- [33] P. Resnik, *Using information content to evaluate semantic similarity in a taxonomy*, Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1995, pp. 448–453.
- [34] Reuters-21578 collection, <http://www.research.att.com/~lewis>.
- [35] G. Salton, *Automatic term class construction using relevance – A summary of work in automatic pseudoclassification*, Information Processing and Management **16** (1980), no. 1, 1–15.
- [36] G. Salton, *Automatic text processing. The transformation and retrieval of information by computer*, Addison-Wesely Publishing Co., Reading, Massachusetts, 1989.
- [37] G. Salton, J. Allan, and C. Buckley, *Automatic structuring and retrieval of large text files*, Communications of the ACM **37** (1994), no. 2, 97–108.
- [38] G. Salton and C. Buckley, *Improving retrieval performance by relevance feedback*, Journal of the American Society for Information Science **41** (1990), no. 4, 288–297.

- [39] G. Salton, E.A. Fox, and H. Wu, *Extended boolean information retrieval*, Communications of the ACM **26** (1983), no. 11, 1022–1036.
- [40] G. Salton and M.J. McGill, *Introduction to modern information retrieval*, McGraw Hill, New York, 1983.
- [41] H. Schütze, D.A. Hull, and J.O. Pederson, *A comparison of classifiers and document representations for the routing problem*, Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1995, pp. 229–237.
- [42] Stop list, http://www.iiasa.ac.at/docs/R_Library/libsrchs.html.
- [43] J.T. Tou and R.C. Gonzalez, *Pattern recognition principles*, Addison-Wesely Publishing Company, 1974.
- [44] V.V.Raghavan and C.T. Yu, *Experiments on the determination of the relationships between terms*, ACM Transactions on Database Systems **4** (1979), no. 2, 240–260.
- [45] S.M. Weiss and C.A. Kulikowski, *Computer systems that learn*, Morgan Kaufman, San Mateo, California, 1991.
- [46] S. K. M. Wong, W. Ziarko, V. V. Raghavan, and P. C. N. Wong, *On modeling of information retrieval concepts in vector spaces*, ACM Transactions on Database Systems **12** (June, 1987), no. 2, 299–321.

- [47] S.K.M. Wong and Y.Y. Yao, *Query formulation in linear retrieval models*, Journal of the American Society for Information Science **41** (1990), no. 5, 334–341.
- [48] S.K.M. Wong, W. Ziarko, V.V. Raghavan, and P.C.N. Wong, *Extended boolean query processing in the generalized vector space model*, Information Systems **14** (1989), no. 1, 47–63.
- [49] S.K.M. Wong, W. Ziarko, and P.C.N. Wong, *Generalized vector space model on information retrieval*, Proceedings of the 8th Annual International ACM-SIGIR Conference (New York), June, 1985, pp. 18–25.
- [50] T.W. Yan and H.G. Molina, *Index structures for information filtering under the vector space model*, Proceedings of the International Conference on Data Engineering, 1994, pp. 337–347.
- [51] Y.Y. Yao, *Measuring retrieval effectiveness based on user preference of documents*, Journal of the American Society for Information Science **46** (1995), no. 2, 133–145.
- [52] Y. Zhang, V.V. Raghavan, and J.S. Deogun, *An object-oriented modeling of the history of optimal retrievals*, Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (Chicago, Illinois), 1991, pp. 241–250.