1-1-2003

Thesaurus-aided learning for rule-based categorization of Ocr texts

Jeffrey Scott Coombs
University of Nevada, Las Vegas

THESAURUS AIDED LEARNING FOR RULE-BASED CATEGORIZATION OF OCR TEXTS

by

Jeffrey Scott Coombs

Bachelor of Science
University of Nevada, Las Vegas
2000

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science Degree
Department of Computer Science
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
May 2002

INFORMATION TO USERS

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

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

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

UNLV
UNIVERSITY OF NEVADA LAS VEGAS

Thesis Approval

The Graduate College
University of Nevada, Las Vegas

March 17, 2002

The Thesis prepared by

Jeffrey Scott Coombs

Entitled

Thesaurus Aided Learning for Rule-Based Categorization of OCR Texts

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

Examination Committee Chair

Dean of the Graduate College

Examination Committee Member

Examination Committee Member

Graduate College Faculty Representative

ABSTRACT

Thesaurus Aided Learning for Rule-Based Categorization of OCR Texts

by

Jeffrey Scott Coombs

Dr. Kazem Tagva, Examination Committee Chair
Professor of Computer Science
University of Las Vegas, Nevada

The question posed in this thesis is whether the effectiveness of the rule-based approach to automatic text categorization on OCR collections can be improved by using domain-specific thesauri. A rule-based categorizer was constructed consisting of a C++ program called C-KANT which consults documents and creates a program which can be executed by the CLIPS expert system shell. A series of tests using domain-specific thesauri revealed that a query expansion approach to rule-based automatic text categorization using domain-dependent thesauri will not improve the categorization of OCR texts. Although some improvement to categorization could be made using rules over a mixture of thesauri, the improvements were not significantly large.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

Thanks to Steve Lumos, Doina Bein, and Emma Pease for help with LaTeX. I am grateful to Steve Lumos, Allen Condit, and Julie Borsack and the "Geologists" in the trailer for constructing the test collections Small-DOE and Big-DOE.

I would like to thank all of the staff at the Information Science Research Institute for their discussions about this work as it progressed. I especially wish to thank Dr. Ashok Singh for introducing me to statistical analysis, Dr. John Minor for introducing me to artificial intelligence and machine learning, and most of all to Dr. Kazem Taghva for introducing me to the field of information science.

CHAPTER 1

INTRODUCTION

Automatic text categorization is the assignment of a document to a predefined category or class by a computer program. Although automatic text categorization has been studied extensively [47, 60], the problems which arise for categorizing documents which have been scanned using Optical Character Recognition (OCR) technology have not. Following up on previous work by the Information Science Research Insitute (ISRI) which studied the behavior of a statistical categorizer called BOW [31] on OCR texts [53, 51], this thesis looks into the effectiveness of a rule-based approach to the text categorization of OCR documents.

In particular the main question for this thesis is whether a domain-specific, manually constructed thesaurus increases the effectiveness of rule-based categorization for OCR texts. Thesauri have been used to aid document retrieval systems since the 1970's [48], and there have been mixed results using thesauri to aid retrieval on OCR collections [8]. There have also been studies on thesaurus aided rule-based categorizers for non-OCR text collections [22], but the effectiveness of rule-based categorizers on OCR text collections has not been studied.

A rule-based automatic categorizer was constructed using the algorithms IREP [13] and RIPPER [5]. The rule-building part of the categorizer can be viewed as the knowledge acquisition component to an expert system [14]. As such, it was relatively straightforward to construct the rule-builder in C++ to output a program which can be run in a standard expert system shell. In this case, the "off-the-shelf" program

CLIPS [46] is used to apply the rules constructed by the rule-builder to documents formatted as "facts" for the expert system.

The Department of Energy (DOE) donated a large collection of OCR documents from its Licensing Support Network (LSN) to the Information Science Research Institute (ISRI) at the University of Nevada, Las Vegas [52]. Contractors for the DOE created a thesaurus of terms from documents in the LSN collection which was also made available to ISRI. The documents in the LSN collection were prepared by LSN participants, and the thesaurus accordingly contains terms specific to academic disciplines relevant to LSN, such as geology and nuclear physics.

Human experts working for ISRI manually created two collections from the LSN documents. The human experts separated documents into categories developed by the Nuclear Regulatory Commission for categorizing documents about the Yucca Mountain Depository. These guidelines are referred to as the *3.69 Topical Guidelines* [6]. The first collection contains documents from only seven categories of the Topical Guidelines and is therefore called "Small-DOE." It was used for setting parameters for the rule-building software and for comparison with other categorizers tested on the same collection. The second collection was much larger and covered most of the 3.69 Topical Guidelines. This collection, called "Big-DOE" was the collection used for testing. These two collections are described in greater detail in chapter 4.

The automatic categorizer was tested against the judgments of the human experts who categorized the two collections. Both Big-DOE and Small-DOE were divided into training and test sets. After learning rules from the training set, the categorizer's judgments were compared with those of the human experts on the test sets. Tests were made with rules learned from texts enhanced with thesaurus information, and these were compared with rules learned without thesaurus information. The standard measures *precision*, *recall*, and F_β are used to evaluate the results [24].

Chapter 2 provides an introduction to the concept of automatic text categorization and surveys the various approaches to the problem. Chapter 3 covers the concept of rule-based text categorization and the approach taken to construct such a categorizer for this project. The categorizer used is essentially a knowledge acquisition engine (called C-KANT) which creates rules for the well-known expert system shell CLIPS. Since C-KANT uses the IREP and RIPPER algorithms to generate rules, these algorithms are presented in chapter 3 as well.

In chapter 4 the standard evaluation measures for text categorization are defined. This chapter also describes the collections studied and the nature of the thesauri used. Chapter 5 presents the overall results of the tests as well as category by category results. It will be shown that thesaurus aided learning does not improve the categorizing ability of a rule-based categorizer. Finally, chapter 6 states the conclusion of the study and offers prospects for further research.

CHAPTER 2

AUTOMATIC TEXT CATEGORIZATION

The aim of this research is to determine if augmenting a rule-based automatic text categorizer with domain-specific thesauri will improve its performance when dealing with OCR processed text. In this chapter, the concept of automatic text categorization is defined and some of the main approaches taken to categorization are described. This description will provide a background for understanding the specifics of the rule-based approach in chapter 3.

Definition

With the invention of electronic text it has become infeasible for humans alone to categorize the large number of documents generated. As we will see in the next section, many attempts have been made to create programs for computers which automatically place documents in their proper groupings with little human intervention.

The basic idea of text categorization is to take a given document and put it in its proper place within some organizing conceptual system. An example of text categorization would be the Library of Congress Number *QA 76.9 D33154* for William Frakes' book Information Retrieval. The symbols *QA* tell us it is a book on mathematics, and the number *76.9* indicates that the subject matter belongs to "electronic computers, computer science" [38]. Online directories of Web documents such as *Yahoo!* and the *DMOZ Open Directory Project* (the directory supplement to the Google search engine) are also examples of text categorization. All of the texts in these three collections are still manually classified by human editors, which is one of the reasons

	d_1	...	d_j	...	d_n
c_1	$a_{1,1}$	...	$a_{1,j}$	...	$a_{1,n}$
...	...	...	...	...	...
c_i	$a_{i,1}$	...	$a_{i,j}$	...	$a_{i,n}$
...	...	...	...	...	...
c_m	$a_{m,1}$	...	$a_{m,j}$	...	$a_{m,n}$

Figure 2.1: Decision Matrix

they are slow to update and are often incomplete [42, 18]. Automatic text categorizers are currently being used for document filtering, automatic text indexing with controlled vocabulary, automated meta-data generation, word sense disambiguation, the construction of Web directories, the automated routing of technical documents, and sorting e-mail [47, 28, 22].

Following Sebastiani [47] we may define text categorization more precisely in terms of an $m \times n$ *decision matrix* such as that depicted in figure 2.1. The object of text categorization is to assign a value $a_{i,j} \in \{0, 1\}$ for each document d_j in a collection with respect to every category c_i of some user defined categorization scheme. If $a_{i,j} = 1$, then d_j is determined to belong to c_i. Otherwise, d_j is not in c_i.

Types of Automated Text Categorization

At the end of the 1980's a "semi-automatic" categorizer called CONSTRUE was constructed using a traditional expert system approach. It may be called "semi-automatic" because, although it automatically parsed and categorized news stories, the rules it used were generated by human experts [17]. The major drawback with such an approach is the expense of modifying the system which requires a great deal of time in consultation with expensive human experts.

As the 1990's progressed, machine learning techniques borrowed from artificial intelligence were applied to the task. The aim of these was to minimize the role of the human experts used in CONSTRUE. Algorithms were devised which would take information from a set of training documents and generate a function $CSV_i : D \rightarrow [0, 1]$.

Cat.	No.	Document Content
A	d_1	Pease porridge hot. Pease porridge cold.
A	d_2	Pease porridge in the pot. Nine days old.
B	d_3	Some like it hot. Some like it cold.
B	d_4	Some like it in the pot. Nine days old.
?	d_5	I hate cold pease.

Figure 2.2: A Sample Document Collection

This function maps documents d_j from a document set D to a *Categorization Status Value (CSV)* which indicates the strength of the categorizer's "belief" that d_j should be placed in category c_i. Next, a *threshold value* τ is defined such that if $CSV_i(d_j) \geq \tau$, then the categorizer will decide to place d_j in category c_i, and otherwise won't [47].

There have been many different approaches to automatic text categorization which reflect the many approaches to the general problem of machine learning [60, 47]. These approaches may be roughly divided into two groups, *statistical* and *symbolic*, depending on the extent to which CSV_i is defined in terms of numerical (statistical), as opposed to symbolic, concepts [47, 22, 51].

As an example to illustrate the differences among the various categorizers, consider the sample document collection in figure 2.2 borrowed from Witten [59]. There are two categories, A and B, each containing two documents for training the categorizer. There is one test document (d_5) available for categorizing.

Statistical Categorizers

Statistical categorizers borrow concepts from statistics such as probability and regression to define the categorization status value function CSV_i. Six of the major approaches to constructing statistical categorizers are presented here: Probabilistic, Rocchio, k-Nearest Neighbor, Regression Model, Neural Network, and Support Vector Machine categorizers.

Probabilistic Categorizers define CSV_i in terms of probabilities. They use Bayes' Rule for conditional probability to calculate $P(c_i|d_j)$, the probability that given a

document d_j, that document belongs to a category c_i. Since $P(c_i|d_j)$ is not readily available, Bayes' Rule

$$CSV_i = P(c_i|d_j) = \frac{P(c_i)P(d_i|c_i)}{P(d_i)} \tag{2.1}$$

is used to calculate $P(c_i|d_j)$ using quantities deducible from a training set of documents.

Consider the document collection in figure 2.2. Dividing the number of training documents assigned to category A by the total number of documents in the training collection gives an estimate for $P(c_1) = P(A)$, the probability that a randomly selected document belongs to category A. Doing the same for category B we have:

$$P(A) = \tfrac{2}{4} = 0.5 \quad P(B) = \tfrac{2}{4} = 0.5.$$

Probabilistic categorizers represent documents as vectors. The simplest way to do this is to assign 1 to a vector entry if it contains a term appearing in the training collection and 0 otherwise. I will call this the "bit-vector approach," and it will be complicated later. Our document collection is shown as bit vectors in figure 2.3. Note that d_5 is only represented by terms which already appeared in the training set. The terms *I* and *hate* are discarded and are thus considered irrelevant to the categorization.

Given that we are representing documents as term vectors, we can rewrite $P(d_j)$ as $P(< t_1, \ldots, t_n >)$. This is the probability that a document occurs in our collection, and it will be constant relative to the category c_i assuming that the documents in the collection are unique and that the collection remains unchanged for the duration of the categorization. Since we will be looking for the c_i for which equation 2.1 is

maximum, we need not calculate $P(d_j)$ and can drop the expression from Bayes' theorem [32].

Calculating $P(< t_1, \ldots, t_n > |c_i)$ is difficult unless the assumption is made that terms are conditionally independent given their category. In other words, the probability of observing the conjunction of the terms $< t_1, \ldots, t_n >$ in a given category will be equivalent to the product of the probabilities that each individual term will be observed in a given category. Bayesian categorizers which make this assumption are called *Naive Bayes* categorizers. With this assumption, we have the equation

$$P(< t_1, \ldots, t_n > |c_i) = \prod_i P(t_i|c_i) \tag{2.2}$$

To determine where our test document d_5 is to be placed, we need to compare the two values:

$$P(A| < cold, pease >) = P(A)P(cold|A)P(pease|A) = .5 \times .5 \times 1 = .25$$
$$P(B| < cold, pease >) = P(B)P(cold|B)P(pease|B) = .5 \times .5 \times 0 = 0$$

$P(cold|A)$ is calculated by taking the number of documents in category A which contain the term *cold* (d_1) and dividing by the total number of documents assigned to the category (d_1 and d_2). The other conditional probabilities are calculated in the same way:

$$P(cold|A) = \tfrac{1}{2} = .5 \quad P(pease|A) = \tfrac{2}{2} = 1$$
$$P(cold|B) = \tfrac{1}{2} = .5 \quad P(pease|B) = \tfrac{0}{2} = 0$$

Although $CSV_A < CSV_B$ we can't automatically conclude that document d_5 should be placed in category A. That decision depends ultimately on the threshold value τ. A 0.25 probability may not be large enough to place document d_5 in

Doc.	cold	days	hot	in	it	like	nine	old	pease	porridge	pot	some	the
1	1	0	1	0	0	0	0	0	1	1	0	0	0
2	0	1	0	1	0	0	1	1	1	1	1	0	1
3	1	0	1	0	1	1	0	0	0	0	0	1	0
4	0	1	0	1	1	1	1	1	0	0	1	1	1
5	1	0	0	0	0	0	0	0	1	0	0	0	0

Figure 2.3: Sample Documents Represented by Bit-Vectors

category A. Determining τ is an important part of constructing a categorizer and an overview of the topic is available in Sebastiani's work [47, section 7].

Details for this type of categorizer are given as an example of probabilistic machine learning by Mitchell [32] and have been evaluated by Moulinier [33] and Lewis [26]. Lewis elsewhere gives an overview of the many variations on the probabilistic categorizer [25].

Rocchio Categorizers have the deepest roots in information retrieval. They are derived from attempts to increase the effectiveness of text retrieval systems by modifying the query in such systems through the addition of terms or by changing the parameters of a query based on the nature of the text collection [15]. As in the case of the Probabilistic Categorizers, documents are viewed as vectors of terms (figure 2.3). The Rocchio method computes a weight vector $< w_{1i}, \ldots, w_{ri} >$ (where r is the same length as our document vectors) for each category c_i using the formula

$$w_{ki} = \left(\frac{\beta}{|\{\vec{d} : \vec{d} \in c_i\}|} \times \sum_{\{\vec{d}:\vec{d} \in c_i\}} w_{kj} \right) - \left(\frac{\gamma}{|\{\vec{d} : \vec{d} \notin c_i\}|} \times \sum_{\{\vec{d}:\vec{d} \notin c_i\}} w_{kj} \right) . \quad (2.3)$$

The weight vector $< w_{1i}, \ldots, w_{ri} >$ is also called the *profile* of its corresponding category. The variables β and γ provide parameters for determining the relative importance of positive examples versus negative examples for a category. If γ is set to 0 and β to 1, then the profile vector gives the *centroid* of the positive examples [9, 47].

Within the Rocchio framework, there are several approaches to determining where document d_5 belongs. The simplest approach is to calculate CSV_i as the dot product of d_5's vector and each category's profile:

$$CSV_i = \sum_{k=1}^{r} w_{ki} \times t_{k5} \tag{2.4}$$

where t_{k5} is the kth term of d_5. A more sophisticated approach might use the *cosine similarity* function

$$CSV_i = cos_i = \frac{\sum_k^r w_{ki} \times t_{k5}}{\sqrt{\sum_k^r w_{ki}^2} \times \sqrt{\sum_k^r t_{k5}^2}} \tag{2.5}$$

If we set $\beta = 1$ and $\gamma = 0$, the profiles of the two categories A and B are

$$\vec{A} = <.5, .5, .5, .5, 0, 0, .5, .5, 1, 1, .5, 0, .5>$$
$$\vec{B} = <.5, 0, .5, 0, 1, 1, .5, .5, 0, 0, .5, 1, 1>$$

Using the inner product we get

$$CSV_A = 1.5$$
$$CSV_B = .5$$

and the cosine similarity gives

$$CSV_A = cos_A = 1.0607$$
$$CSV_B = cos_B = .3536$$

The Rocchio method is easy and efficient to implement, but it assumes that there

is only one centroid per category. For categories better modeled using multiple centroids, the method does not perform well [47, 27].

The *k-Nearest Neighbors (k-NN) Categorizers* take a test document d and search for the k most similar documents to d. The categories of the k documents are noted, and the similarity measure between the documents is used to rank the category. For example, consider classifying our example document d_5. To compute the distance between documents, we can use the *Euclidean distance* measure:

$$Dist(d_i, d_j) = \sqrt{\sum_{i=1}^{n}(d_i - d_j)^2} \tag{2.6}$$

although other measures of distance, such as cosine similarity, can be used [63]. The distances between document d_5 and the other documents will be:

$$Dist(d_1, d_5) = 2$$
$$Dist(d_2, d_5) = 8$$
$$Dist(d_3, d_5) = 4$$
$$Dist(d_4, d_5) = 11$$

If we set $k = 3$ and consider the three closest neighbors to document d_5, then

$$CSV_i = \sum_{j=1}^{k=3} \delta(c_i, f(d_j)) \tag{2.7}$$

where c_i is our ith candidate category and $f(d_j)$ returns the category to which d_j belongs. The function $\delta(a, b)$ is a decision function which returns 1 if $a = b$ and 0 otherwise. In this example,

Doc.	cold	days	hot	in	it	like	nine	old	pease	porridge	pot	some	the
1	2	0	2	0	0	0	0	0	4	4	0	0	0
2	0	2	0	2	0	0	2	2	2	2	2	0	2
3	2	0	2	0	4	4	0	0	0	0	0	4	0
4	0	2	0	2	2	2	2	2	0	0	2	2	2

Figure 2.4: Sample Documents Represented as TFIDF Vectors

$$CSV_A = \delta(A, f(1) = A) + \delta(A, f(2) = A) + \delta(A, f(3) = B) = 2$$
$$CSV_B = \delta(B, f(1) = A) + \delta(B, f(2) = A) + \delta(B, f(3) = B) = 1$$

This implementation of k-NN will most likely select category A for document d_5 (depending on how τ is defined). Although k-NN categorizers have been shown to be effective, they use a form of "lazy" learning such that there is no training stage *per se* and all calculations are performed at classification time. Hence the actual classification of documents may be less efficient than categorizers that do their training prior to classification [47, 60].

Regression Model Categorizers automatically learn a multivariate regression model from a training set of documents and their categories [61, 62]. These training data are represented by pairs of vectors, called *input* and *output* vectors respectively. The input vector is a traditional vector consisting of the weights of the words in the document. The output vector is the categories of the corresponding documents. A Linear Least Squares Fit (LLSF) is calculated on the training pairs of vectors producing a matrix of word-category coefficients. This matrix is a mapping from a given document to a vector of weighted categories. When the category weights are sorted, a ranking of candidate categories is obtained for the document.

The matrix of *input* vectors from our example is shown in figure 2.4, which is

simply the array of training documents from figure 2.3 with TFIDF weights, where

TF_{ij} = the number of times word w_j occurs in the ith document

and

$$IDF_j = \log_2\left(\frac{N}{df_j}\right) + 1.$$

The value N is the number of documents in the collection and df_j is the number of documents containing word w_j. The matrix of input vectors is the *input matrix.*

Corresponding to the input matrix is the *output matrix*, which is the matrix of *output* vectors:

Doc	*A*	*B*
1	1	0
2	1	0
3	0	1
4	0	1

Each row has a 1 if a training document is in one of the categories A or B and 0 if not. The LLSF problem is to find a matrix $F_{l>n}$ that minimizes the sum of residual squares:

$$\sum_{i=1}^{m} \| \vec{e_i} \|_2^2 = \| F\vec{i_i}^T - \vec{o_i}^T \|_2^2 = \| FI^T - O^T a \|_F^2, \tag{2.8}$$

where $I_{m\times n}$ and $O_{m\times l}$ are the input and output vectors, I^T and O^T their transposes, and $\vec{i_i}$ and $\vec{o_i}$ the ith input and output vectors. The vector $\vec{e_i} = F\vec{i_i}^T - \vec{o_i}^T$ is the error

of F in the mapping from $\vec{i}_i$ to $\vec{o}_i$. Furthermore, $\| \ldots \|_2 = \sqrt{\sum_j^l v_j^2}$ is the 2-norm of an $l \times 1$ vector and $\| \ldots \|_F = \sqrt{\sum_i^m \sum_j^l M_{ij}^2}$ is the Frebenius Matrix norm of an $l \times m$ matrix [62].

The LLSF problem can be solved using the singular value decomposition (SVD):

$$F = O^T U S^{-1} V^T \tag{2.9}$$

where O is the output vector and U, S and V are derived from the SVD of the input matrix I. The input matrix $I = USV^T$ where the columns of U are the eigenvectors of II^T and the columns of V are the eigenvectors of $I^T I$. S is diag(s_1,...,s_p) and contains p nonzero singular values $s_1 \geq \ldots \geq s_p > 0$ and $p \leq min(m, n, l)$ [62]. The s_i are in fact the square roots of the nonzero eigenvalues of both II^T and $I^T I$ [49].

For our example, we get the $F_{2\times 13}$ matrix:

$$\begin{bmatrix} .064 & .184 & .021 & .184 & .205 & .205 & .184 & .184 & .021 & .021 & .184 & .204 & .184 \\ -.247 & .071 & -.204 & .071 & -.133 & -.133 & .071 & .071 & -.204 & -.204 & .071 & -.133 & .071 \end{bmatrix}$$

This matrix can now be used to generate the output vector for the test document d_5. Let $\vec{x} = \vec{d_5}$. Then the output vector for $\vec{x}$ is given by

$$\vec{y} = (F\vec{x}^T)^T \tag{2.10}$$

In our case

$$\vec{y} = <.169, -.902>.$$

Finally, we will place $\vec{x}$ in the category with the highest relevance, where relevance

is defined by the cosine similarity (equation 2.5) between the output vector of a document, such as $\vec{y}$, and a category.

$$relevance(\vec{x},\vec{c}) = cos(\vec{y},\vec{c}) = \frac{y_1c_1 + y_2c_2 + \ldots + y_lc_l}{\sqrt{y_1^2 + y_2^2 + \ldots + y_l^2}\sqrt{c_1^2 + c_2^2 + \ldots + c_l^2}} \quad (2.11)$$

Relevance scores range from -1 to 1 [62]. If we express the categories simply as the bit vectors

$\vec{A}$ = <1,0> $\vec{B}$ = <0,1>

we get the following relevance values:

$$CSV_A = relevance(\vec{x},\vec{A}) = \frac{.169}{\sqrt{.169^2+(-.902)^2}} = .184$$
$$CSV_B = relevance(\vec{x},\vec{B}) = \frac{-.902}{\sqrt{(.169)^2+(-.902)^2}} = -.983$$

Thus the LLSF categorizer will conclude that document d_5 is more relevant to category A.

Neural Network Categorizers create a neural network for each category and attempt to learn a mapping from words or documents (represented as vectors) to a category. The simplest neural network algorithm applied to text categorization is called *Perceptron* [7, 36]. An n-dimensional weight vector $w = < w_1, w_2, \ldots, w_n >$ is created such that w_i is the weight of term t_i. Intially an equal weight is assigned to each term. Each document is represented by a vector of strengths s_i such that document $\vec{d} = < s_1, s_2, \ldots, s_n >$. The strengths can be determined in various ways. One way is a bit-vector where $s_i = 1$ if term t_i is in the document and $s_i = 0$ otherwise. Or the strength could be the TFIDF value of a term. For our example, we use TFIDF representation from figure 2.4 [7].

The Perceptron algorithm will predict that a document $\vec{d}$ belongs to a category c represented by the weight vector $\vec{w}$ iff

$$\sum_{j=1}^{m} w_{ij} s_{ij} > \tau$$

where τ is a threshold value.

To train a category weight vector, a training document is passed through the Perceptron algorithm. If the training document belongs to c but the Perceptron algorithm says it does not, then a promotion parameter $\alpha > 0$ is added to all weights wi corresponding to a non-zero strength s_i. Conversely, if a document does not belong to c but the Perceptron algorithm predicts that it does, then α is subtracted from all such weights.

For example, suppose we set $\tau = .95$, $\alpha = 1$ and the initial thirteen weights (following Dagan [7]) of the weight vector to $.95/13 \approx .073$:

$$w_A = <.073,.073,.073,.073,.073,.073,.073,.073,.073,.073,.073,.073,.073>$$

If we pass document d_1 through the Perceptron algorithm, it predicts incorrectly that document d_1 does not belong to category A. The weight vector will be updated to:

$$w_A = <1.073,.073,1.073,.073,.073,.073,.073,.073,1.073,1.073,.073,.073,.073>$$

The final weight vectors for the two categories are:

$$w_A = <-.927,1.073,-.927,.073,-.927,-.927,.073,.073,1.073,1.073,.073,-.927,.073>$$
$$w_B = <.073,.073,.073,.073,1.073,1.073,.073,.073,-.927,-.927,.073,1.073,.073>$$

Using these weight vectors, Perceptron will not place document d_5 in either category with $\tau = .95$.

Perceptron is an example of a linear neural network. Non-linear networks with multiple "layers" of vectors have been constructed but with little improvement in effectiveness for text categorization [57]. Perceptron is also an *additive* learner. Better categorizers are *multiplicative* learners, that is, the parameter α is multiplied or divided from the weights instead of added or subtracted [7]. One drawback with neural networks is that some manual or automatic method is required to set the parameters α and τ. The neural-network categorizer BalancedWinnow performed well compared to other classifiers [7].

Support Vector Machine (SVM) Categorizers make use of the concept of a support vector machine developed by Vapnik [56]. SVM's represent an attempt to improve on neural networks by removing the ad hoc nature of assigning values to the parameters α and τ in neural networks and by developing a general mathematical foundation for neural networks and similar automatic learners.

Joachims applied SVM's to the text categorization task [19, 20, 21]. Support vectors are simply those training examples, expressed as vectors, which rest closest to the margin or hyperplane separating two categories of objects. In figure 2.5 the vectors in circles are the support vectors.

To find the optimal hyperplane between two classes of training examples, support vector machines in their basic linear form try to minimize the Euclidean norm of $\vec{w}$:

$$Minimize : \| \vec{w} \| = \vec{w}^T \vec{w} \quad so\ that : \ \forall_i : y_i[\vec{w} \cdot \vec{d} + b] \geq 1 \tag{2.12}$$

where y_i is +1 if document d_i is in a category c and -1 if not. Joachims represented documents using the TFIDF vector representation as in figure 2.4. Although SVM's can be used to construct polynomial learning machines and two-layer neural networks, SVM's for text categorization are limited to linear learning because of the large size of

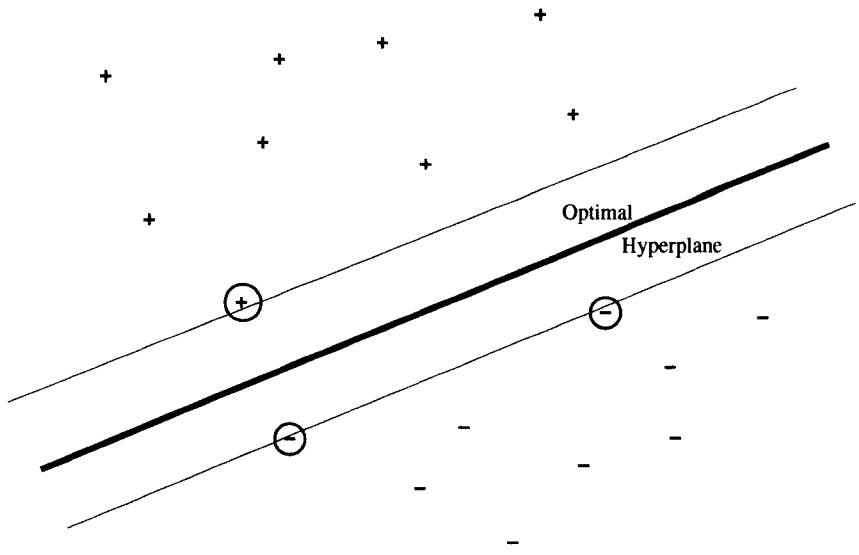

Figure 2.5: Support Vectors and their Hyperplane

document collections. Document collections are not always linearly separable, however, and provisions must be made to deal with such cases, such as adding slack values or ignoring training samples which contribute to the inseparability of the data [19].

To simplify this optimization problem Lagrange multipliers are used to convert the problem to a quadratic optimization problem for minimizing:

$$-\sum_{i=1}^{n}\alpha_i + \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \vec{d_i}\cdot\vec{d_j} \tag{2.13}$$

given the conditions that

$$\sum_{i=1}^{n}\alpha_i y_i = 0 \; and \; \forall_i : \alpha_i \geq 0 \tag{2.14}$$

Algorithms for solving 2.13 can be found in [23] and [41]. Since our sample is so small simple inspection revealed that 2.13 is minimized (assuming slack variables .01) when

$\alpha_1 = \alpha_3 = 0$ and $\alpha_2 = \alpha_4 = .01$. Since $\alpha_1 = \alpha_3 = 0$, documents d_2 and d_4 are the support vectors in this case.

The category status value for the SVM approach can now be defined as

$$CSV_i = sign[\vec{w} \cdot \vec{d} + b] \tag{2.15}$$

where $CSV_A = sign[\cdot] = +$ and $CSV_B = sign[\cdot] = -$. The value of $\vec{w}$ is given by

$$\vec{w} = \sum_{i=1}^{n} \alpha_i^* y_i \vec{d_i} \tag{2.16}$$

where the α_i^*'s are the coefficients which make 2.13 minimum. Since $\alpha_1^* = \alpha_3^* = 0$, our $\vec{w}$ is derived from $\alpha_2^* y_2 \vec{d_2} + \alpha_4^* y_4 \vec{d_4}$ with the result:

$\vec{w}$ = <0, 0, 0, 0, -0.2, -0.2, 0, 0, 0.2, 0.2, 0, -0.2, 0>

To derive b Joachims [19] picks one training document from each category is arbitrarily such that

$$b = \frac{1}{2}(\vec{w} \cdot \vec{d_+} + \vec{w} \cdot \vec{d_-}). \tag{2.17}$$

Selecting documents d_1 and d_3 gives us $b = -.02$. The resulting instantiation of 2.15 correctly categorizes the training documents and places document d_5 in category A. Dumais [9], Taira [55], and Yang [63] have evaluated SVM categorizers, which did well compared to k-NN, neural net, LLSF, and Naive Bayes categorizers.

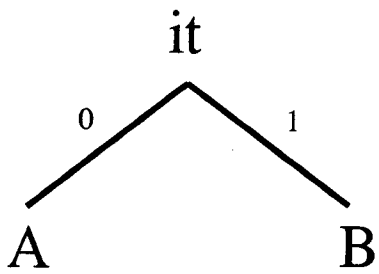

Figure 2.6: Decision Tree for Sample Documents

Symbolic Categorizers

Symbolic categorizers come in two basic varieties: Decision Tree and Rule-based categorizers. The Decision Tree approach is presented here. Chapter 3 discusses Rule-based categorizers in detail.

Decision Tree Categorizers use J.R. Quinlan's decision tree algorithms [44, 32] to select informative terms from test documents. An *information gain* measure defines the informativeness of terms. The algorithms then predict the category for a document based on the occurrence of term combinations in the tree. Implementations of text categorizers have been built using Quinlan's decision tree induction programs ID3 [44, 12], C4.5 [5, 19], and C5 [30]. To apply ID3 (following Mitchell [32]) to our example from figure 2.3, we first define the set T of all terms from the training documents 1-4 of our collection. If all our examples belonged to a single category, a single node tree giving that category would be returned.

For our example, however, we want to select the term which best separates training documents in category A from those in category B. The best term is the one with the highest *Gain*:

$$Gain(D,t) = Entropy(D) - \sum_{v=Values(T)} \frac{|D_v|}{|D|} Entropy(D_v) \tag{2.18}$$

where D is the collection of training documents, t is term from T, $Values(T)$ is the set of all values of T (in the bit-vector case the possible values are {0,1}), and $D_v = \{d \in D \mid T \text{ has value } v \text{ in } d\}$. Entropy is defined as

$$Entropy(S) = \sum_{i=1}^{c} -p_i \; log_2 \; p_i. \tag{2.19}$$

where c is the number of categories, in our case $c = 2$.

For example, we would compute $Gain(D, cold)$ for the training documents from our collection in figure 2.3 as follows. The entropy over the whole collection is

$$Entropy(D) = -\tfrac{2}{4}\log_2\tfrac{2}{4} - \tfrac{2}{4}\log_2\tfrac{2}{4} = 1$$

because there are 2 out of 4 documents in category A and 2 out of 4 documents in category B. The term *cold* can take two values: 0 and 1, and S_1 contains documents d_1 and d_3 while S_0 contains documents d_2 and d_4. Hence,

$$Entropy(S_1) = -\tfrac{1}{2}\log_2\tfrac{1}{2} - \tfrac{1}{2}\log_2\tfrac{1}{2} = 1.$$

Five terms share the maximum Gain of 1: *it*, *like*, *pease*, *porridge*, and *some*. If we pick *it* arbitrarily, ID3 will create a tree with *it* as its root, split the examples into those where *it* is 0 (documents d_3 and d_4) and 1 (documents d_1 and d_2), and then for each branch of the tree, call itself again. When it calls itself, it will consider only documents d_3 and d_4 on one branch and d_1 and d_2 on the other. It will also exclude *it* from the set of possible terms for all future branches.

The first branch will only have documents from category A to work on. When ID3 has a branch with only documents from a single category, it will stop work on that branch and label the bottom node with the name of that category, in this case A. The

same occurs on the other branch but it's leaf is assigned B. Our little tree is shown in figure 2.6. Any document with a 0 in its vector for the term *it* will be classified in A and in B otherwise. Document d_5 accordingly will be placed in category A.

Rule-based or *decision rule* classifiers automatically learn the type of conditional rules used in a traditional expert system such as CONSTRUE. However, rather than relying on human experts to develop these rules, learning algorithms are applied to the training documents to construct the rules. Among these types of categorizers are CHARADE [34], DL-ESC [29], RIPPER [5], RULECLAS [58], SCAR [35], and SWAP-1 [2].

Rule-based classifiers have the advantage over other classifiers (except for standard expert systems) that how classification is done is more evident to humans. Instead of relying on complex mathematical formulae, the rules are expressed in terms of logical conditionals. For example, the rule-making software C-KANT, which was constructed for this project, created the rules in figure 2.7. The first rule states that if both terms *controls* and *conservation* are found together in a document, then that document should be classified in category 12.1 of the 3.69 Topical Guidelines for the Nuclear Regulatory Commission (see figure 4.5). Category 12 documents are to cover "information for preparation of a geologic repository environmental impact statement" and section 12.1 documents cover "environmental" aspects. So, C-KANT has concluded that documents referring to *controls* and *conservation*, or to *shrubs* alone, or to both *permitting* and *accordance*, should be placed in category 12.1.

In this chapter the concept of automatic text categorization was defined and examples of statistical categorizers were presented. Also, one type of symbolic categorizer, the decision tree categorizer, was discussed. Details of rule-based categorizers in general and specifically the engine which was used in this project are the subject of the next chapter.

```
IF {"controls" and "conservation"}
 is\are in a doc
then the document is in CATEGORY cat-12.1
IF {"shrubs"}
 is\are in a doc
then the document is in CATEGORY cat-12.1
IF {"permitting" and "accordance"}
 is\are in a doc
then the document is in CATEGORY cat-12.1
IF {"demographic"}
 is\are in a doc
then the document is in CATEGORY cat-12.2
IF {"rail" and "destination"}
 is\are in a doc
then the document is in CATEGORY cat-12.3
IF {"reservation"}
 is\are in a doc
then the document is in CATEGORY cat-12.3
IF {"adopt" and "accidents" and "activities"}
 is\are in a doc
then the document is in CATEGORY cat-12.3
```

Figure 2.7: Rules Generated by Rule-Based Categorizer

CHAPTER 3

C-KANT: A KNOWLEDGE ACQUISITION ENGINE

The question posed by this thesis is whether thesaurus additions to a rule-based automatic text categorizer will improve performance. To answer this question, a rule-based categorizer was constructed. In this chapter the two basic components of the categorizer, CLIPS and C-KANT, are described. The rule learning algorithms IREP and RIPPER which are used in C-KANT will then be presented in detail.

CLIPS: The C in C-KANT

The rule-based text categorizer used for this study consists of two parts. The first part is the "off the shelf" expert system tool CLIPS. The second was constructed in the programming language C++ specifically for this project and is dubbed C-KANT, an acronym for *Clips-Knowledge Acquisition eNgine for Text categorization.* A brief introduction to CLIPS follows, after which C-KANT will be discussed in detail.

CLIPS stands for *C Language Integrated Production System* and was originally developed by NASA's Johnson Space Center as a rule-based expert system tool. Starting with version 5.0, CLIPS also included support for object-oriented and procedural programming [46].

An expert system (see figure 3.1) has at least three basic components: (1) a set of facts, (2) a set of rules called the *knowledge base* of the system, and (3) an inference engine. The rules of the expert system take the form of the logical conditional. In English, such rules have the form *if* X *then* Y, where X is called the *antecedent* of the rule and Y is the *consequent.* The facts are statements which may (or may not) match

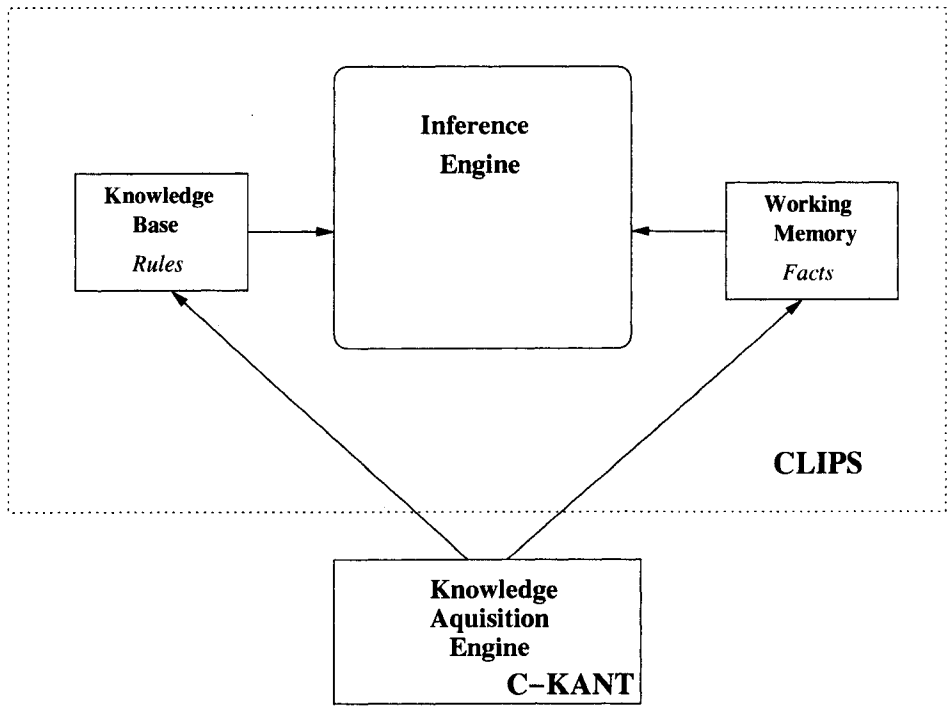

Figure 3.1: The Components of an Expert System

the contents of the antecedent X of a rule. The inference engine in the expert system has the ability to determine if the facts match any of the antecedents of the rules, and if a match is found, to perform whatever commands are stated in the consequent [14].

As an expert system tool, CLIPS provides an inference engine as well as the ability to understand suitably expressed facts and rules. But before it can function as an expert system, CLIPS must be given rules and facts as input. CLIPS will then use its inference engine to determine what results or actions follow from those rules as applied to the given facts. Of course, one way to enter the rules and facts is to enter them manually. Human beings, known as knowledge engineers can create an expert system by formulating the rules and facts. This manual approach was in fact the one used to construct the early text categorization system called CONSTRUE, which was built for the Reuters corporation to categorize news stories [16].

The manual approach, however, is subject to the *knowledge acquisition bottleneck* [14]. Traditional expert systems require that a human knowledge engineer engage in extensive interviewing with experts who possess the desired knowledge. Repeated

interviews are needed in order to construct, test, and revise the resulting system so that it adequately models the experts' knowledge. Since the interviewing process is very expensive, there is a great incentive to automate the process of transferring knowledge to the expert system from the human expert. In the case of text categorization, the number of texts, as in the case of large, unstructured text databases such as the World Wide Web, may simply be too great for human beings to classify economically [47, 14].

C-KANT

C-KANT is designed to provide automatic knowledge acquisition for text categorization. Of course, human expertise is not entirely removed from the process since C-KANT must be trained with documents originally categorized by human experts. However, once these training documents are found, C-KANT creates the rules and the facts for CLIPS's inference engine. Figure 3.2 depicts C-KANT's functions in relation to CLIPS.

C-KANT performs three tasks which result in (1) a complete CLIPS program as well as (2) the facts needed by CLIPS to categorize documents. Typically, C-KANT works are follows. First, C-KANT will parse through a collection of training documents and generate a set of positive and negative examples for each category. Since C-KANT is working with a training set of documents, it is assumed that the documents have already been categorized by experts. A positive example, therefore, of a category c is a document which an expert has placed in category c, and a negative example one which the expert decides does *not* belong to c.

C-KANT may optionally apply certain standard information retrieval techniques to help reduce the dimensionality of the training documents, such as stop-word removal and stemming, at this stage as well. Stemming is the removal of suffixes to

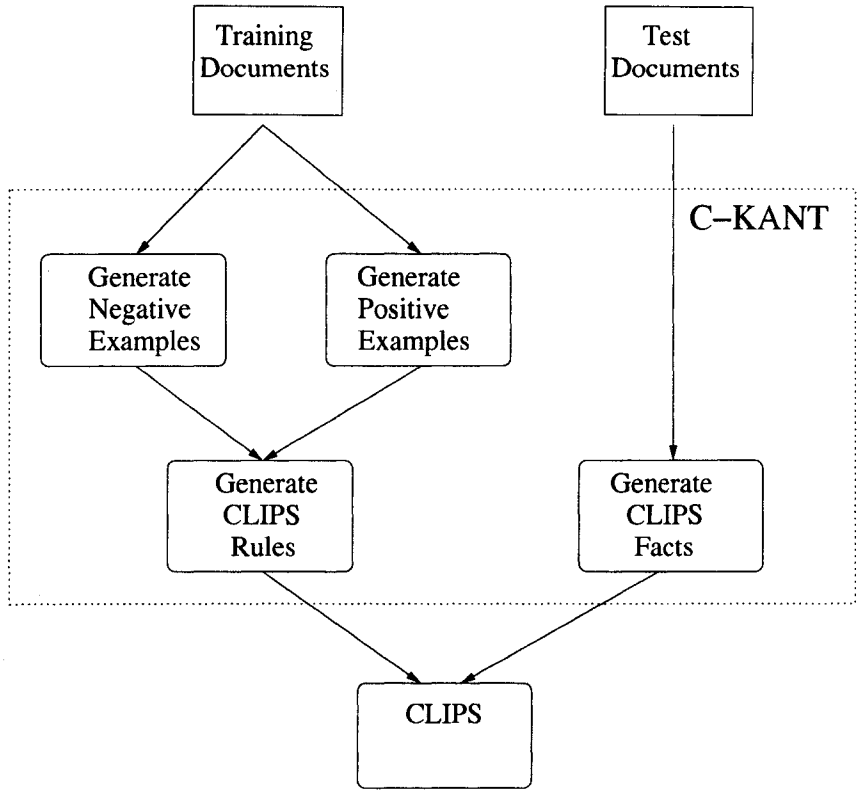

Figure 3.2: C-KANT's Structure

conflate morphological variants of a term. For example, the terms *engineering*, *engineered*, and *engineer* are all "stemmed" to *engineer*. Stemming is a traditional tool for improving results in information retrieval and automatic text categorization [11]. A stop-word list removes very common terms such as *is* and *the* whose inclusion would normally hurt results [59].

C-KANT's second task is to use the positive and negative examples to generate the rules for each category. This process makes use of the IREP algorithm which will be discussed in detail in the next section. As an option, optimizations can be applied using the RIPPER algorithm, which is presented later in this chapter.

On its third task, C-KANT will look through a set of test documents and transform these into facts which CLIPS can understand. The resulting program plus facts is then loaded into CLIPS. CLIPS will categorize the documents in the fact list as well as calculate measures of performance such as precision and recall for the categorization.

The IREP algorithm

C-KANT uses the *Incremental Reduced Error Pruning (IREP)* algorithm to generate rules based on the documents from its training set [13, 5]. IREP proceeds in two stages. First, it attempts to *grow* a rule, and then it tries to *prune* that rule.

IREP begins growing rules by constructing a rule with an empty antecedent such as

$$\_\_\_ \in d_j \rightarrow d_j \in c_i.$$

This rule states that if any term is in a document d_j, then that document will belong to category c_i. Each unique term from a given set of documents, listed in a dictionary for this purpose, is then added to the antecedent in a "greedy" fashion. For example, suppose the dictionary contains terms $t_1, t_2, \ldots, t_n$. Then for each term t_k, IREP constructs a rule

$$t_k \in d_j \rightarrow d_j \in c_i.$$

Each rule so constructed will cover a certain number of positive and negative examples. A rule *covers* an example iff the example satisfies the antecedent of the rule. Thus, if the rule

$$t_1 \in d_j \rightarrow d_j \in c_i$$

is under consideration, it covers all the documents, positive and negative, which contain the term t_1. A rule with an empty antecedent is satisfied by any example.

To determine which candidate rule is "best," each new rule r_{i+1} is compared to

the rule determined at the previous stage, r_i, using the *information gain* formula [45]:

$$Gain(r_i, r_{i+1}) \equiv T_{i+1}^{+} \times \left(-\log_2 \frac{T_i^{+}}{T_i^{+} + T_i^{-}} + \log_2 \frac{T_{i+1}^{+}}{T_{i+1}^{+} + T_{i+1}^{-}} \right). \tag{3.1}$$

T_m^{+} is the number of positive examples which are covered by rule r_m and T_m^{-} is the number of negative examples covered.

When the term t_k with the highest gain is found, the algorithm will attempt to enhance the rule containing t_k alone by considering rules which use t_k conjoined with other terms in the dictionary. So for each $l \neq k$, IREP evaluates the gain of

$$t_k \in d_j \wedge t_l \in d_j \rightarrow d_j \in c_i.$$

In this case, both t_k and t_l must appear together in the same document for that document to count as a covered example. IREP will continue to add atoms of the form $t_\alpha \in d_j$ to the antecedent of the rule until there is no more information gain from doing so or other stopping conditions are met, as we will see below.

After a rule with the maximum information gain is grown, IREP tests to see if a shorter and thus more efficient version of the rule is as good as the original, longer version. To do so, it attempts to *prune* the rule. Pruning is necessary to avoid a common problem for both decision tree and decision rule categorizers called *overfitting* of the training data [13, 5]. Overfitting occurs when the tree or the rules train so closely to the training data that they do not generalize well to new examples. Pruning is meant to reduce the "growth" of rules and trees so that their shorter versions might better cover unseen examples.

In the pruning stage, the data set for a category, consisting of positive and negative examples, is divided into two sets: a data set for growing and a data set for pruning. Usually two-thirds of the data for a category is used for growing and one-third for

pruning. This was the ratio used for C-KANT's tests. IREP will take the rule learned in the growing stage and successively remove atoms one by one to determine if the removals degrade the rule. The measure of the "goodness" of a rule r_i in this case is determined by the formula:

$$f(r_i) = \frac{U_i^+ - U_i^-}{U_i^+ - U_i^-} \tag{3.2}$$

where U_i^+ is the number of positive examples in the pruning set covered by pruned rule r_i and U_i^- is the number of negative examples covered. The pruned rule with the maximum value for $f(r_i)$ is kept and that rule is added to the rule set for the category.

Deciding when to stop the IREP cycle is more an art than a science. For C-KANT, the rule-building cycle terminates if either (1) all of the positive examples have been covered, or if (2) all the negative examples have been covered, or if (3) no progress has been made, that is, no rule was found in a given cycle that covered any new positive examples, or (4) if the *description length* of the current rule set and its examples are more than 64 bits longer than the smallest set so far. This last criterion makes use of *Minimum Description Length (MDL)* heuristic developed by Quinlan and applied to rule-based categorization by Cohen [5]. It will be discussed in greater detail in the next section. The IREP algorithm is summarized in figure 3.3.

RIPPER Optimizations

William Cohen's RIPPER [3, 5] algorithm is a modification of IREP. Since the modification led to better results in experiments, the RIPPER algorithm was included as an option in C-KANT. There are two main changes to the IREP algorithm in RIPPER. The first has to do with the stopping criterion for the learning of a rule set. The second modification is the addition of multiple optimizing runs which attempt to

Note: DL(X) returns the description length of X.

```
functionIREP(Data):
begin
RuleSet ← empty rule set
while Data has positive examples and negative examples
      and progress is made:
      split Data into GrowData and PruneData
      Rule ← GrowRule(GrowData)
      Rule ← PruneRule(PruneData)
      if DL(RuleSet ∪ Rule) > min (DL(RuleSet)) + 64
        add Rule to RuleSet
      else
        return RuleSet and end IREP
      remove positive and negative examples from Data covered by Rule
end while
return RuleSet
end
```

Figure 3.3: The IREP algorithm

replace rules in a learned rule set with more effective rules. Both modifications make extensive use of the *Minimum Description Length (MDL)* measure.

The MDL measure is based on the communication model of information typical of information theory. In general it is a measure of the length of a theory T and the data D on which T is based. The description length of T is the cost of a message encoding T, the *theory cost*, and the cost of encoding D given T is true. The first cost represents the complexity of the theory and the second the extent to which the theory fails to account for the data [43].

The philosophical justification for MDL comes from Ockham's razor, which claims that the simpler a theory the better. The MDL principle states that the theory with the shortest description length is the better theory. Following [43] C-KANT calculates the length of a theory from the length of its n rules:

$$DL(T) = \sum_{i=1}^{n} -\log_2 \frac{k_i^2}{N} \tag{3.3}$$

where k in the number of terms in the ith rule's hypothesis and N is the number terms in the dictionary. The description length of the data is given by

$$\begin{aligned} DL(D) \;=\; & -\log_2(C+1) \\ & + fp \times (-\log_2 \frac{fp}{C}) \\ & + (C - fp) \times (-\log_2(1 - \frac{fp}{C})) \\ & + \log_2(U+1) \\ & + fn \times (-\log_2 \frac{fn}{U}) \\ & + (U - fn) \times (-\log_2(1 - \frac{fn}{U})) \end{aligned} \tag{3.4}$$

where C and U are respectively the number of documents covered and not covered by a rule, fp is the number of false positives and fn the number of false negatives.

RIPPER uses the description length measure in three ways. First, as mentioned in the last section, it provides a heuristic for stopping the rule-building loop in IREP. Second, each rule set is compressed using MDL as the criterion for compression as follows. Each rule in a rule set is examined beginning with the last rule added. Any rule which increases the description length is deleted thus compressing the rule set.

The third use of the description length measure is in additional optimizing steps. After IREP finishes constructing a rule set, the rule set is optimized to reduce the size of the set and hopefully increase its accuracy. Each rule r in the rule set, in the order it was added, is compared with two alternative rules, a *replacement* for r and a *revision* of r. The replacement rule is created by growing and pruning a new rule r'. Pruning is governed by the goal of minimizing error (defined by equation 3.2) over the entire rule set by comparing the rule set with r to the rule set with r' replacing r. The revision of r, r'', is grown by greedily adding terms to r instead of the empty rule. Finally r is replaced by one of the three r, r', r'' depending on which has the shortest description length after compression [5].

After optimization the rule set may end up covering fewer examples. For this reason IREP is called again on the uncovered examples. Cohen determined after experimentation that running the optimization step twice was optimal [5]. The RIPPER algorithm is given in figure 3.4.

This chapter described a rule-based categorizer consisting of two parts, a rule-generating component C-KANT and a rule-applying component CLIPS. The algorithms IREP and RIPPER used by C-KANT to generate rules were explained. The next task is to discuss how the categorizer was applied to the problem of determining the effectiveness of using thesauri in categorization. This is the topic of the next chapter.

```
function Optimize(RuleSet, Data)
begin
    foreach rule r ∈ RuleSet
      split Data into GrowData and PruneData
      r' ← GrowRule(GrowData)
      r' ← PruneRule(PruneData)
           guided by error on RuleSet + r' - r
      r'' ← GrowRuleFrom(r,GrowData)
      r'' ← PruneRule(PruneData)
           guided by error on RuleSet + r'' - r
      replace r with min(∀x ∈ {r, r', r''} : DL(Compress(RuleSet − c + x)))
end

function RIPPER(Data)
begin
    RuleSet ← IREP(Data)
    repeat 2 times
      RuleSet ← Optimize(RuleSet,Data)
      UnCoveredData ← examples in Data not covered
           by rules in RuleSet
      RuleSet ← RuleSet + IREP(UnCoveredData)
    end repeat
end
```

Figure 3.4: The RIPPER Algorithm

CHAPTER 4

TEST ENVIRONMENT

This chapter outlines the basic testing environment for this study. It first defines the standard measures for text categorization: precision, recall, and F_1. Next it describes the text collections used for testing. Finally, it presents the thesauri used in the experiments.

Evaluating Text Categorization

Automatic text categorizers are measured in terms of their "effectiveness." Effectiveness is usually defined using the *contingency table model.* In this model, it is assumed that each document in a collection has been assigned at least one category by an expert and that the expert's decision is correct.

This last assumption is a necessary simplification to deal with a significant difficulty. Automatic text categorization can suffer from *inter-indexer inconsistency,* the frequently occurring phenomenon of two experts giving contradictory judgments concerning the classification of a document. This phenomenon indicates an unavoidable subjective factor in the text categorization process. However, to make testing of automatic categorizers possible, it is assumed that the aim of these categorizers is to emulate the decisions of an expert (or consistent set of experts) on the assumption that the expert's judgments are correct [47].

Given an expert's judgment as to the "correct" category c_i for a document d_j, and an automatic categorizer's guess as to which category document d_j belongs, there are four possibilities. The expert and categorizer can both agree that d_j is in c_i, a result

		Expert's Judgment		
		$d_j \in c_i$	$d_j \notin c_i$	
Categorizer's Guess	$d_j \in c_i$	TP	FP	$TP + FP$
	$d_j \notin c_i$	FN	TN	$FN + TN$
		$TP + FN$	$FP + TN$	N

Figure 4.1: The Contingency Table for Text Categorization

called a *true positive.* The expert can claim that d_j belongs to c_i but the categorizer disagrees, which is a *false negative* result. Or, the categorizer may conclude that d_j belongs in c_i but the expert says it does not, which is a *false positive.* Finally both the expert and the categorizer can agree that d_j does *not* belong in c_i, a *true negative.*

For a given category c_i the number of decisions for each of these four types is counted. Let TP stand for the number of true positives counted for c_i, FP the false positives, FN the false negatives and TN the number of true negatives. These numbers can now be represented by an array called a *contingency table* (figure 4.1).

Two important measures borrowed from information retrieval [10], *recall* and *precision*, can be defined in terms of the entries and marginal values of the contingency table above:

$$recall = TP/(TP + FN) \tag{4.1}$$

$$precision = TP/(TP + FP) \tag{4.2}$$

The recall measure is the ratio of the documents correctly categorized over all the documents the expert thought *should* be in the category. Precision is the ratio of the documents correctly categorized over all the documents the categorizer *in fact* placed in the category. Another way to put this (following Lewis [24]) is that recall is the proportion of documents correctly categorized while precision is the proportion of categorized documents which are correct. Both values are real numbers from 0 to 1.

Precision and recall tend to be inversely related. High recall usually entails lower precision and higher precision usually lowers recall. For this reason, the goal of a categorizer is to try to achieve the highest values for both measures at the same time. It is easy to build a trivial categorizer (a *trivial acceptor*) with a recall of 1 for every category by placing every document in every category. However, the precision of such a categorizer will be very low. On the other hand, precision can be boosted by seldom placing any document in any category, which would tend towards very low recall. Such a categorizer could be called a *trivial rejector* [47].

Two other measures used in categorization tests are *fallout* and *overlap.*

$$fallout = FP/(FP + TN) \tag{4.3}$$

$$overlap = TP/(TP + FP + FN) \tag{4.4}$$

Fallout measures the proportion of documents incorrectly categorized. It gives the proportion of incorrectly categorized documents over the total number of documents which the expert decided should *not* be placed in the category.

Overlap indicates how much two categorizations are alike. In this case, overlap indicates how close the automatic categorizer's judgments were to the expert's, but it is sometimes used to determine how close two categorizations are without assuming one or the other is correct [24].

Each of these four measures is subject to the possibility of division by zero. For example, if a categorizer does not place any document in a particular category, the numerator of the precision formula for that category is zero. In such cases, C-KANT sets the precision, and any other measure where division by zero threatens, to an extremely low number. This procedure was also applied in the rare case that zero is divided by itself.

To use precision and recall to evaluate a set of classifications, the individual precisions and recalls are averaged. There are two ways of computing these averages. The first is *microaveraging*:

$$microaverage\ precision = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n}(TP_i + FP_i)} \tag{4.5}$$

$$microaverage\ recall = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n}(TP_i + FN_i)} \tag{4.6}$$

where subscripts indicate that TP_i, FP_i, etc., are measures for the i*th* category and n the total number of categories. Microaveraging "globally" sums over the individual decisions of the categorizer over all categories.

The second method for averaging these measures is *macroaveraging*:

$$macroaverage\ precision = \frac{\sum_{i=1}^{n} P_i}{n} \tag{4.7}$$

$$macroaverage\ recall = \frac{\sum_{i=1}^{n} R_i}{n}, \tag{4.8}$$

where P_i and R_i are the precision and recall of the ith category respectively and n is the total number of categories. Here precision and recall are first calculated within each category and then the category results are divided by the number of categories.

Researchers differ as to which of the two provides the better measure of effectiveness for categorization. Microaveraging tends to reward categorizers which do well only on categories with large numbers of positive examples. Macroaveraging rewards categorizers which do well when the number of positives varies widely among the categories. In other words, such categorizers perform well when some categories have

few positive examples in them but others have many. The majority of researchers prefer microaveraging [47]. Both measures will be made available in this study.

For many automatic text categorization systems it is possible to define and use the *breakeven point* of the precision and recall curves as the measure of effectiveness. Since precision and recall are typically inversely related, the breakeven point is simply the point at which their curves intersect. Such systems usually have available a numerical real value such as their categorical status value CSV which can represent the "confidence" that the categorizer has that a document belongs to a given category. The user must then define a threshold value which determines which confidence values are high enough to place the document in a category and which are not. Changing the threshold value is one way to tune such categorizers for better performance. The breakeven point in fact is the optimal setting which maximizes both precision and recall.

Rule-based categorizers do not typically return a confidence value. Instead they make a boolean decision that a document belongs to a category or not. Such categorizers place a document in a category or not depending on whether the antecedent of the relevant rule is satisfied by the document's attributes. For this reason, another value called the F_β *function* is used to define effectiveness in rule-based systems. For $0 \leq \beta \leq \infty$,

$$F_\beta = \frac{(\beta^2 + 1) \times P \times R}{\beta^2 \times P + R}, \tag{4.9}$$

where P and R are averages of precision and recall. Since there are two ways to average the two, microaveraging and macroaveraging, the F_β function will be computed in both ways.

The β in the F_β *function* represents the relative importance given to precision versus recall. When $\beta = 0$, F_β is precision, and as β nears ∞, F_β approaches recall.

Moulinier and Yang have shown that the breakeven point of a given categorizer is always less than or equal to its F_1 value [35, 63]. For this reason, I will present the F_1 value in both the microaverage and macroaverage versions.

The Collections

The text collections used in this study are a subset of a large collection of documents donated to ISRI by the Department of Energy (DOE). The DOE is constructing a large database of scientific, legal, and official documents for online legal discoveries by the DOE and other interested parties. Since many of the documents which will be placed in the database are in a hard copy form, they must be converted to an electronic form using scanning and Optical Character Recognition (OCR) technology. To study the properties of OCR text in information retrieval and text categorization, a representative sample of approximately 2,600 OCR documents (140,000 pages) called the Licensing Support Network (LSN) Prototype was created, and a copy was donated to ISRI for research purposes [52].

Such OCR text usually contains at least two types of errors: *segmentation* errors and *classification* errors. Segmentation errors encompass such errors as recognizing single letters as multiple characters, for example, reading 'rn' for 'm', reading multiple characters as single letters, e.g., 'ci' for 'd', and incorrect concatenation and division of terms, such as recognizing 'c at' for 'cat'. Classification errors are errors such as replacing 'o' with '9' in 'J9hn' [54]. Samples of OCR text from one of the collections used in this study appear in figure 4.2.

Two collections from the LSN database have been manually separated by human experts working for ISRI into categories developed by the Nuclear Regulatory Commission for categorizing documents about the Yucca Mountain Nuclear Waste Depository. These guidelines are referred to as the *3.69 Topical Guidelines* [6]. A list of the category names in the Topical Guidelines appear in figures 4.4 and 4.5.

```
be potentially damaging to structures (14orehard) - A f ield
test was conducted useing a backhOe mounted hydraulic ha=er
v1hich is being used elsewh4we on the yucca Mountain Project.
'The test was witnessed by the LLNL Principal investigator and
other heated block experiment of f icials - The observed results
can be summarized at; follows: The vibrations resulting fro'm
the hammer breaking the rock were observable, however no
discernable crack dilations occured on the surface where theY
would inost. likely occur. The hammer was ef fective in
fragmenting the surf ace rock. It was estimated that the
hammer fragmented at least 200 cubic feet of rock in 20

The issuance of the license will not be inimical to the common
defense and security and will riot constitute an unreasonable risk
to the health and safety of the public. A DGE GGAifiGatieR that it
4-4-provide at the geGIGgiG FePGGitGPy operatiGp'&amp;amp; aFea 66irh
safeg-uards as it require,.,. ornparable IDGE suFfaGe faGilitle6 W
prarneta the Wmmen defense and 6eGUFitY Will
--Gnst..... . .-buttable pFeGumptien Gf Gality to he GOmmen defepae
 and 60GUFity-
```

Figure 4.2: Examples of OCR Text

Small-DOE		
	Documents	
Categories: 7	Training	Testing
2.1	6	3
2.2	8	4
2.4	10	5
4.1	21	11
12.1	25	13
12.2	10	5
12.3	10	5
Total	**90**	**46**

Figure 4.3: The Small-DOE Collection

One collection, which we will call *Small-DOE*, consists of 136 documents. These were randomly divided into a training set of 90 documents and a test set of 46. Since the collection was so small, all 90 training documents were used as negative examples in the following way. When training for a given category such as 2.1, all training documents from all of the other categories were considered as negative examples for category 2.1.

The second collection, called *Big-DOE* consists of 1619 documents. This was split into 1074 training and 545 testing documents. Ten documents from each category were randomly selected and used as a pool of negative examples for training. That is, all documents in the pool except those marked as belonging to a given category c_i were considered negative examples for c_i. The number of training (Tr) and testing (Te) documents available in *Big-DOE* for each category is given in figures 4.4 and 4.5.

Adding Thesauri Terms to Rules

One approach used in information retrieval to achieve better results is to enhance the indexing or querying of a large document collection by adding terms from thesauri created for these tasks [48, 1]. These thesauri are often limited to specific domains of knowledge pertinent to the particular document collection. Such a thesaurus was

Big-DOE and the 3.69 Topical Guidelines			
Cat	**Tr**	**Te**	**Description**
0.1	20	10	Exclusionary Short Term Documents
0.2	21	11	Exclusionary Long Term Documents
1.0	20	10	General Information
1.1	28	14	General Facility Description
1.2	30	15	Basis for Licensing Authority
1.3	20	10	Schedules Relevant to the NRC/DOE Repository Programs
1.4	33	17	Any Publicly Available Information on Certification of Safeguards
1.5	0	0	Any Publicly Available Information on the Physical Security Plan
1.6	20	10	Site Characterization
1.7	0	0	License Specifications
1.8	19	10	Information Relevant to NRC Findings Regarding Compliance with Statutes
2.0	12	6	The Natural Systems of the Geologic Setting
2.1	20	10	Geologic System
2.2	20	10	Hydrologic System
2.3	14	7	Geochemical System
2.4	32	16	Climatological and Meteorological Systems
2.5	19	10	Integrated Natural System Response to the Maximum Design Thermal Loading
2.6	13	7	Processes and Events (anticipated and unanticipated, potentially disruptive)
2.7	17	9	Effectiveness of Natural Barriers Against the Release of Radioactive Material to the Environment
3.0	7	4	Geologic Repository Operations Area (GROA): Physical Facilities
3.1	33	17	Surface Facilities
3.2	18	9	Shafts/Ramps
3.3	24	12	Underground Facility
3.4	20	10	Interface of Structures, Systems, and Components
3.5	20	10	Retrievability of Waste
3.6	20	10	Effectiveness of the GROA against the Release of Radioactive Materials to the Environment
4.0	8	4	Engineered Barrier Systems
4.1	21	11	Waste Package
4.2	20	10	Waste Form
4.3	0	0	Underground Facility
4.4	20	10	Engineered Barrier System Waste Package Emplacement Environment
4.5	20	10	Engineered Barrier System Alternative Design Features
4.6	20	10	Effectiveness of Engineered Barriers Against the Release of Radioactive Material to the Environment

Figure 4.4: 3.69 Topical Guidelines 0.1-4.6

Big-DOE and the 3.69 Topical Guidelines continued			
Cat	**Tr**	**Te**	**Description**
5.0	20	10	Overall System Performance Assessment
5.1	20	10	Basic Approach
5.2	20	10	System Description
5.3	20	10	Cumulative Release of Radioactive Materials
5.4	19	10	Undisturbed Performance
6.0	0	0	Conduct of Repository Operations
7.0	6	3	Land Ownership and Control
7.1	15	8	Plans for Restricting Controlled Area Access
7.2	20	10	Plans for Regulating Land Use Outside the Controlled Area
7.3	0	0	Plans for Regulating Land Use at the GROA
7.4	6	3	Other Types of Legal Interests
8.0	28	14	Quality Assurance (QA) Records
8.1	22	11	QA Records for Site Characterization
8.2	20	10	QA Records for Design and Construction
8.3	0	0	QA Records Including Records Covering Operations, Permanent Closure, Decontamination, and Decommissioning
8.4	7	4	QA Records for All Relevant Research Activities
9.0	20	10	Emergency Planning
10.0	20	10	Radiation Protection
10.1	19	10	Ensuring that Radiation Exposures Are As Low As Is Reasonably Achievable (ALARA)
10.2	20	10	Radiation Sources
10.3	19	10	Radiation Protection Design Features
10.4	20	10	Estimated Onsite Dose Assessment
10.5	14	7	Health Physics Program
10.6	17	9	Estimated Offsite Dose Assessment
11.0	14	7	Any Alternatives Considered (e.g., design interpretations, models)
12.0	18	9	Information for Preparation of a Geologic Repository Environmental Impact Statement
12.1	20	10	Environmental
12.2	21	11	Socioeconomic
12.3	20	10	Transportation

Figure 4.5: 3.69 Topical Guidelines 5.0-12.0

created by professional thesaurus builders for the Licensing Support Network (LSN) database and donated to ISRI [50, 52].

There exist international and national standards for the construction of thesauri such as the ANZI [39] and ISO [40] guidelines. According to those standards, there are three basic inter-term relationships: equivalence, hierarchical, and associative [1]. Equivalent terms include synonyms, abbreviations, and specially coded terms. For example, in the LSN thesaurus, *US DOD* is related in this way to *Department of Defense.* Usually, one term in this relation is a preferred usage and the other non-preferred. In the LSN thesaurus, *Department of Defense* is preferred to the abbreviation *US DOD.* The preferred term is prefixed by *USE* and the non-preferred by *UF.* Displayed as a traditional thesaurus, the entries for these terms might appear as:

US DOD

USE Department of Defense

Department of Defense

UF US DOD

The second type of inter-term relationship is the hierarchical. Hierarchical terms are related with respect to levels of superordination and subordination. The superordinate term will be prefixed with *BT* for "broader than." So, the term *Metamorphic Rocks* will be prefixed with *BT* in relation to the subordinate *Amphibolites.* The prefix *NT* appears before subordinate terms. These terms would appear in a thesaurus as:

Metamorphic Rocks

NT Amphibolites

Gneisses

Granulites

Marble, etc.

Amphibolites

BT Metamorphic Rocks

The third relationship defined by thesaurus standards is the associative relationship. This group is reserved for terms which have some kind of conceptual relationship but do have an equivalent meaning and are not hierarchically related. Terms which are associatively related are marked by the abbreviation *RT* for "related to." In the LSN thesaurus, the phrase *Metamorphic Rocks* is related to *Basement Rock* in this way. Their thesaurus entries would appear as:

Metamorphic Rocks

RT Basement Rock

Basement Rock

RT Metamorphic Rocks

Several thesauri were extracted from the LSN thesaurus which is stored as a relational database. The four used for experiments were the BT thesaurus, NT thesaurus, RT thesaurus, and the UF thesaurus. The first consists of all terms which have the hierarchical "broader than" relationship, and the second, the terms with the "narrower than" hierarchical relationship. The third contains "related to," that is, associatively related, terms, which are neither synonyms nor hierarchically related. The UF thesaurus contains synonyms in the "use for" relationship.

Terms were added to the document representations created by C-KANT from these thesauri. For each term or phrase found in a document, if a related term or phrase from a thesaurus was found, this term or phrase was added to the document.

For example, if the term *amphibolites* is found in a document, and the BT Thesaurus is used, then the term *metamorphic rocks* is added to the document. Thus, using the BT thesaurus amounts to adding terms which are conceptually broader to the collection while adding terms from the NT thesaurus adds conceptually narrower terms. Intuitively one might think that adding BT terms to a document should help categorization since adding, say, *metamorphic rocks* to one document that contains the term *marble* and also to another containing *amphibolites* should make the documents appear more similar to one another. In the same way, adding NT to a document might make a document concerned with a general concept such as *metamorphic rocks* more similar to more specific documents which only discuss *marble.*

There are also *levels* to a thesaurus which may or may not affect categorization. For example, if *animals* appears in a document the NT thesaurus would add *vertebrates.* If consulted again, the same thesaurus will now add all NT-related terms for *vertebrates*, such as *amphibians*, *birds*, etc. Once these terms are added, all the NT-related terms *amphibians*, *birds*, and so forth will be added as well.

C-KANT is able to train using a user-specified number of levels. In the following we will generally distinguish tests with respect to the number of levels consulted in the thesaurus. So, for example, BT2 will mean that 2 levels of BT terms were added to the documents. Hence part of our results will indicate if the number of levels added has a significant affect on the performance of C-KANT.

This chapter described the basic testing environment for the experiments. It defined the basic evaluation measures, precision, recall, and F_β as well as their microaverages and macroaverages. It then presented the LSN prototype document collection as well as the thesauri used to enhance C-KANT. The next chapter provides the results of the tests.

CHAPTER 5

TEST RESULTS

This chapter presents the results of experiments using a thesaurus aided rule-based categorizer. The experiments were performed using two types of rule sets: thesaurus-homogeneous and thesaurus-heterogeneous sets. The former rule sets tended to degrade performance. The heterogeneous rule set can in some cases improve results but unfortunately not to a significant extent. These results suggest that thesaurus-aided learning does not increase the effectiveness of rule-based categorization for OCR text.

In the following the performance of C-KANT is first compared to that of another categorizer which was tested over OCR text data. Next the concept of a thesaurus-homogeneous rule set is defined. The results of the tests using homogeneous rule sets are then presented and are followed by a category by category analysis of these results. This analysis inspired the construction of a thesaurus-heterogeneous rule set, which is then defined. Finally the results of tests over the heterogenous rule sets are presented and analyzed.

Thesaurus-Homogeneous Rule Sets

The Small-DOE collection defined in the last chapter was used for optimizing C-KANT's performance and to determine if C-KANT functioned at a level acceptably close to other categorizers. ISRI has performed tests on the Small-DOE collection using McCallum's BOW, a statistical categorizer [31]. In those tests, the best result obtained was 97.06 in "average accuracy" [51]. Average accuracy seems to mean

Microaverage Changes						
	Recall	Precision	F_1	Δ Recall	Δ Precision	Δ F_1
def	0.941	0.564	0.706			
BT1	0.949	0.557	0.702	+ 0.005	− 0.007	− 0.004
BT3	0.941	0.541	0.687	+ 0.000	− 0.016	− 0.019
NT1	0.938	0.557	0.699	− 0.003	− 0.007	− 0.007
NT3	0.932	0.564	0.703	− 0.009	+ 0.000	− 0.003
RT1	0.936	0.535	0.681	− 0.005	− 0.029	− 0.025
RT3	0.949	0.542	0.690	+ 0.005	− 0.022	− 0.016
UF1	0.954	0.556	0.702	+ 0.013	− 0.008	− 0.004
UF3	0.951	0.554	0.701	+ 0.010	− 0.010	− 0.001
UN1	0.940	0.552	0.696	− 0.001	−0.012	− 0.010
Macroaverage Changes						
	Recall	Precision	F_1	Δ Recall	Δ Precision	Δ F_1
def	0.941	0.590	0.725			
BT1	0.945	0.577	0.717	+ 0.004	− 0.013	− 0.008
BT3	0.938	0.572	0.711	− 0.003	− 0.018	− 0.014
NT1	0.932	0.578	0.713	− 0.009	− 0.012	− 0.012
NT3	0.926	0.592	0.722	− 0.015	+ 0.002	− 0.003
RT1	0.933	0.561	0.700	− 0.008	− 0.029	− 0.025
RT3	0.948	0.571	0.713	+ 0.007	− 0.019	− 0.012
UF1	0.951	0.585	0.724	+ 0.010	− 0.005	− 0.001
UF3	0.949	0.582	0.722	+ 0.008	− 0.008	− 0.003
UN1	0.935	0.578	0.714	− 0.006	− 0.012	− 0.011
def = default, no thesaurus expansions						
BTn = nth level BT thesaurus; NTn = nth level NT thesaurus						
RTn = nth level RT thesaurus; UFn = nth level UF thesaurus						
UNn = nth level BTn + NTn + RTn + UFn						

Figure 5.1: Changes in Recall, Precision, and F_1 over Big-DOE

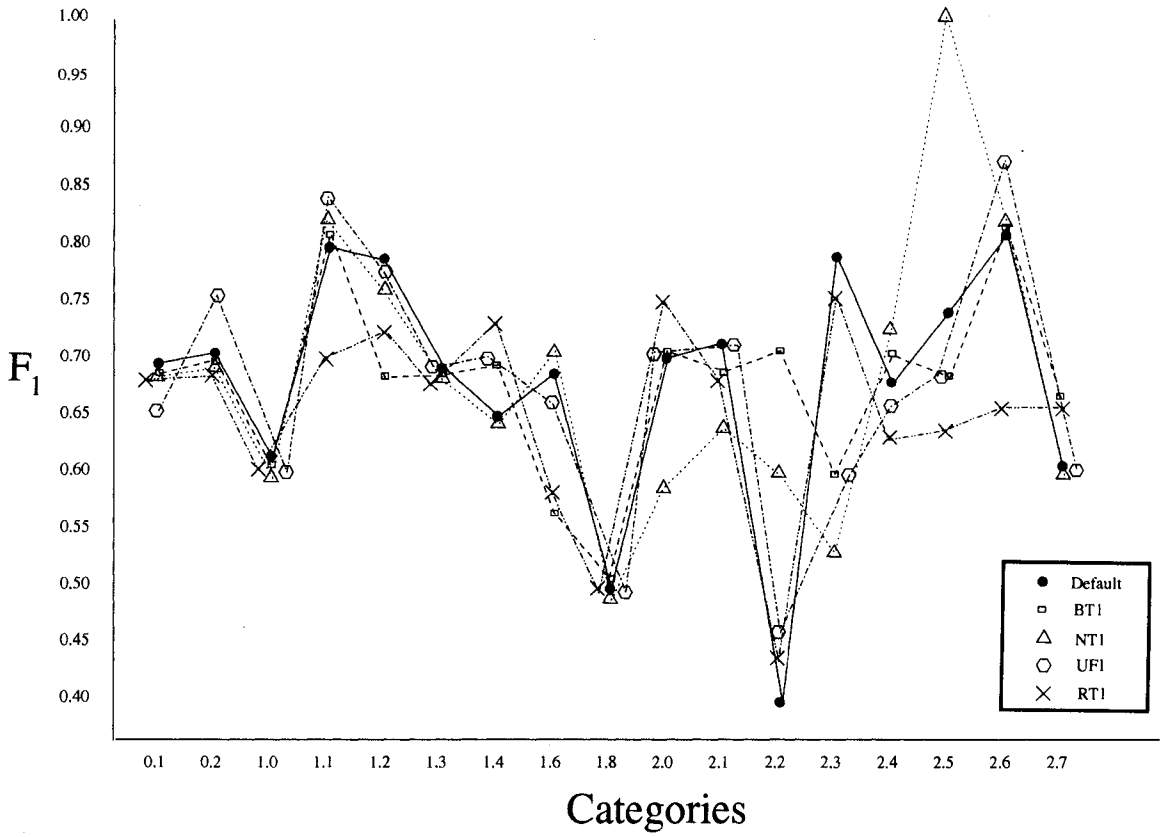

Figure 5.2: Comparison of F_1 Over Categories 0.1 to 2.6

microaveraged precision multiplied by 100. C-KANT, using stemming, a short stop-word list, and Ripper optimizations achieved a microaveraged precision of 0.979 with a microaveraged recall of 1.000 and a microaveraged F_1 of 0.989. C-KANT, therefore, seems to be getting very similar results to BOW on OCR collections. It is interesting to note that C-KANT did not use the dimensionality reduction techniques suggested by [51].

In tests on both the Small-DOE and the Big-DOE collections, stemming, a simple stop-word list, and Ripper optimizations were used since experiments indicated that these were optimal settings for C-KANT. Thesaurus terms were added to both training documents and testing documents using the same thesaurus for each. So, for example, when one level of "broader than" terms (BT1) was added to the training documents, the same type was added to the testing documents. Both training and test documents were expanded based on the idea that expanding the documents with

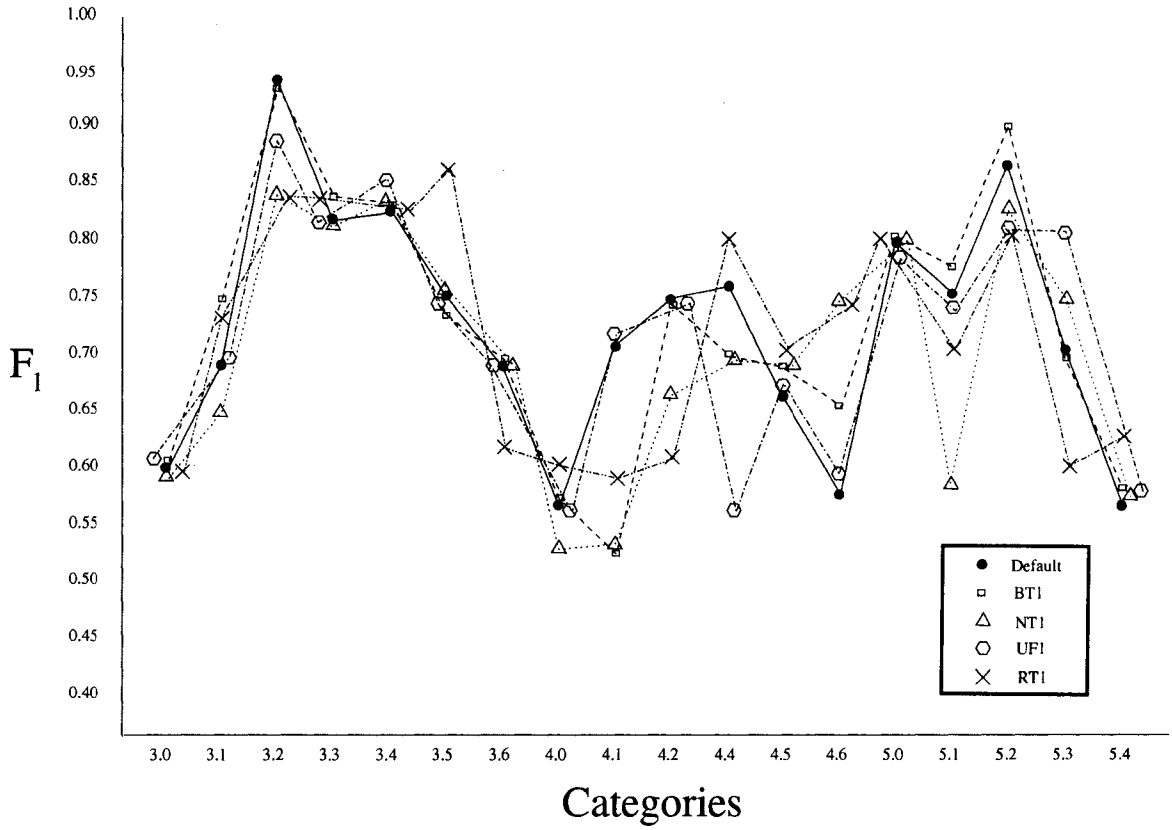

Figure 5.3: Comparison of F_1 Over Categories 3.0 to 5.4

thesaurus terms might tend to cluster documents within a category by introducing terms which would make them more similar. As a result, the hope was that the rule-building software would learn the "clustering" terms and thus be more effective. Testing indicated if the test set was not expanded with the same thesaurus as the training, the categorizer performed very poorly. For this reason, testing documents were expanded with thesaurus terms as well.

Since these rule sets were derived from training documents which were expanded with the same thesaurus for every category, I call them "thesaurus-homogeneous" rule sets. Tests were also performed using rule sets in which rules were learned from texts expanded with differing thesauri. These "thesaurus-heterogeneous" rule sets are discussed in the next section.

Figure 5.1 lists the microaveraged changes in precision, recall, and the F_1 measure for various types of thesaurus expansions. In particular it compares a default test on

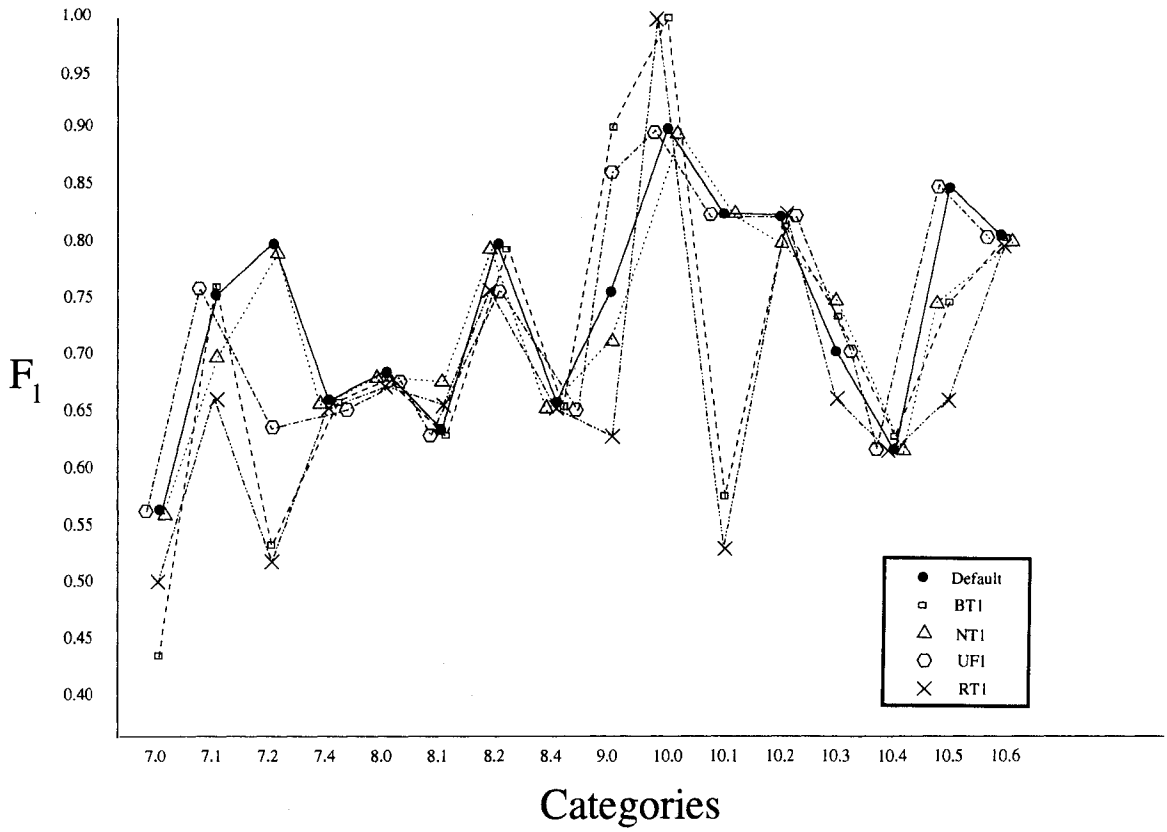

Figure 5.4: Comparison of F_1 Over Categories 7.0 to 10.6

the Big-DOE test collection with tests using expansions at one and three levels of the BT, NT, RT, and UF Thesauri. The default in this case learned rules from texts not expanded by thesauri and the rules were applied to a test set of documents which were not expanded.

The results show that using thesauri in this way tends to worsen the categorizer's effectiveness. Although in the cases of BT1 and RT3 the recall rose somewhat from the default case, this rise was accompanied by a lowering of precision. Thus the idea of using thesauri to help cluster documents seems to be a mistake.

In addition to expanding the texts using each of the four thesauri alone, one test was run using one level of all four. Following Dimitrova [8] I call this UN1 as an abbreviation for a one level "Union" expansion, since it constitutes a "union" of all the thesauri. According to Dimitrova, UN1 did not improve text retrieval. Consistent

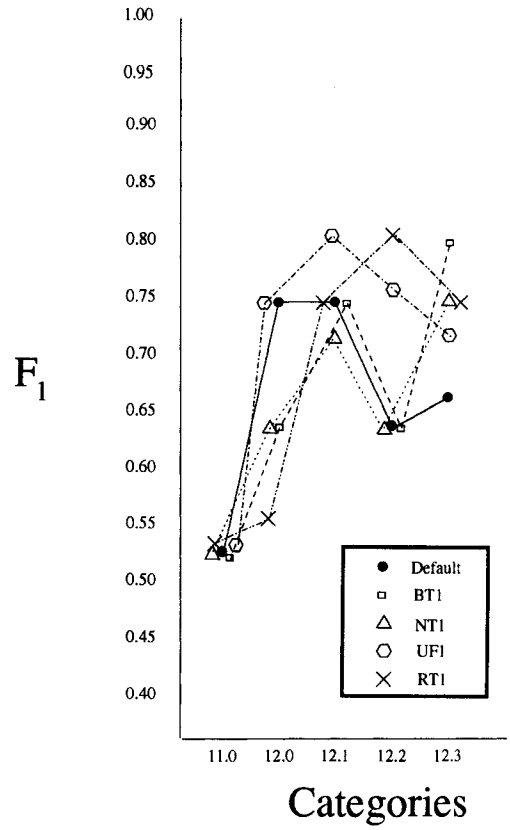

Figure 5.5: Comparison of F_1 Over Categories 11.0 to 12.3

with that result, figure 5.1 shows that UN1 did not improve categorization compared to the default test.

In some cases the differences in the level of the thesaurus used revealed some interesting results. NT3 showed an improvement over NT1 as did RT3 over RT1. The first result seems to suggest that adding the more specific, "narrower" terms to both the training and the test set tends to aid the clustering effect. Adding more and more broader terms seems to worsen the categorizer since BT3 performed worse than BT1.

Explaining the benefit of levels in the RT case is more difficult. Associatively related terms in thesaurus construction, as we saw in the last chapter, are terms which are neither hierarchically related nor are they synonyms. They constitute terms that seem related (in the eyes of the thesaurus builder at least) but do not fit in the other two groups. In fact loading up a thesaurus with such terms is frowned

```
One-way Analysis of Variance Homogeneous Rule Set
   Analysis of Variance for F1
   Source      DF        SS        MS        F        P
   ThesType     9    0.0319    0.0035     0.27    0.982
   Error      540    7.0722    0.0131
   Total      549    7.1041

                                        Individual 95% CIs For Mean
                                        Based on Pooled StDev
   Level        N      Mean     StDev  -------+---------+---------+---------
   def         55    0.7139    0.1041           (-----------*-----------)
   BT1         55    0.7052    0.1119         (-----------*-----------)
   BT3         55    0.6975    0.1195      (-----------*-----------)
   NT1         55    0.7012    0.1064       (-----------*-----------)
   NT3         55    0.7104    0.1117           (-----------*-----------)
   RT1         55    0.6879    0.1067  (-----------*-----------)
   RT3         55    0.7000    0.1201       (-----------*-----------)
   UF1         55    0.7117    0.1066           (-----------*-----------)
   UF3         55    0.6957    0.1426     (-----------*-----------)
   UN1         55    0.7036    0.1095        (-----------*-----------)
                                        -------+---------+---------+---------
   Pooled StDev =    0.1144              0.675     0.700     0.725
```

Figure 5.6: ANOVA for Homogeneous Rule Sets

upon by thesaurus builders since the relation is vaguely defined and adding large numbers of such terms can hurt effectiveness [1]. Thus, it is surprising that there is improvement as levels are added. This result probably cannot be easily duplicated in other thesauri since what terms are considered RT terms tends to be ad hoc.

The macroaveraged measures give similar results to the microaveraged as shown in figure 5.1. Since there is little variation in the number of training and testing documents available for each category, microaveraging and macroaveraging are very close. The most interesting number is that NT3 had a small increase over the default in precision.

However, an analysis of variance (ANOVA) test using the statistical software MINITAB reveals that none of the microaveraged changes in figure 5.1 are significantly large. Output from the MINITAB ANOVA test is shown in figure 5.6. ANOVA

offers a test for the hypothesis (H_0) that the average of the F_1 measure in all of the tests are the same, as opposed to the hypothesis (H_1) that at least two of the averages differ significantly. The very large p-value of 0.982 in figure 5.6 indicates that it is very likely a mistake to reject H_0 [37].

Of particular interest in figure 5.6 is the display of the 95% Confidence Intervals (CIs). These show the range of the most likely values F_1 will take in experiments. The fact that these intervals overlap implies that thesaurus expansion did not significantly effect the performance of the rule-based categorizer.

Thesaurus Heterogeneous Rule Sets

A category by category breakdown of the results from the last section offered some tantalizing prospects. Figures 5.2, 5.3, 5.4, and 5.5 show that in certain categories, thesaurus additions improved results. For example, Figure 5.2 shows that BT1 beat the default by a large margin in category 2.2 while NT1 had better success with category 2.5 than the default.

These category by category results indicate that it might be more useful to apply thesaurus additions on a category by category basis. Thus, if experimentation shows that adding terms from a particular thesaurus is effective for a particular category, then that thesaurus should be used for that category but not necessarily for others. In the case of a rule-based system, this approach can be applied by using rules trained from texts with terms added from a thesaurus for one category while rules for another category would be trained with texts which may be expanded by another thesaurus.

To test this idea, a heterogenous rule set called HodgePodge (HP) was created. HP includes those rules for a given category which were learned from training documents expanded with that thesaurus which gave the best performance in the original, homogeneous test. In those cases in which the default was best, no thesaurus expansion was used. For example, the rule set trained on NT1 expanded documents was

Microaverage Changes						
	Recall	Precision	F_1	Δ Recall	Δ Precision	Δ F_1
def	0.941	0.564	0.706			
HP-NoExp	0.837	0.610	0.706	− 0.104	+ 0.046	+ 0.000
HP-BT1	0.892	0.595	0.714	− 0.049	+ 0.031	+ 0.008
HP-NT1	0.875	0.493	0.631	− 0.066	− 0.071	− 0.075
HP-RT1	0.925	0.435	0.591	− 0.016	−0.129	− 0.115
HP-UF1	0.849	0.601	0.703	− 0.092	+ 0.037	− 0.003
HP-UN1	0.963	0.349	0.512	+ 0.022	− 0.215	− 0.190
Macroaverage Changes						
	Recall	Precision	F_1	Δ Recall	Δ Precision	Δ F_1
def	0.941	0.590	0.725			
HP-NoExp	0.845	0.641	0.729	− 0.096	+ 0.051	+ 0.004
HP-BT1	0.898	0.622	0.735	− 0.043	+ 0.032	+ 0.010
HP-NT1	0.883	0.583	0.702	− 0.050	− 0.007	− 0.023
HP-RT1	0.922	0.532	0.675	− 0.019	− 0.058	− 0.050
HP-UF1	0.855	0.629	0.725	−0.086	+ 0.039	+ 0.000
HP-UN1	0.958	0.491	0.649	+ 0.017	− 0.099	− 0.076

def = default, Neither Training nor Test Set Expanded

HP-NoExp = HP Rule Set + Test Set not Expanded

HP-BT1 = HP Rule Set + Test Set Expanded by BT1

HP-NT1 = HP Rule Set + Test Set Expanded by NT1

HP-RT1 = HP Rule Set + Test Set Expanded by RT1

HP-UF1 = HP Rule Set + Test Set Expanded by UF1

HP-UN1 = Test Set Expanded by BT1, NT1, RT1, UF1

Figure 5.7: Changes Using HodgePodge Rule Sets.

One-way Analysis of Variance Heterogeneous Rule Set

Analysis of Variance for F

Source	DF	SS	MS	F	P
ThesType	6	0.3644	0.0607	2.55	0.020
Error	378	9.0187	0.0239		
Total	384	9.3831			

Level	N	Mean	StDev	Individual 95% CIs For Mean Based on Pooled StDev
def	55	0.7139	0.1041	(-------*-------)
NE	55	0.7024	0.1322	(-------*--------)
BT1	55	0.7167	0.1439	(-------*--------)
NT1	55	0.6766	0.1733	(-------*--------)
RT1	55	0.6766	0.1733	(-------*--------)
UF1	55	0.7015	0.1417	(-------*-------)
UN1	55	0.6210	0.1945	(-------*-------)

Pooled StDev = 0.1545

0.600 0.650 0.700 0.750

Figure 5.8: ANOVA for Heterogeneous Rule Sets

used for category 2.5, and the rule set trained on documents with no thesaurus expansion was used for category 3.2. HP was tested over the test set in an unexpanded form (HP-NoExp), and also with the test set expanded with NT1, BT1, UF1, RT1, and UN1. The results are in figure 5.7.

HP did best on the BT1 expanded test set (HP-BT1), and this was the only case where a thesaurus aided rule set beat the default. However, the margin of improvement was not statisically significant under an analysis of variance (ANOVA) test (figure 5.8). One might still argue that HP-BT1 is a better categorizer than the default because its precision is higher. However, an ANOVA test indicates that the improvement in precision is not a significantly large improvement. An ANOVA test supplemented with pairwise comparisions indicated that the only significant difference among the tests in figure 5.7 is that using UN1 to expand the test set is significantly worse compared to the default.

It may be that HP does not significantly improve results because although the rules reflect "optimal" thesaurus expansions, the test documents cannot be expanded by the appropriate thesaurus. In other words, using NT1 to expand the rules for category 2.5 might be fine if the test documents in that category were also so expanded. However, to do so would amount to knowing the category of a test document prior to categorization, which would be, in effect, cheating.

In this chapter two attempts were made to improve text categorization using thesaurus-homogeneous rule sets and a thesaurus-heterogeneous rule set. Homogeneous rule sets were trained on a collection of training documents all expanded by the same thesaurus. The Heterogeneous rule set contained a mixture of rules. Some rules were trained from documents expanded with one thesaurus and other rules (for other categories) were based on documents trained from a different thesaurus. In both cases it was shown that thesaurus aided learning for the rule sets did not improve their categorization to a significant extent. This suggests that using thesauri to aid rule-based categorization of OCR texts is not a good way to improve such categorizers.

CHAPTER 6

CONCLUSION AND FURTHER RESEARCH

The tests from the previous chapter show that a query expansion approach to rule-based automatic text categorization using domain-dependent thesauri will not improve the categorization of OCR texts. This conclusion raises several new questions. For example, would these results apply to other types of categorizers? It would be interesting to extend this study not only to statistical and decision tree classifiers, but also to rule-based categorizers which learn rules differently from IREP and RIPPER, such a CHARADE and TDIS [34].

In addition, as mentioned earlier, C-KANT made use of common dimensionality reduction techniques such as stemming and stop word removal. However, it has been recommended in [51] that more extensive reductionality techniques should be applied when categorizing OCR text. These were not applied in C-KANT's case. C-KANT could be modified to apply such dimensionality techniques to see if these techniques do aid performance. In fact, some rules made use of "garbage" words learned from the OCR text. For example, one rule stated that if the term *ac4vantage* appeared in a document, that document should be placed in category 1.2. Another claimed that if *controlquote* is in a document, then the document belongs to category 8.2.

More aggressive reductionality techniques may reduce the number of such rules. The results in [51] were derived from tests using the statistical categorizer BOW. It would be interesting to know if rule-based categorizers would also perform better with more aggressive dimensionality reduction.

More experimentation with C-KANT's parameters may also improve results. For

example, only one of several formulations of the minimum description length (MDL) were used in the RIPPER optimizations [43]. It would be interesting to determine by experiment if other formulations would improve or worsen results.

Another path for future research is suggested by Junker in [22]. Junker would replaced a rule such as

$$t_1 \in d_j \wedge t_2 \in d_j \wedge \cdots \wedge t_n \in d_j \rightarrow d_j \in c_i$$

with something like

$$\theta_1 \in d_j \wedge t_2 \in d_j \wedge \cdots \wedge t_n \in d_j \rightarrow d_j \in c_i$$

where θ_1 has a thesaurus relationship to t_1, and the second rule performs better than the first in terms of some measure such as *information gain* over the test examples. Junker's approach achieved only mixed results, but it represents an interestingly different kind of thesaurus expansion to those used in our study.

Finally a more ambitious approach to rule-building may have better results for text categorization as well. CLIPS allows for a much more sophisticated representation of documents using its frame structure [14]. In our study, documents were only represented as a "bag of terms" occurring in the "body" of the documents. Some studies have incorporated the relative position of terms into rules, but with little improvement in performance [4]. Perhaps describing a document in a more complex fashion using information from the OCR process itself could lead to a better categorizer.

BIBLIOGRAPHY

[1] J. Aitchison, A. Gilchrist, and D. Bawden. Thesaurus construction and use: A practical manual. Fitzroy Dearborn, 4th edition, 2000.

[2] C. Apte, F. Damerau, and S. M. Weiss. Automated learning of decision rules for text categorization. ACM Transactions on Information Systems 1994, 12(3):233–251, 1994.

[3] W. W. Cohen. Fast effective rule induction. In International Conference on Machine Learning, pages 115–123, 1995.

[4] W. W. Cohen. Text categorization and relational learning. In Proceedings of International Conference on Machine Learning ICML-95, pages 124–132, 1995.

[5] W. W. Cohen and Y. Singer. Context-sensitive learning methods for text categorization. ACM Transactions on Information Systems, pages 141–173, 1999.

[6] Nuclear Regulatory Commission. Regulatory guide 3.69. URL: <http://www.nrc.gov/NRC/RG/03/03-069.html>(viewed June 20, 2001), 1996.

[7] I. Dagan, Y. Kerov, and D. Roth. Mistake-driven learning in text categorization. In Claire Cardie and Ralph Weischedel, editors, Proceedings of the Second Conference on Empirical Methods in Natural Language Processing, pages 55–63. Association for Computational Linguistics, Somerset, New Jersey, 1997.

[8] Elena Dimitrova. Retrieval effectiveness for OCR text using thesauri. Master's thesis, University of Nevada, Las Vegas, 1999.

[9] S. Dumais, J. Platt, D. Hecherman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In Proceedings of Conference on Information and Knowledge Management CIKM 98, pages 148–155, 1998.

[10] W. B. Frakes. Introduction to information storage and retrieval systems. In William B. Frakes and Ricardo Baeza-Yates, editors, Information Retrieval: Data Structures and Algorithms, pages 1–12. Prentice-Hall, 1992.

[11] W. B. Frakes. Stemming algorithms. In William B. Frakes and Ricardo Baeza-Yates, editors, Information Retrieval: Data Structures and Algorithms, pages 131–160. Prentice-Hall, 1992.

[12] N. Fuhr and C. Buckley. A probabilistic learning approach for document indexing. Information Systems, pages 223–248, 1991.

[13] J. Fürnkranz and G. Widmer. Incremental reduced error pruning. In Proceedings of the 11th Annual Conference on Machine Learning, pages 70–77, 1994.

[14] J. Giarratano and G. Riley. Expert Systems: Principles and Programming. ITP, 3rd edition, 1998.

[15] D. Harman. Information Retrieval, Data Structures and Algorithms, chapter Relevance Feedback and Other Query Modification Techniques, pages 241–263. Prentice Hall, Englewood Cliffs, NJ 07632, 1992.

[16] P. J. Hayes, P. M. Andersen, I. B. Nirenburg, and L. M. Schmandt. TCS: a shell for content-based text categorization. In Proc. of CAIA-90, 6th IEEE Conf. on Artificial Intelligence Applications, pages 320–326, Santa Barbara, CA, 1990.

[17] P. J. Hayes and S. P. Weinstein. Construe/tis: A system for content-based indexing of a database of news stories. In Second Annual Conference on Innovative Applications of Artificial Intelligence, 1990.

[18] Yahoo! Inc. What is the yahoo! directory? URL: <http://help.yahoo.com/help/us/dir/dir-01.html> (viewed Jan. 18, 2002).

[19] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In Proc. of ECML-98, 10th European Conf. on Machine Learning, pages 137–142, Chemnitz, DE, 1998.

[20] T. Joachims. Making large-scale support vector machine learning pratical. In B. Schölkopf, C. Burges, and A. Smola, editors, Advances in Kernel Methods, pages 169–184. MIT Press, 1999.

[21] T. Joachims. Transductive inference for text classification using support vector machines. In Proceedings of ICML-99, 16th International Conference on Machine Learning, 1999.

[22] M. Junker and A. Abecker. Exploiting thesaurus knowledge in rule induction for text classification. In Ruslan Milkov, Nicolas Nicolov, and Nilokai Nikolov, editors, Proceedings of RANLP-97, 2nd International Conference on Recent Advances in Natural Language Processing, pages 202–207, Tzigov Chark, BL, 1997.

[23] L. Kaufmann. Solving the quadratic programming problem arising in support vector classification. In B. Schölkopf, C. Burges, and A. Smola, editors, Advances in Kernel Methods, pages 147–168. MIT Press, 1999.

[24] D. D. Lewis. Evaluating text categorization. In Proceedings of the Speech and Language Workshop, pages 312–318, 1991.

[25] D. D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In Proc. of ECML-98, 10th European Conf. on Machine Learning, pages 4–15, Chemnitz, Germany, 1998.

[26] D. D. Lewis and M. Ringuett. Comparison of two learning algorithms for text categorization. In Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94), 1994.

[27] D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka. Training algorithms for linear text classifiers. In SIGIR '96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 298–306, 1996.

[28] D. D. Lewis, D. L. Stern, and A. Singhal. ATTICS: a software platform for online text classification. In Proc. of SIGIR-99, 22nd ACM Intl. Conf. on Research and Development in Information Retrieval, pages 267–268, Berkeley, CA, 1999.

[29] H. Li and K. Yamanishi. Text classification using ESC-based stochastic decision lists. In Proceedings of CIKM-99, 8th ACM International Conference on Information and Knowledge Management, pages 122–130, Kansas City, MO, 1999.

[30] Y. H. Li and A. K. Jain. Classification of text documents. The Computer Journal, 41(8):537–546, 1998.

[31] A. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. URL: <http://www.cs.cmu.edu/~mccallum/bow> (viewed March 6, 2002), 1996.

[32] T. M. Mitchell. Machine Learning. McGraw-Hill, 1997.

[33] I. Moulinier. Is learning bias an issue on the text categorization problem? Technical report, LAFORIA-LIP6, Universite Paris VI, 1997.

[34] I. Moulinier and J. Ganascia. Applying an existing machine learning algorithm to text categorization. In Connectionist, statistical, and symbolic approaches to learning for natural language processing, pages 343–354. Springer Verlag, 1996.

[35] I. Moulinier, G. Raškinis, and J. Ganascia. Text categorization: a symbolic approach. In Proceedings of SDAIR-96, 5th Annual Symposium on Document Analysis and Information Retrieval, pages 87–99, Las Vegas, NV, 1996. Information Science and Research Institute.

[36] H. T. Ng, W. B. Goh, and K. L. Low. Feature selection, perceptron learning, and a usability case study for text categorization. In 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97), pages 67–73, 1997.

[37] G. Oehlert. A First Course in Design and Analysis of Experiments. W.H. Freeman, 2000.

[38] Library of Congress. Library of Congress classification outline. URL: <http://www.loc.gov/catdir/cpso/lcco/lcco.html> (viewed Jan. 18, 2002).

[39] National Information Standards Organization. ANZI/NISO Z39.19:1993. Guidelines for the Construction, Format, and Managements of Monolingual Thesauri. NISO Press, 1994.

[40] International Organization for Standardization. ISO 2788: Guidelines for the establishment and development of monolingual thesauri. ISO, 1986.

[41] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In J. Principe, L. Gile, N. Morgan, and E. Wilson, editors, Neural Networks for Signal Processing VII: Proceedings of the 1997 IEEE Workshop, pages 276–285. IEEE, 1997.

[42] DMOZ Open Directory Project. About the open directory project. URL: <http://dmoz.org/about.html> (viewed Jan. 18, 2002).

[43] J. Quinlan. Mdl and categorical theories (continued). In Proceedings of the 12th International Conference on Machine Learning, pages 464–470, Lake Tahoe, CA, 1995.

[44] J. R. Quinlan. Induction of decision trees. Machine Learning, pages 81–106, 1986.

[45] J. R. Quinlan. Learning logical definitions from relations. Machine Learning, 1990. Introduction to FOIL.

[46] R. Savely and C. Culbert. CLIPS Reference Manual, Version 6.10, 1998. URL: <http://www.ghgcorp.com/clips/download/documentation/> (viewed Jan. 28, 2002).

[47] F. Sebastiani. Machine learning in automated text categorization. ACM Computing Surveys, 2002. Accepted for publication.

[48] P. Srinivasan. Thesaurus construction. In William B. Frakes and Ricardo Baeza-Yates, editors, Information Retrieval: Data Structures and Algorithms, pages 161–218. Prentice-Hall, 1992.

[49] Gilbert Strang. Linear algebra and its applications. Harcourt, Brace, Jovanovich, 3rd edition, 1988.

[50] Kazem Taghva, Julie Borsack, and Allen Condit. The effectiveness of thesauri-aided retrieval. In Proc. IS&T/SPIE 1999 Intl. Symp. on Electronic Imaging Science and Technology, San Jose, CA, January 1999.

[51] Kazem Taghva, Thomas A. Nartker, and Julie Borsack. Recognize, categorize, and retrieve. In Proc. of the Symposium on Document Image Understanding Technology, pages 227–232, Columbia, MD, April 2001. Laboratory for Language and Media Processing, University of Maryland.

[52] Kazem Taghva, Tom Nartker, Julie Borsack, and Allen Condit. Unlv-isri document collection for research in OCR and information retrieval. In Proc. IS&T/SPIE 2000 Intl. Symp. on Electronic Imaging Science and Technology, San Jose, CA, January 2000.

[53] Kazem Taghva, Tom Nartker, Julie Borsack, Steve Lumos, Allen Condit, and Ron Young. Evaluating text categorization in the presence of OCR errors. In Proc. IS&T/SPIE 2001 Intl. Symp. on Electronic Imaging Science and Technology, pages 68–74, San Jose, CA, January 2001.

[54] Kazem Taghva and Eric Stofsky. Ocrspell: An interactive spelling correction system for OCR errors in text. Intl. Journal on Document Analysis and Recognition, 3(3):125–137, March 2001.

[55] H. Taira and M. Haruno. Feature selection in SVM text categorization. In Proceedings of AAAI-99, 16th Conference of the American Association for Artificial Intelligence, pages 480–486, 1999.

[56] Vladimir N. Vapnik. The Nature of Statistical Learning Theory. Springer Verlag, 2 edition, 2000.

[57] E. Weiner, J. O. Pedersen, and A. S. Weigend. A neural network approach to topic spotting. In Proceedings of the Fourth Annual Symposium on Documeent Analysis and Information Retrieval (SDAIR'95), pages 317–332, Las Vegas, NV, 1995.

[58] C. Wenzel and R. Hoch. Text categorization of scanned documents applying a rule-based approach. In Fourth Annual Symposium on Document Analysis and Information Retrieval (SDAIR), pages 333–346, Las Vegas, NV, 1995. Information Science and Research Institute (ISRI).

[59] I. Witten, A. Moffat, and T. Bell. Managing Gigabytes: Compressing and indexing documents and images. Morgan Kaufmann, 2nd edition, 1999.

[60] Y. Yang. An evaluation of statistical approaches to text categorization. Information Retrieval, 1:69–90, 1999.

[61] Y. Yang and C. G. Chute. A linear least squares fit mapping method for information retrieval from natural language texts. In Proceedings of the 14th International Conference on Computational Linguistics (COLING 92), pages 447–453, 1992.

[62] Y. Yang and C. G. Chute. An example-based mapping method for text categorization and retrieval. ACM Transactions on Information Systems, 12(3):252–277, July 1994.

[63] Y. Yang and X. Liu. A re-examination of text categorization methods. In 22nd Annual International SIGIR, pages 42–49, 1999.

VITA

Graduate College
University of Nevada, Las Vegas

Jeffrey Scott Coombs

Local Address:
6701 Squaw Mt. Dr. #203
Las Vegas, NV 89130

Degrees:
Bachelor of Science, Computer Science, 2000
University of Nevada, Las Vegas

Publications:
Kazem Taghva and Jeffrey Coombs, Hairetes: A Search Engine for OCR Documents, Technical Report 2002-01, Information Science Research Institute, University of Nevada, Las Vegas, 2002.

Thesis Title:
Thesaurus Aided Learning for Rule-Based Categorization of OCR Text

Thesis Examination Committee:
Chairperson, Kazem Taghva, Ph. D.
Committee Member, Dr. Thomas Nartker, Ph. D.
Committee Member, Dr. John Minor, Ph. D.
Graduate Faculty Representitive, Dr. Ashok Singh, Ph. D.