Incremental Document Classification
In a Knowledge Management Environment

by

Shun Zhou

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Incremental Document Classification in a Knowledge Management Environment

Shun Zhou

Master of Science

Department of Computer Science

University of Toronto

May 2001

This thesis studies document classification problem in a knowledge management environment. Most of the published studies of document classification focus on batch classifiers. Although these classifiers can have high performance, they require a batch training with a large number of sample documents. Such a requirement is usually unrealistic in a knowledge management environment. Moreover, these classifiers cannot improve performance or adjust themselves after the initial batch training, leading to low performance in a changing environment. The thesis proposes incremental classifiers as a solution. The batch naive Bayes (NB) and k-nearest-neighbors (kNN) classifier are adapted into incremental classifiers. Adapting published implementations for the batch NB and kNN classifiers, a series of experiments are designed to find an efficient implementation for the incremental versions.

Acknowledgements

First of all, I would like to thank my supervisor, John Mylopoulos. I am deeply grateful not only to his continuous encouragement and invaluable directions to my study, but also to his considerate assistance to other aspects of my life. Secondly, I would like to thank Grigoris Karakoulas, the second reader of my thesis. His broad knowledge on machine learning and generous donation of time make it possible for me to accomplish this study.

It is an exciting and valuable experience to work in the EXIP project team, including Attila Barta, Raoul Jarvis, Greg McArthur and Patricia Rodriguez Gianolli. Thanks to all of them for the help that I have received.

I am indebted to many people in our department. Jianguo Lu is the person whom I will go to whenever I need consultation or reference books. Samir Girdhar helped me solve a great deal of Unix problems. Greg McArthur, Manual Kolp, Vasiliki Kantere, and Patricia Rodriguez Gianolli helped me with data preparation for the document classification with respect to the auto manufacturer model. Jianguo Lu, Patricia Rodriguez Gianolli, and Manual Kolp kindly allowed me to share their CPU's, making it possible to finish my time and space consuming experiments.

Thanks to Yiming Yang and Thomas Galen Ault in CMU for kindly sharing with me their stemming algorithms. Thanks to David D. Lewis in Bell Labs for sending me his recommendations on the stopwords.

Finally, I would like to thank my parents and my younger sister for their endless love and support. This dissertation is dedicated to them.

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter 1: Introduction

1.1. Background

The information revolution has led to a pressing need to support efficient information retrieval, management, and analysis for knowledge workers. For example, in a large corporation, strategic business analysts keep track of trends that are relevant to their organization and its strategic objectives. They monitor news stories and other reports as they become available, looking for evidence that these objectives remain on track, or have encountered obstacles. To accomplish their task, analysts have to search, download, and organize a large amount of information manually although there has been a significant improvement on network connectivity and computing platforms. Moreover, analysts do not have tools for sharing their knowledge and collaborating their daily work, leading to inconsistency and redundant work.

In order to solve these problems, a number of technologies have been studied, such as data retrieval and analysis for structured data, semi-structured data, and unstructured data. These technologies are bundled into a solution through an integration architecture often called an enterprise information portal (EIP) [Shilakes98]. Typically, EIPs offer facilities for search, retrieval, analysis and organization of structured data and documents, along with facilities for network connectivity and computing platform interoperation.

However, the problems indicated above are not fully addressed by current EIP technologies. We are currently developing the executive information portal (EXIP), a prototype system intended to make two contributions. Firstly, the system uses a semantic model to capture knowledge shared by a group of collaborating knowledge

workers. The model is used to drive information retrieval, classification, and analysis. For example, a telecommunication company can use a semantic model which consists of strategic goals, events, actors, and links. While monitoring trends within the company and the market, analysts map retrieved information to the elements in the model. A news document from CNN could contribute positive evidence to the event, "competitor buys content provider", or a report from Forrester Research could contribute negative evidence to the goal, "become the largest ISP in Canada". Working within the context of the model, analysts organize information consistently, preventing them from redundant retrieval, management, and analysis.

Secondly, the EXIP provides a toolset which can evolve the model semi-automatically and requires minimal feedback from users. Specifically, the model can be modified by users, or automatically, using machine learning techniques. Moreover, the classification algorithms, which are used to classify documents automatically, evolve as well, along with the model.

Within the context of the EXIP project, this thesis focuses on document classification. Another M.Sc. thesis [Jarvis2001] presents our work on the semantic model. [Mylopoulos2001] offers a comprehensive overview of the EXIP project.

1.2. Contributions of the Thesis

The document classification component automatically classifies documents from external sources with respect to the semantic model. There already exist many sophisticated document classifiers. Most of these, however, focus on batch training. Given the requirements of the EXIP, these classifiers require an unrealistic large number of sample documents to achieve high performance. Moreover, these classifiers cannot improve performance or adjust themselves after the batch training. This is undesirable, given our objective of supporting evolution for the classification

component.

The main contribution of the thesis is to adapt existing classifiers to meet two requirements:

(1) The EXIP classifier will be able to continuously improve its performance by interacting with users;

(2) The classifier will achieve acceptable performance with a small number of sample documents.

Several different types of classifiers can potentially be adapted to meet our requirements. This thesis focuses on using naive Bayes (NB) [McCallum1998] [Baker1998] and k nearest neighbors (kNN) [Dasarathy1991] [Masand1992] classifiers, and shows how such adaptations can be accomplished. The incremental classifiers proposed in the thesis share a common theoretical basis with batch classifiers. The algorithms of batch classifiers (NB and kNN) are presented in pseudo-code form. Also in pseudo-code, the thesis illustrates the particular adaptations to each batch classifier, making them incremental.

A second contribution of the thesis consists of a large number of experiments. These experiments are designed to characterize the performance of incremental adaptations of each batch classifier. These experiments are also designed to look for the best implementation options for each incremental classifier. Such options include which term weighting scheme to use, whether to use feature selection, etc.

A final contribution of the thesis is to study how the classification component can be integrated into the EXIP system. In particular, the thesis explains how the semantic model can be converted into a flat classification taxonomy, and how to define a proper document schema to articulate the collaboration among the classification component, the document management component, and the XML document server. For demonstration purpose, the thesis builds a semantic model for the auto industry, and

uses parts of the Reuters document collection to simulate the working environment of strategic business analysts.

1.3. Organization of the Thesis

Chapter 2 presents theories of document classification and known algorithms for the batch NB and kNN classifiers. Chapter 3 proposes adaptations of batch classifiers into incremental ones. Chapter 4 describes our experiments designed to discover an efficient implementation of the incremental classifiers. Chapter 5 describes the architecture of the EXIP and document classification component. Moreover, the chapter introduces the clustering algorithm which is used to search for semantic clusters in the documents under a model element. Chapter 6 describes experiments designed to simulate the working environment of the document classification component in an operating EXIP system. Experiment results are also presented and analyzed. Chapter 7 summarizes the contributions of the thesis and discusses future work.

Chapter 2: Theories of Document Classification

This chapter presents the theoretic baseline of machine learning techniques applied to the problem of document classification. Firstly, we start with a brief introduction of document classification and how machine learning techniques can automate it. Secondly, a number of fundamental concepts, such as the vector space model, term weighting, and feature selection, are introduced. Thirdly, we give detailed descriptions of kNN and NB classifiers which will be used in our experiments and case study.

2.1. Document Classification and Machine Learning Techniques

Document classification is the task of assigning documents to two or more predefined categories. For example, a news document generated in the Reuters news agency is classified into a number of topics, such as "crude oil", "foreign currency exchange", "acquisition" and so on. If a document can be assigned to more than one category, the process is called multi-category classification. Whereas, if a document could be assigned to only one category, it is called singular-category classification. Multi-category classification is more common than singular-category classification.

In industrial practice, document classification is usually done manually by one or more knowledge workers. They have the required expertise to understand documents, and they are familiar with the organization's business so as to understand the purpose of the documents. It is not difficult to understand why this job is very time-consuming and error-prone.

Many approaches have been proposed to make document classification automated. The Carnegie Group used a rule-based, expert-system approach to build a text

categorization shell. [Hayes1990] Although the system performed well, it took 2 man/year to find optimal classification rules. Moreover, such a system is difficult to reuse, because the classification rules only work for a particular document collection. If the system is applied to a different collection, another set of classification rules have to be developed. Due to these problems, [Ng1997] used an automatic machine learning approach based on perceptron learning. The automatic approach performed not as well as the rule-based approach. However, by manually modifying and augmenting the set of words to be used as features in a topic categorizer, the authors achieved accuracy very close to the rule-based approach. More recent studies of document classification have advanced to ones using purely automatic approaches. These approaches do not require any experts to generate classification rules or modify a feature set. Instead, a number of sample documents are classified and labeled manually for training purposes. These approaches extract knowledge from the training set and then classify incoming documents automatically.

How can a computer program acquire knowledge to perform automatic document classification? There already exist a large -- and increasing -- number of machine learning techniques which can do just that and which have been applied to document classification. Among them we note nearest neighbor [Dasarathy1991] [Masand1992] [Tomek1976], inductive rule learning [Craven1998], neural networks [Wiener1995] [Ng1997], support vector machines (SVM) [Joachims1998], decision trees [Quinlan1996], linear least-squares fit (LLSF) [Yang1994], naïve Bayes (NB) [McCallum1998][Baker1998] classifiers, etc. A document classification system is often named after the machine learning technique that it uses. For example, a kNN classifier is a document classification system which uses the nearest neighbors technique.

[Yang1999] compared most of the commonly used classifiers using the standard benchmark -- Reuters-21578. The authors conclude that the kNN, SVM, and LLSF classifiers outperform other classifiers. Among them, the kNN classifier adopts an

instance-based approach whose conceptual simplicity makes its adaptation to an incremental classifier straightforward. Therefore, we choose the kNN classifier as one of the classifiers for our experiments.

The studies about NB classifiers are controversial. Some studies [Joachims1998] [Yang1999] have shown that NB classifiers have lower performance than other classifiers. While [McCallum1998] did a study on NB classifiers and they found that the NB classifier with the multinomial model performed comparably to other classifiers. Moreover, NB classifiers are computationally economical. Consequently, we use the NB classifier with the multinomial model as the other classifier for our experiments.

2.2. The Vector Space Model and Term Weighting

The vector space model is one of the most commonly used models for information retrieval [Manning1999]. It has been also intensively utilized in document classification. The popularity of the model is mainly due to its conceptual simplicity. The model represents documents in a Euclidan n-dimensional space. Each dimension of the space corresponds to a term[1] in the vocabulary. The value of each dimension is the weight of its term which is calculated by a chosen weighting scheme. All documents are vectors in the space, so the similarity of two documents can be computed by looking at the angle, or distance of the two vectors, which is called similarity computation.

The main term weighting schemes are term frequency, document frequency, collection frequency. Another weighting scheme, tfidf, combines term frequency and document frequency.

[1] We use 'term' instead of 'word' as the information unit of document classification. There are two reasons: (1) a term could be a phrase which consists of more than one word; (2) a term could be numbers or combinations of symbols other than words.

2.2.1. Term frequency

In the vector space model, the simplest way to assign a weight to a term is to use the count of a term in a document. The weighting scheme is based on the assumption that a frequent term is more likely to be a good description of the content of the document, although the assumption is not applicable if the term is a stopword. In order to prevent a few most frequent terms from dominating a document, term frequency is often dampened by functions such as $f(tf)=\sqrt{tf}$ or $f(tf)=1+\log(tf)$. The dampened term frequency more precisely reflects the term importance. For example, if a term occurs five times in a document, it does mean that the term is more important than a term that occurs only one time but it does not mean that the term is five times more important as the other term [Manning1999].

Term frequency can also be normalized with the length of document. Normalized weights are comparable across documents of different length.

2.2.2. Collection frequency

The collection frequency of a term is the total number of occurrences in the document collection. Given a term t_i and a document collection D, the collection frequency of t_i is

$$cf_i = \sum_{d_j \in D} tf_{i,j}, \qquad (2.1)$$

where $tf_{i,j}$ denotes the term frequency of term t_i in document d_j.

2.2.3. Document frequency

The document frequency of a term is the number of documents in the collection where the term occurs. Formally, the document frequency of t_i in D is

$$df_t = \sum_{d_j \in D} f(d_j, t_t), \tag{2.2}$$

$$\text{where } f(d_j, t_t) = \begin{cases} 0 & t_t \text{ does not occur in } d_j \\ 1 & t_t \text{ does occur in } d_j \end{cases}.$$

2.2.4. tfidf

The tfidf (term frequency-inverse document frequency) weighting scheme combines term frequency and document frequency. Given a term t_i and a document d_j, the term frequency is $tf_{i,j}$, the document frequency is df_i, and tfidf is

$$weight(i, j) = (1 + \log(tf_{i,j}))\log\frac{N}{df_i}, \tag{2.3}$$

where N is the total number of documents and $weight(i, j)$ equals to 0 if $tf_{i,j} = 0$.

2.3. Feature Selection Approaches

A median-size document collection could have tens of thousands of different terms. This high dimensionality is computationally prohibitive for many machine learning algorithms. For example, few neural networks can handle such a large number of input nodes. Bayes algorithms, on the other hand, become computationally intractable unless an independence assumption among features is imposed [Yang1997].

Feature selection approaches remove non-informative terms according to document collection statistics, or they construct new features which combine lower level features into higher level orthogonal dimensions. Many feature selection approaches have been proposed, such as document frequency [Yang1997], information gain [Mitchell1996],

mutual information [Church89], χ^2 [Schutze1995], and term strength [Wilbur1992], etc.

Some commonly used techniques such as removing stopwords and stemming can also reduce dimensionality.

2.4. The Architectures of Classifiers

Before presenting detailed descriptions of the NB and kNN classifiers, we explain the two types of classifier architecture and we choose the one which performs best in our study. These are the *global-classifier* architecture and the *category-classifier* architecture.

The *global-classifier* architecture has only one classifier to perform classification on all categories. Given a query document d, the classifier computes the relevancy scores of all the categories in set C. To perform singular-category classification, the classifier chooses the category which has the highest relevancy score. To perform multi-category classification, the classifier selects a category if and only if the category's relevancy score to d is greater than a threshold. The threshold is the minimum relevancy score for a category to be relevant to a document. A category may or may not have the same threshold with other categories. Regardless of these implementation options, the classifier learns suitable thresholds using the training documents.

Validation is the commonly used approach for choosing an optimal threshold. The idea is to divide the training document set into two sets: set A for training and set B for validating. Set A is used to train the classifier, then the classifier performs classification tests on set B with different threshold values. The value which performs best is chosen as the threshold.

The *category-classifier* architecture has a local classifier for each category instead of a global classifier for all categories. We call a classifier of a particular category a category classifier. Given a query document d, and a category c, the category classifier of c makes a choice over two possibilities: 'd is relevant to c' and 'd is not relevant to c', which can be defined as two target categories for the category classifier. We use $c_{relevant}$ and $c_{nonrelevant}$ to denote them. The category classifier computes the relevancy scores of d to $c_{relevant}$ and $c_{nonrelevant}$. If $c_{relevant}$ has a higher score then it means d is relevant to category c; if $c_{nonrelevant}$ has a higher score then it means d is not relevant to c. The classification system takes all relevant categories as the selected categories for the query document.

Compared to the *global-classifier* architecture, the *category-classifier* one can convert a multi-category problem into a singular-category problem, therefore it doesn't need extra training documents to perform validation for thresholds. Given our goal of achieving high performance with as few training documents as possible, we choose the *category-classifier* architecture for our classifiers, avoiding the need to use thresholds and making efficient use of training documents.

2.5. The Naive Bayes Classifier

2.5.1. Theory

NB classifiers make the "Naïve Bayes assumption". That is, occurrences of terms in a document are independent of each other and independent of their positions. This assumption is false in reality, since text has to conform to syntactic and semantic constraints. However, Naïve Bayes classifiers have shown performance comparable to other top-performing classifiers. [McCallum1998] This paradox is explained by [Friedman1997] and [Domingos1997] who argue classification estimation is only a function of the sign of the function estimation; the function approximation can still be poor while classification accuracy remains high.

Given a category c and a document d, the NB classifier uses the posterior probability $P(c|d)$ as the score of c upon d. Under the *category-classifier* architecture, the NB classifier has a classifier for each category. A category classifier needs to compute the scores of two categories: "d relevant to c" and "d nonrelevant to c". The decision making involves choosing the maximum value between $P(c_{relevant}|d)$ and $P(c_{nonrelevant}|d)$.

Since we do not know the value of $P(c_{relevant}|d)$ or $P(c_{nonrelevant}|d)$, we need to use Bayes' rule to compute them as follows:

$$P(c \mid d) = \frac{P(d \mid c)}{P(d)} P(c), \tag{2.4}$$

where c is either $c_{relevant}$ or $c_{nonrelevant}$. Both $P(d|c)$ and $P(c)$ can be computed from the training document set by maximum-likelihood estimation. Particularly, $P(c)$ is computed by the ratio of the number of training documents labeled with category c to the total number of training documents.

$$P(c) = \frac{N_c}{N} \tag{2.5}$$

In order to estimate $P(d|c)$, we need to use a probabilistic model for text generation. In recent studies of Naïve Bayes classifiers, two different probabilistic models have been used: the multivariate Bernoulli model and the multinomial model. In the multivariate Bernoulli model, a document is represented by a binary-attribute vector. Each attribute indicates whether the corresponding term occurs in the document. Under the Naive Bayes assumption, the following formula is the probability of document d given its category c.

$$P(d \mid c) = \prod_{j=1}^{|V|} [B_j P(t_j \mid c) + (1 - B_j)(1 - P(t_j \mid c))] \tag{2.6}$$

where B_j is dimension j of the vector for document d. B_j equals to either 0 or 1. Using the Laplacean prior, we can estimate the probability of term t_j in category c,

$$P(t_j \mid c) = \frac{1 + \sum_{i=1}^{|D|} B_j P(c \mid d_i)}{2 + \sum_{i=1}^{|D|} P(c \mid d_i)}, \tag{2.7}$$

where $P(c|d_i) \in \{0,1\}$ is given by the document's category label.

In the multinomial model, a document is represented by a vector of term counters. Under the Naïve Bayes assumption, a document d can be seen as $|d|$ independent multinomial trials, where $|d|$ is the length of d. Then, the probability of document d given its category c is the multinomial distribution:

$$P(d \mid c) = |d|! \prod_{j=1}^{|V|} \frac{P(t_j \mid c)^{n(d,t_j)}}{n(d,t_j)!}, \tag{2.8}$$

where $n(d, t_j)$ is the number of occurrences of t_j in d and $|V|$ is the size of the vocabulary. $P(t_j|c)$ is probability of term t_j in category c which can be computed from the training document set. Similar to the multivariate model, we use the Laplacean prior and estimate $P(t_j|c)$ as follow:

$$P(t_j \mid c) = \frac{1 + \sum_{i=1}^{|D|} n(d,t_j) P(c \mid d_i)}{|V| + \sum_{k=1}^{|V|} \sum_{i=1}^{|D|} n(d_i,t_k) P(c \mid d_i)}, \tag{2.9}$$

where $P(c|d_i) \in \{0,1\}$ is given by the document's category label.

[McCallum 98] compared the two models and concluded that the multinomial model outperforms the multivariate model in general. For this reason, we choose the multinomial model in our experiments.

2.5.2. The NB classification algorithm

The score of a query document d_{query} with respect to a category c is the posterior probability $P(c \mid d_{query})$. In order to compute it, we take logarithm:

$$\log P(c \mid d) = \log P(d \mid c) + \log P(c) - \log P(d). \qquad (2.10)$$

$P(d)$ has the same value with respect to either $c_{relevant}$ or $c_{nonrelevant}$ so we don't have to compute $\log P(d)$.

Furthermore,

$$\log P(d \mid c) = \log\left[|d|! \prod_{j=1}^{|V|} \frac{P(t_j \mid c)^{n(d,t_j)}}{n(d,t_j)!} \right]$$

$$= \log |d|! + \sum_{j=1}^{|V|} n(d,t_j) \log P(t_j \mid c) - \sum_{j=1}^{|V|} \log n(d,t_j)!, \qquad (2.11)$$

where both $\log |d|!$ and $\sum_{j=1}^{|V|} \log n(d,t_j)!$ are independent of c, so we have to compute only $\sum_{j=1}^{|V|} n(d,t_j) \log P(t_j \mid c)$.

In Figure 2.1, $D_{training}$ denotes the set of training documents.

/* Batch training with document set $D_{training}$ */

$$P(c_{relevant}) = \frac{\text{the number of documents relevant to } c \text{ in } D_{training}}{\text{the total number of documents in } D_{training}}$$

$$P(c_{irrelevant}) = \frac{\text{the number of documents nonrelevant to c in } D_{training}}{\text{the total number of documents in } D_{training}}$$

for every word t_j in the vocabulary V **do**

$$P(t_i \mid c_{relevant}) = \frac{1 + \sum_{d \in D} n(d,t_i) P(c_{relevant} \mid d)}{|V| + \sum_{t \in V} \sum_{d \in D} n(d,t) P(c_{relevant} \mid d)}$$

$$P(t_i \mid c_{nonrelevant}) = \frac{1 + \sum_{d \in D} n(d,t_i) P(c_{nonrelevant} \mid d)}{|V| + \sum_{t \in V} \sum_{d \in D} n(d,t) P(c_{nonrelevant} \mid d)}$$

end for

/* Classifying query document d_{query} */
Let score($c_{relevant}$) and score($c_{nonrelevant}$) be 0
for every word t_j in the query document d_{query} **do**
    score($c_{relevant}$) = score($c_{relevant}$) + $n(d, t_j)$·log$P(t_j \mid c_{relevant})$
    score($c_{nonrelevant}$) = score($c_{nonrelevant}$) + $n(d, t_j)$·logP($t_j \mid c_{nonrelevant}$)
end for
score($c_{relevant}$) = score($c_{relevant}$) + log$P(c_{relevant})$
score($c_{nonrelevant}$) = score($c_{nonrelevant}$) + log$P(c_{nonrelevant})$
Choose $c_{relevant}$ or $c_{nonrelevant}$ whichever has the higher score

Figure 2.1 The NB classification algorithm

2.6. The k Nearest Neighbors classifier

2.6.1. Theory

The k Nearest Neighbers classifier (kNN) is an instance-based learning algorithm which has been studied in pattern recognition for a long time [Dasarathy91]. In the document classification literature, it is also known as memory-based reasoning (MBR) [Masand92]. The kNN classifier is generally considered as one of the top-performers in document classification. [Iwayama95, Yang99]

The kNN classifier represents documents in document vector space. The base case of the kNN classifier is the single nearest-neighbor classifier whose rationale is very simple: in order to classify a query document, the classifier searches the training set for its most similar document and assigns the query document to the categories of the nearest neighbor. Similarly, the kNN classifier consults the k most similar documents in the training set instead of only one.

2.6.2. Similarity Computation

In order to calculate the similarity of two vectors, the classifier can use the values of cosine, overlap, and distance.

Cosine and Distance

Both the values of cosine and distance can be used for similarity computation if the query vectors are in an n-dimension space, where each term is an axis and the term's weight is the value of the axis. The cosine value is used to calculate the angle of the two vectors, whereas the distance value is used to calculate the distance between two vectors. Given vector $\alpha = <a_1, a_2, \ldots, a_n>$ and vector $\beta = <b_1, b_2, \ldots, b_n>$, the cosine value of α and β is

$$\cos(\alpha,\beta)=\frac{|<\alpha,\beta>|}{|\alpha||\beta|}, \qquad (2.12)$$

where $|<\alpha,\beta>| = \sum_{i=1}^{n} a_i \cdot b_i$, $|\alpha| = \sum_{i=1}^{n} a_i$ and $|\beta| = \sum_{i=1}^{n} b_i$.

The distance of α and β is

$$d(\alpha,\beta)=\sqrt{\sum_{i=1}^{n}(\alpha(a_i)-\beta(b_i))^2} \qquad (2.13)$$

Overlap

The overlap value does not require an n-dimensional space. Given vectors α and β, $\alpha = <a_1, a_2, \ldots, a_n>$ and $\beta = <b_1, b_2, \ldots, b_m>$, where a_i and b_j are elements represented as pairs $<t_i, w_i>$ and $<t_j', w_j'>$, and t's are terms, and w's are weights. The overlap value

calculates the element overlaps of a vector first, and the overlap of the two vectors is the sum of the element overlaps. Formally, the overlap value of element a_i is

$$\text{overlap}(a_i) = \begin{cases} \min(w_i, w_i') & \text{if there is an element } b_j \text{ in } \beta \text{ so that } t_i = t_j \\ 0 & \text{otherwise} \end{cases} \quad (2.14)$$

A good measure of document similarity is critical to the performance of kNN classifiers. Our experiments were designed in part so that we can compare alternative similarity measures.

2.6.3. The kNN classification algorithm

Under the *category-classifier* architecture, the kNN classifier has a category classifier for each category. The category classifier computes scores for two possibilities of relevant, $c_{relevant}$ and nonrelevant, $c_{nonrelevant}$. To compute the score of $c_{relevant}$, the k-nearest-neighbors documents are first found, their similarity to the query document d is then computed. The sum of the similarities is the score of $c_{relevant}$ to d. Similar computation takes place for the score of $c_{nonrelevant}$.

The following pseudocode explains how the kNN classifier is trained with a set of documents $D_{training}$, and then how it classifies a query document d_{query}.

```
/* Batch training with document set D_training */
for every document d in D_training do
    Transform d into a document vector v_d
    Add v_d into training document vector set DV_training
end for

/* Classifying query document d_query */
Transfrom d_query into a vector v_query
Search for the k nearest neighbors in DV_training
```

Let score($c_{relevant}$) and score($c_{irrelevant}$) be 0
for every document vector v in the k nearest neighbors **do**
    if (v is labeled with c)
        score($c_{relevant}$) = score($c_{relevant}$) + sim(v, v_{query})
    else
        score($c_{nonrelevant}$) = score($c_{nonrelevant}$) + sim(v, v_{query})
    end if
end for
Choose $c_{relevant}$ or $c_{nonrelevant}$ whichever has the higher score.

Figure 2.2 The kNN classification algorithm

Chapter 3: Incremental Classifiers

This chapter studies how to make classifiers incremental. Before presenting algorithms, we explain the training and feature selection for incremental classifiers.

3.1. Incremental Training

Incremental classifiers can start classifying documents without training. When the classifier classifies a document, users can give positive feedback if they agree with the classifier, or negative feedback if they think the classifier is wrong. The classifier adjusts itself with the feedback. Since this training is repeatedly applied to one document after another during the classifier's lifetime, we call it incremental training.

It is not difficult to anticipate that incremental classifiers have a start performance problem. At the beginning, incremental classifiers do not have enough classification knowledge, or even have no knowledge at all if there is no initial training, so users have to give feedback frequently. To overcome this problem, a short initial training is usually given to incremental classifiers so that performance can be brought up to a ready-to-go level.

3.2. Feature Selection for Incremental Classifiers

Most feature selection approaches compute the dependence between terms and categories by scanning over a large number of training documents. We use notation, $s(t,c)$ for the dependence between term t and category c. The global goodness of a term is calculated by taking the average or the maximum over all categories:

$S(t) = \sum_{i=1}^{|C|} P(c_i)s(t,c_i)$, where $P(c_i)$ is the probability of category c_i,

$$S(t) = \max_{i=1}^{|C|} \{s(t,c_i)\}. \quad (3\text{-}1)$$

Given the χ^2 approach as an example, it applies a χ^2-measure of dependence to a contingency table containing the number of relevant and non-relevant documents in which the term occurs (N_{r+} and N_{n+}, respectively), and the number of relevant and non-relevant documents in which the term does not occur (N_{r-} and N_{n-}, respectively). Formally,

$$s(t,c) = \frac{N(N_{r+}N_{n-} - N_{r-}N_{n+})^2}{(N_{r+} + N_{r-})(N_{n+} + N_{n-})(N_{r+} + N_{n+})(N_{r-} + N_{n-})}, \quad (3\text{-}2)$$

where N is the total number of documents. The contingency table contains $|V| \times |C|$ items, and V is the vocabulary and C is the category set. As indicated before, the vocabulary of a median-sized document collection could have tens of thousands of terms, and the category set could have hundreds of categories, so it is both time- and space-consuming to maintain such a contingency table. To make the situation even worse, incremental classifiers need to update the contingency table upon accepting user feedback, in contrast to batch classifiers, which construct the table only at the initial training.

Moreover, changing the feature set will change the presentation of document vectors. This makes it difficult to accumulate information incrementally. For example, the kNN classifier has to modify all the vectors in memory, if the feature set is changed.

In order to solve these problems, we have made two adaptations. Firstly, we use the term frequency approach for feature selection. Term frequency evaluates the goodness of a term by the number of its occurrences. The calculation can be done within a document. Secondly, we develop algorithms whose computation uses local vector spaces instead of a global vector space. Particularly, when kNN classifiers compute

similarity, a pair of documents are not mapped into a global vector space. Instead a local vector space is created by merging terms in these documents.

3.3. The Incremental Naive Bayes Classifier

The batch NB classifier computes $P(t_j|\, c)$ and $P(\, c)$ according to the statistics of training documents. When classifying a query document with respect to a category, the classifier computes the posterior probabilities of relevant-to-the-category and nonrelevant-to-the-category. Decision is made by choosing the one with the higher score. The incremental NB classifier continuously takes feedback from the user as incremental training, so $P(t_j|\, c)$ and $P(c)$ change after each incremental training. The classifier cannot take the current values of $P(t_j|\, c)$ and $P(c)$ as the final values, instead it has to keep the statistical data and update the data with each incremental training. When classifying a query document, $P(t_j|c)$ and $P(c)$ are computed on the fly so that they can be used to compute the posterior probabilities.

The following algorithm for the incremental NB classifier shows two major adaptations from the batch NB classifier:

(1) the number of relevant documents and the number of nonrelevant documents are updated after each incremental training;

(2) for computing $P(t_j|c)$ as statistical data, two counters are maintained while the classifier is trained by documents. These counters are the number of occurrence of t_j in all the documents relevant to the category and the number of occurrence of t_j in the documents nonrelevant to the category. More formally. We use notations as follows:

$$\text{count}(c_{relevant}, t_j) = \sum_{d \in D_{training}} n(d, t_j) P(c \mid d), \text{ and}$$

$$\text{count}(c_{nonrelevant}, t_j) = \sum_{d \in D_{training}} n(d, t_j)(1 - P(c \mid d)), \qquad (3\text{-}3)$$

$$\text{where } P(c|d) = \begin{cases} 1 & \text{if } d \text{ is relevant to } c \\ 0 & \text{otherwise} \end{cases}$$

/* Incremental training with d_{inc} */
if d_{inc} is labeled with category c
    $|D_c| = |D_c| + 1$
end if
$|D| = |D| + 1$
if d_{inc} is labeled with category c
    $\text{count}(c_{relevant}, t_i) = \text{count}(c_{relevant}, t_i) + n(d_{inc}, t_i)$
else
    $\text{count}(c_{nonrelevant}, t_i) = \text{count}(c_{nonrelevant}, t_i) + n(d_{inc}, t_i)$
end if

/* Classifying query document d_{query} */

$$P(c_{relevant}) = \frac{|D_c|}{|D|}$$

$$P(c_{nonrelevant}) = \frac{|D| - |D_c|}{|D|}$$

for every word t_j in the vocabulary V **do**

$$P(t_i \mid c_{relevant}) = \frac{1 + count(c_{relevant}, t_j)}{|V| + \sum_{t \in V} count(c_{relevant}, t_i)}$$

$$P(t_i \mid c_{nonrelevant}) = \frac{1 + count(c_{nonrelevant}, t_i)}{|V| + \sum_{t \in V} count(c_{nonrelevant}, t_i)}$$

end for
Let $\text{score}(c_{relevant})$ and $\text{score}(c_{nonrelevant})$ be 0
for every word t_j in the query document d_{query} **do**
    $\text{score}(c_{relevant}) = \text{score}(c_{relevant}) + n(d, t_j) \cdot \log P(t_j \mid c_{relevant})$
    $\text{score}(c_{nonrelevant}) = \text{score}(c_{nonrelevant}) + n(d, t_j) \cdot \log P(t_j \mid c_{nonrelevant})$
end for
$\text{score}(c_{relevant}) = \text{score}(c_{relevant}) + \log P(c_{relevant})$
$\text{score}(c_{nonrelevant}) = \text{score}(c_{nonrelevant}) + \log P(c_{nonrelevant})$
Choose $c_{relevant}$ or $c_{nonrelevant}$ whichever has the higher score.

Figure 3.1 The Algorithm of the Incremental NB Classifier

3.4. The Incremental kNN Classifier

The kNN classifier postpones most computation until it classifies a query document. Its training is basically converting documents into vectors. Therefore, the kNN classifier can be easily changed into an incremental classifier with only minor adaptations:

(1) the classifier should be able to add documents into its training set after the initial training;

(2) an upper limit is imposed on the training set size, otherwise the training set will keep growing, slowing down the whole system.

Moreover, the incremental kNN classifier needs to adjust the value of k while the number of training documents increases. The value of k should be small at the beginning, because there are not many documents in the training set and a large value of k will introduce noise, thereby degrading performance. As the classifier incrementally adds to its training set, the value of k can be increased to take advantage of the new training data.

The following algorithm incorporates these extensions to the batch kNN classifier.

```
/* Incremental training with document d */
Transform d into a document vector v_d
Add v_d into training set DV_training
Adjust k according to | DV_training |

/* Classifying query document d_query */
Transfrom d_query into a vector v_query
Search for the k nearest neighbors in DV_training
Let score(c_relevant) and score(c_irrelevant) be 0
for every document vector v in the k nearest neighbors do
    if (v is labeled with c)
        score(c_relevant) = score(c_relevant) + sim(v, v_query)
    else
        score(c_nonrelevant) = score(c_nonrelevant) + sim(v, v_query)
    end if
end for
```

Choose $c_{relevant}$ or $c_{nonrelevant}$ whichever has the higher score.

Figure 3.2 The Algorithm of the Incremental kNN Classifier

Chapter 4: Experiments with the Incremental Classifiers

Based on the theoretic framework described in the previous chapter, we now describe a series of experiments intended to answer three questions:

1) What is the initial performance of the kNN and Naive Bayes classifiers after trained with a small training set?
2) How fast do the classifiers improve their performance by learning incrementally from feedback?
3) How do term weighting schemes, similarity computation approaches, feature selection approaches affect the performance of the kNN classifier? How does feature selection affect the performance of the NB classifier? Which combination is the best for the kNN classifier and NB classifier, respectively?

Before presenting the experiment results, firstly, we describe the benchmark, Reuters-21578, and how we select experiment documents; secondly, we explain the set of performance measures that we use.

4.1. The Reuters-21578 Benchmark

Reuters-21578 has recently become the standard benchmark for comparisons between different document classifiers. The documents in the Reuters-21578 collection appeared on the Reuters newswire in 1987. The documents were assembled and indexed with categories by personnel from Reuters Ltd. Formatting of the documents and production of associated data files were done in 1990 by David D. Lewis and Stephen Harding at the Information Retrieval Laboratory. Several versions of Reuters have been used in the literature. The two important ones are Reuters-22173, Distribution 1.0, 1993, and Reuters-21578, Distribution

1.0, 1996. Several format improvements over its predecessor make Reuter-21578 the most convenient data set for our experiments.

Reuters-21578 contains 21578 documents each of which is in SGML format. Below is a typical document in the collection.

```
<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET"
OLDID="18421" NEWID="2003">
<DATE> 5-MAR-1987 09:19:43.22</DATE>
<TOPICS><D>grain</D><D>wheat</D></TOPICS>
<PLACES><D>usa</D><D>iraq</D></PLACES>
<PEOPLE></PEOPLE>
<ORGS></ORGS>
<EXCHANGES></EXCHANGES>
<COMPANIES></COMPANIES>
<UNKNOWN>
&#5;&#5;&#5;C G
&#22;&#22;&#1;f0012&#31;reute
u f BC-/CCC-ACCEPTS-BONUS-BI  03-05 0117</UNKNOWN>
<TEXT>&#2;
<TITLE>CCC ACCEPTS BONUS BID ON WHEAT FLOUR TO IRAQ</TITLE>
<DATELINE>  WASHINGTON, March 5 - </DATELINE>
<BODY>The Commodity Credit Corporation, CCC, has accepted bids for export bonuses to cover
sales of 25,000 tonnes of wheat flour to Iraq, the U.S. Agriculture Department said. The department
said the bonuses awarded averaged 116.84 dlrs per tonne. The shipment periods are March 15-
April 20 (12,500 tonnes) and April 1-May 5 (12,500 tonnes). The bonus awards were made to
Peavey Company and will be paid in the form of commodities from CCC stocks, it said. An
additional 175,000 tonnes of wheat flour are still available to Iraq under the Export Enhancement
Program initative announced January 7, 1987, the department said.
 Reuter &#3;</BODY></TEXT>
</REUTERS>
```

Figure 4.1 A Reuters Document

A Reuters document is labeled with predefined categories. As you can see in Figure 4.1, there are six predefined category schemes: TOPICS, PLACES, PEOPLE, ORGS, EXCHANGES, COMPANIES. TOPICS is the most commonly used one. Reuters-21578 has 135 predefined categories of TOPICS. (See Figure 5.2) A document could have zero, one, or more topics.

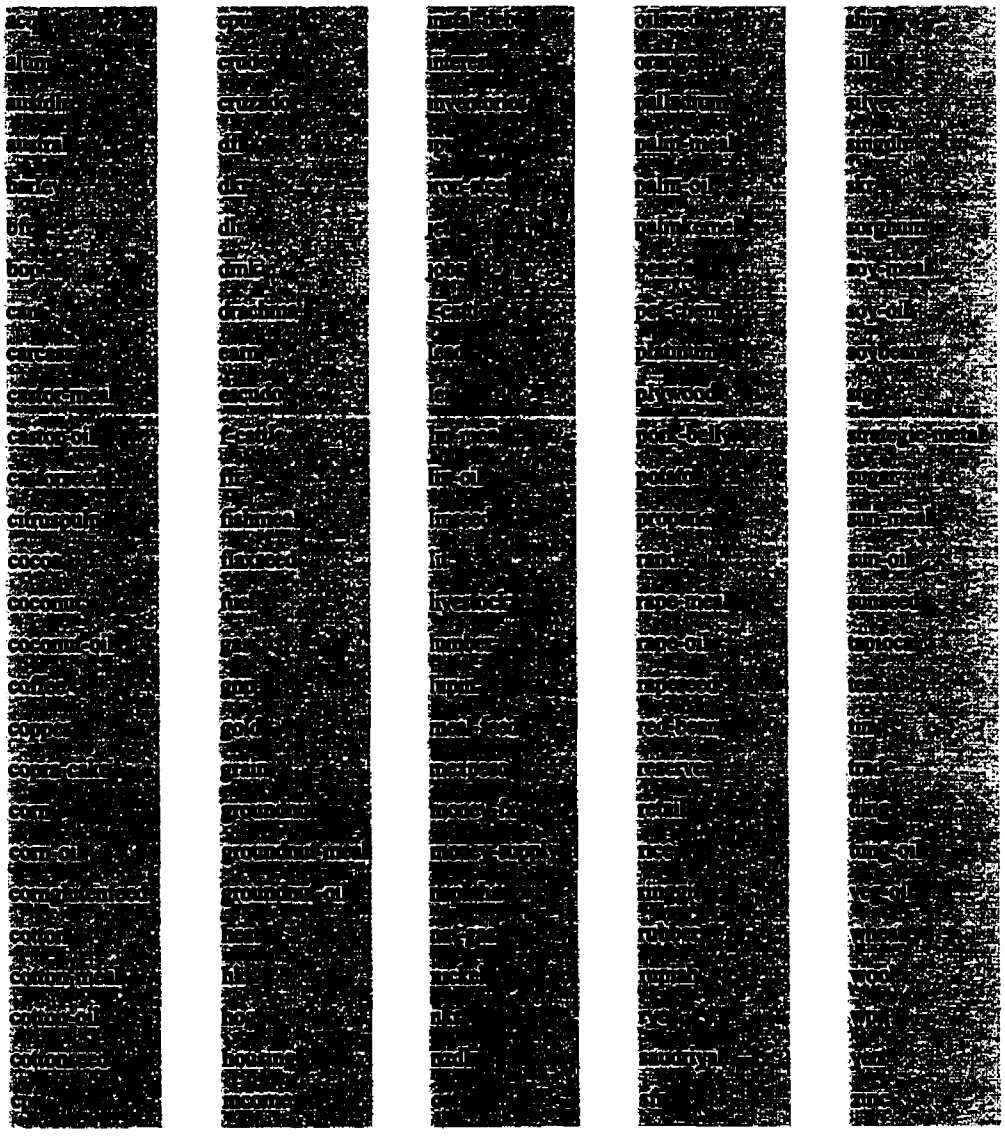


Figure 4.2 Topics in Reuters-21578

The distribution of Reuters documents among topics is extremely uneven. There are quite a few topics which have very few relevant documents or even none, whereas there are a few topics which have thousands of relevant documents.

Incremental classifiers are not supposed to need many training documents to gain ready-to-go performance, however a large number of documents are still needed to test the classifiers and adequately evaluate their performance. Moreover, in order to show learning curves of classifiers, additional training documents are needed after initial training. To carry out our experiments, we chose the ten most common topics.

A document generator has been developed to simulate the working environment of an incremental classifier. For each experiment, the generator creates three document sets:
(1) 50 documents for initial training. The generator randomly selects 5 documents from each category, so the total number of documents is 5×10=50.
(2) 950 documents for incremental training. The generator randomly selects them from the Reuters collection. Document distribution across categories is random.
(3) 2000 documents for testing. The generator randomly selects them from the Reuters collection. Document distribution across categories is random.
The generator ensures that the three sets do not overlap.

The top ten topics include 8599 documents. Their distribution is shown in Table 4.1. Each of them has more than 200 documents which is enough for incremental training and performance testing.

Acq	Corn	Crude	earn	grain	Interest	Money-fx	ship	trade	wheat
2210	223	566	3776	574	424	684	295	514	287

Table 4.1 The Top Ten Reuters Topics

4.2. Performance Measures

A number of performance measures are used in our experiments to provide comprehensive results. Moreover, to make results comparable to other published experiment results, we follow the standard definition of these performance measures.

4.2.1. Precision and recall

In the classification problem, there are a set of targeted relevant documents and the classifier has a set of selected documents that it has labeled as relevant. We use Figure 5.3 to show the situation, where the left circle is the set of documents selected by the system and the right circle is the set of targeted relevant documents.

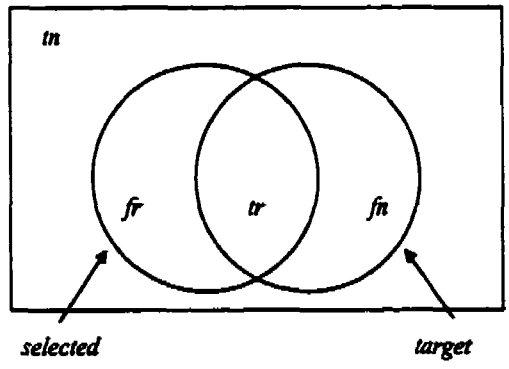


Figure 4.3 An Illustration of Precision and Recall

In the figure,

tr (true relevants) includes cases where the system made right decisions of relevance.

tn (true nonrelevants) includes cases where the system made right decisions of nonrelevance.

fr (false relevants) includes cases where the system made wrong decisions of relevance.
fn (false nonrelevants) includes cases where the system made wrong decisions of nonrelevant.

Precision is defined as a measure of the ratio of selected items that the system selected correctly, over the total number of items selected by the system:

$$precision = \frac{tr}{tr + fr}. \tag{4-1}$$

Recall is defined as the ratio of the target items that the system has selected, over all the target items:

$$recall = \frac{tr}{tr + fn}. \tag{4-2}$$

4.2.2. F1

Generally we have to trade off between precision and recall. For example, if a system can select documents in the collection only when it has very high confidence level, then it gets high precision but low recall. On the other end, a system can get 100 percent recall by selecting all documents as relevant, but precision will be very low.

One way to combine precision and recall into a single measure of overall performance is the *F1* measure. *F1* measure is first introduced by [van Rijsbergen1979] and has been used very commonly as a standard performance measure. *F1* is defined as follows:

$$F1 = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} \tag{4-3}$$

where P is precision, R is recall and α is a factor which determines the weighting of precision and recall. A value of $\alpha = 0.5$ is often chosen for equal weighting of P and R. With this α value, the $F1$ measure simplifies to

$$F1 = \frac{2PR}{P+R} \tag{4-4}$$

4.2.3. Micro-average and Macro-average *F1*

The average of the precision, recall, and $F1$ values can be computed by two approaches. The first approach is macro-average which computes the values for binary decisions on each individual category and then get the average value over all categories. The second approach is micro-average which computes the average value globally over all the $n \times m$ binary decisions where n is the number of total test documents, and m is the number of categories in consideration. [Yang1999]

Intuitively, the micro-average value are more likely to be dominated by the classifier's performance on common categories; the macro-average values tend to be influenced by the classifier's performance on rare categories. In the thesis, both macro-average and micro-average are used for performance evaluation, preventing us from a biased view.

4.2.4. Accuracy

In singular-category classification, accuracy is simply the ratio of correct category assignments to all category assignments, because each category assignment can only choose one category and a document can only have one category. In multi-category classification, a document could have more than one category, and the classifier needs to make as many decisions as the number of categories. The total number of decisions is $n \times m$,

where n is the number of query documents and m is the number of categories. Therefore, accuracy is ratio of correct category decisions to all decisions. For example, there are 10 target categories, c_0, c_1, ..., c_9 and only one query document. If the target category of the query document are c_1, c_2 and c_5 and the classifier has selected c_1, c_2, c_7 and c_8, then accuracy is 70%.

4.3. Experiments with the kNN classifier

4.3.1. The Values of *k*

The parameter k affects the performance of the kNN classifier significantly. If k is too small, the classifier does not attain its top performance because one or two misleading training documents can result in misclassification. On the other hand, if k is too large, the classifier again does not perform precisely because it counts on too many training documents some of which are noise and should be ignored. Figure 4.4 is the micro-average F1 of the kNN classifier with a training set of 750 documents. According to the figure, the classifier has

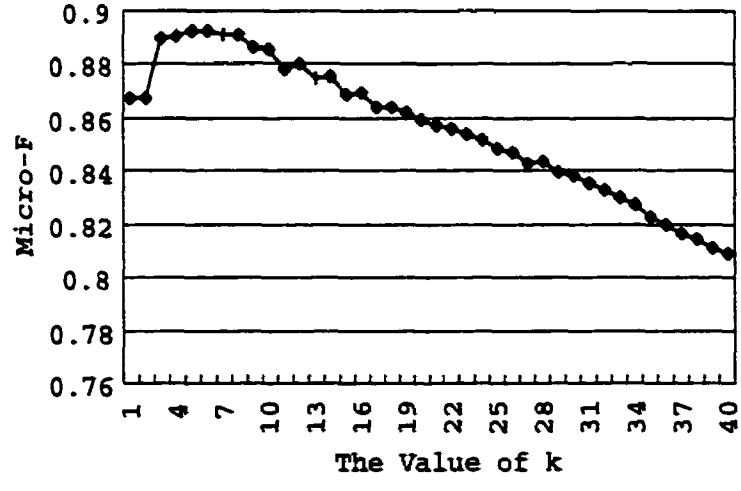


Figure 4.4 The micro-F1 of the kNN classifier with k=70

relatively high and stable performance when *k* ranges from 3 to 10. The values of *k* outside this range lead to interior performance.

While the classifier is trained incrementally, does the top-performance range of *k* values remain unchanged? We did a series of experiments to identify the ranges of k values for different training set sizes. The results are shown in Figure 4.5. The interesting facts are:

1) The start of the top performance range remains approximately the same regardless the training set size.
2) The span of the range grows while the number of training document increases.

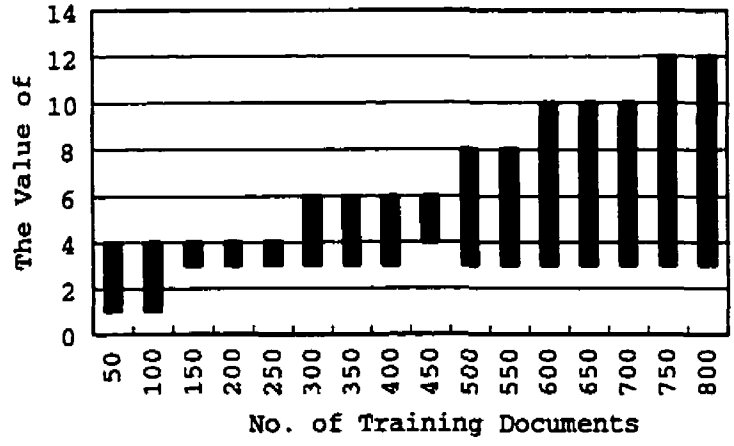
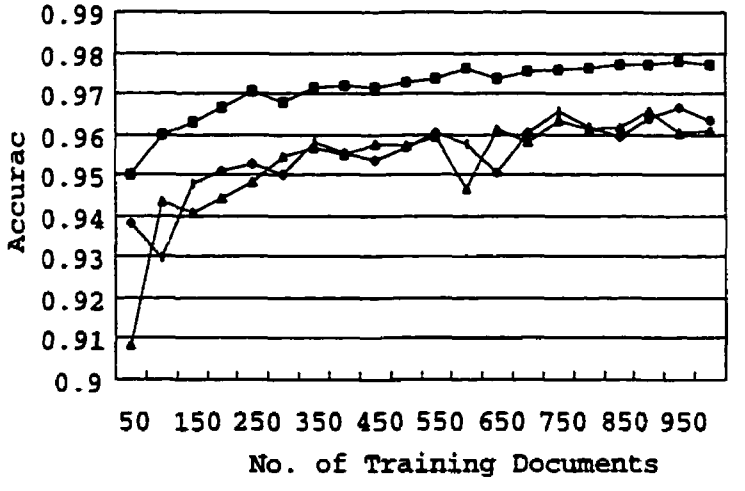
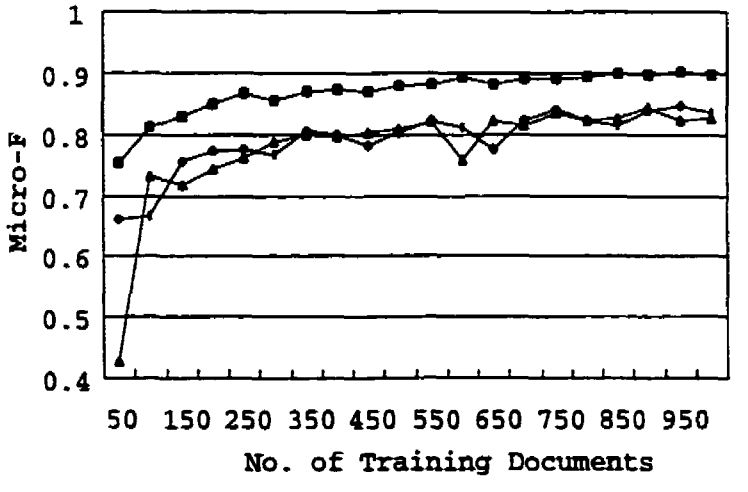


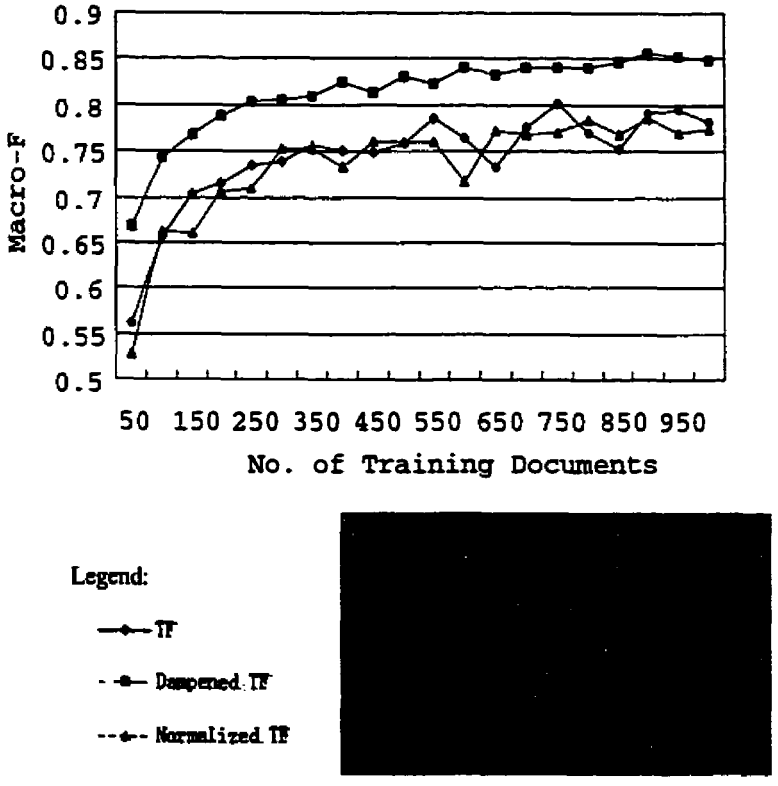
Figure 4.5 The Top-performance Ranges of k Values

4.3.2. Term Weighting

Four term weighting schemes have been tested: term frequency, dampened term frequency, normalized term frequency, and tfidf. In our experiments, tfidf assigns almost every document to nonrelevant with respect to each category. We were not sure what is the cause of this low performance. The performance of tfidf is not included here.





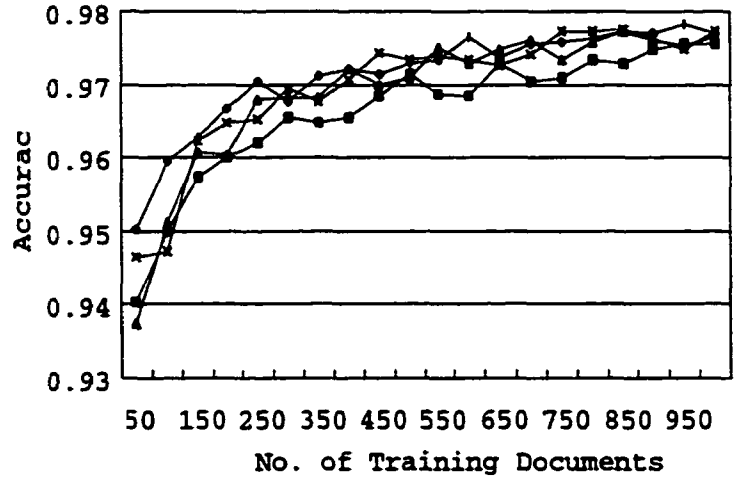


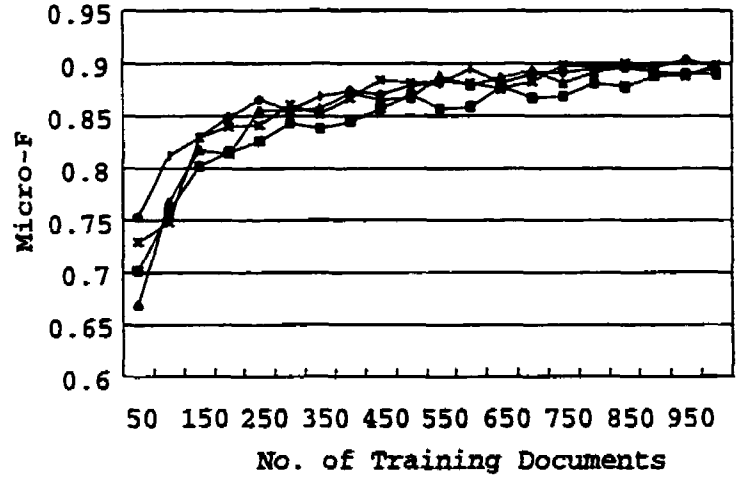
Regarding the dampened term frequency, we use the square root as the dampening function.

The results show the dampened term frequency gives the highest performance.

4.3.3. Feature Selection

The kNN classifier has also been tested without/with feature selection. In the former case, we keep all terms in a document -- except those terms that are in the stopword list or that are removed by stemming. In the latter case, we used feature set sizes of 30, 40 and 50. We scanned through all the Reuters collection and found that the average number of distinct terms in a document is 68 after stopword removing and stemming, so these three sizes are of interest to test.





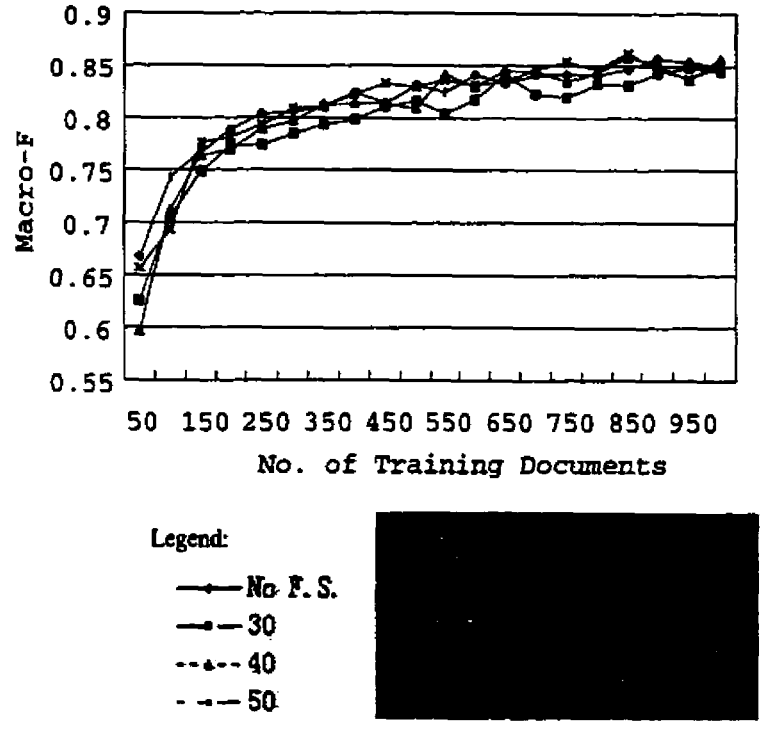
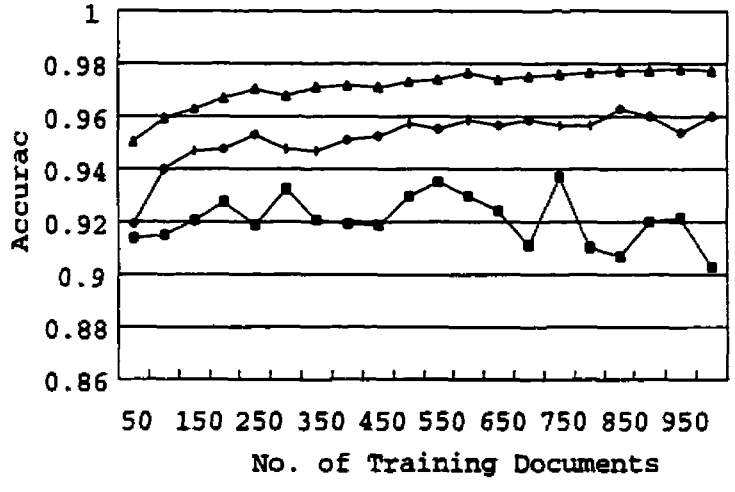


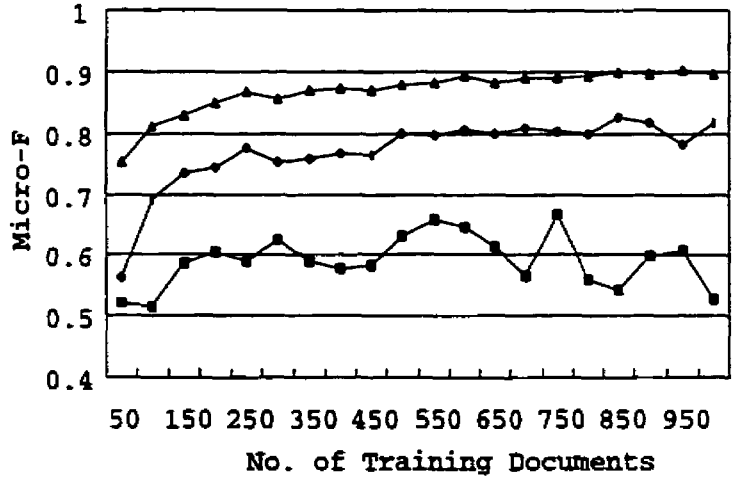
Figure 4.7 The kNN Classifiers without/with Feature Selection

Although performance for no feature selection, and performance for the three feature sizes tend to converge, the classifier without feature selection outperforms the other three in general.

4.3.4. Similarity Computation

This series of experiments are designed to test the kNN classifiers with three different similarity computation approaches: the cosine value, the distance value, and the overlap value.





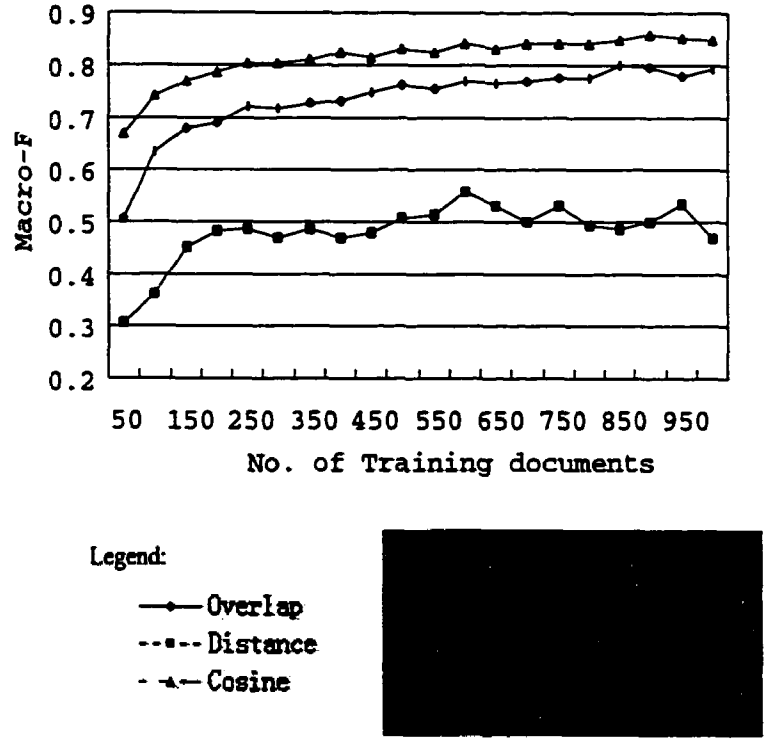


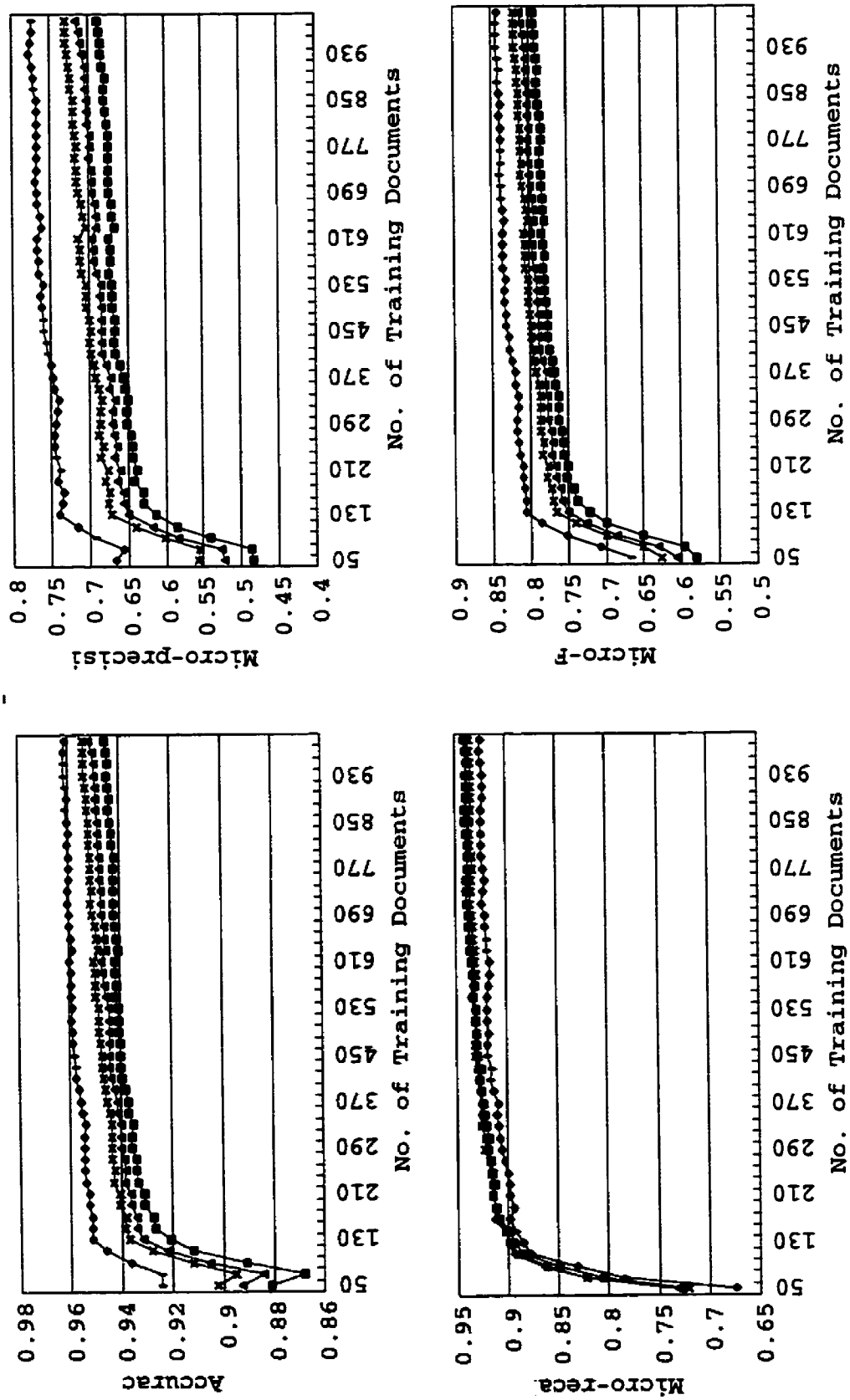
Figure 4.8 The kNN Classifiers with Different Similarity Computations

It is apparent that the cosine value outperforms others and the distance value gives the lowest performance.

4.4. Experiments with the NB Classifier

The NB classifier has been tested with/without feature selection. For the same reason for the kNN classifier, we tested the classifier for feature set sizes of 30, 40 and 50. The results are shown below.

The NB classifier with no feature selection has higher micro and macro precision while the NB classifiers with feature selection have higher micro and macro recall. But, the NB classifier with feature selection has higher overall performance because it has higher F1 and accuracy.


Micro-precisi
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0.45
0.4
50
130
210
290
370
450
530
610
690
770
850
930
No. of Training Documents
Micro-F
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
50
130
210
290
370
450
530
610
690
770
850
930
No. of Training Documents
Accurac
0.98
0.96
0.94
0.92
0.9
0.88
0.86
50
130
210
290
370
450
530
610
690
770
850
930
No. of Training Documents
Micro-reca
0.95
0.9
0.85
0.8
0.75
0.7
0.65
50
130
210
290
370
450
530
610
690
770
850
930
No. of Training Documents

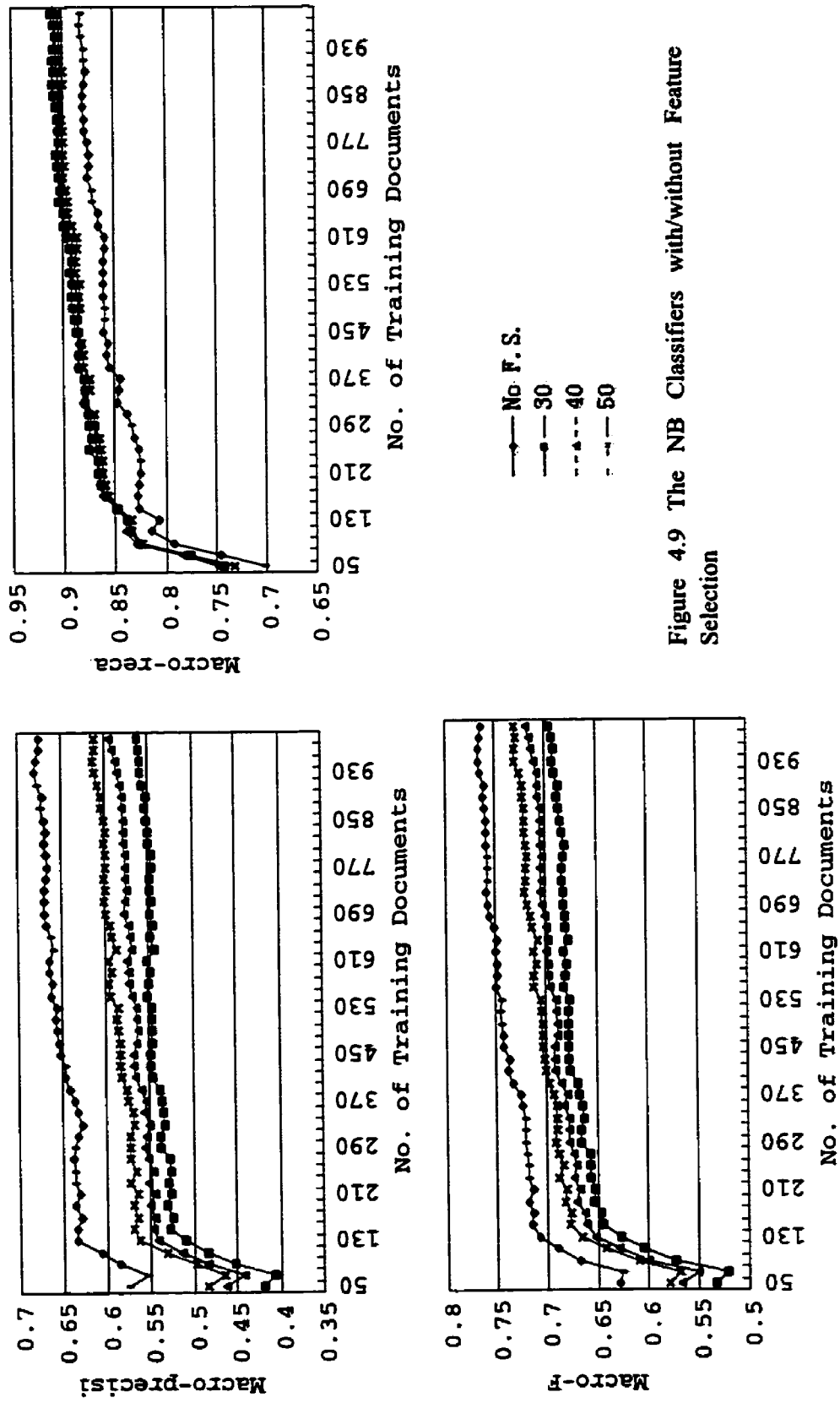


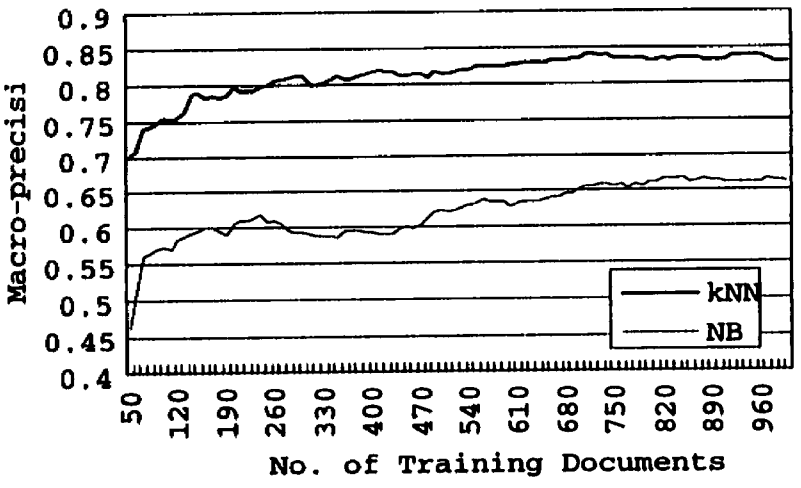
Figure 4.9 The NB Classifiers with/without Feature Selection

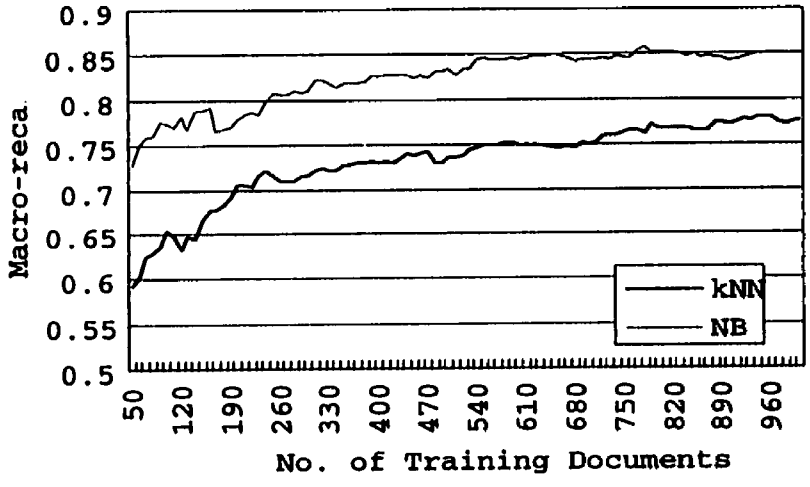
4.5. Performance Comparison of The kNN Classifier vs. the NB Classifier

Given the previous experimental results, we know the best implementation for the kNN or NB classifier. This series of experiments are designed to compare the performance of the kNN and NB classifiers, both with their most efficient implementation. That is the kNN classifier uses the dampened term frequency for weighting scheme, the cosine value for similarity computation, and no feature selection. Moreover, the value of k is in the top performance range. And the NB classifier uses no feature selection.

The experimental results are shown in Figure 4.10. The kNN classifier has higher micro- and macro-precision than the NB classifier, and the kNN classifier has lower micro- and macro-precision than the NB classifier. However, the kNN classifier significantly outperforms the NB classifier in general. This is shown by the overall performance indices: micro-F1, macro-F1, and accuracy.

Acurrac
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
50 120 190 260 330 400 470 540 610 680 750 820 890 960
No. of Training Documents
kNN
NB
Micro-reca
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
50 120 190 260 330 400 470 540 610 680 750 820 890 960
No. of Training Documents
kNN
NB
Micro-F
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
50 120 190 260 330 400 470 540 610 680 750 820 890 960
No. of Training Documents
kNN
NB
Micro-precisi
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
50 120 190 260 330 400 470 540 610 680 750 820 890 960
No. of Training Documents
kNN
NB





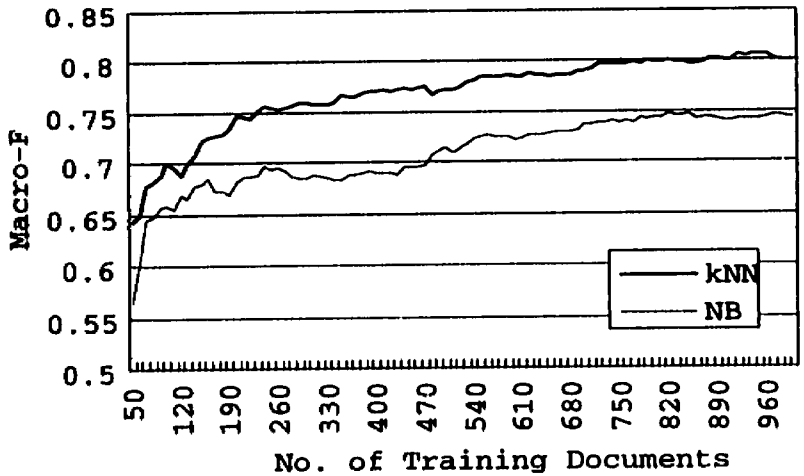


Figure 4.10 Performance Comparison: the kNN Classifier versus the NB Classifier

4.6. Summary of Experiments

The version of the kNN classifier that had the best performance was the one that combined the dampened term frequency for weighting scheme, the cosine value for similarity computation, and no feature selection. Moreover, its value for k was in the top performance range. Similar to the kNN classifier, the naïve Bayes classifier has the top performance when no feature selection is performed.

Both the kNN classifier and the NB classifier meet our two goals for incremental classification:

1) achieve ready-to-go performance after trained with a small training set;

2) incrementally improve performance by interacting with users.

Finally, the kNN classifier attains better overall performance that the NB classifier does. Therefore, we adopt the kNN classifier with aforementioned parameter settings for our EXIP case study.

Chapter 5: The Document Classification Component in the EXIP

The Executive Information Portal (or EXIP) is a prototype toolset that we are currently developing in the Knowledge Management Laboratory of the University of Toronto. It is intended to help a group of strategic business analysts working for a large corporation. Their task is to keep track of current events as they unfold, and make sure that their company's strategic objectives remain on track. To work on this task, business analysts scan news stories (Reuter's, CNN, etc.), analysts' reports (Yankee Group, Forrester Research, and the like) and other document sources, looking for relevant materials. Once they have decided that a particular document is useful, they add it to their own library, write annotations and prepare memos to be circulated to their colleagues. This work is currently done without any tool support, or vanilla computer tools (e.g., a web browser and search engine). The EXIP aims to support the semi-automatic search and classification of documents, also the analysis of collected information with respect to a given set of strategic objectives.

5.1. System Architecture

The EXIP global architecture is shown in Figure 5.1. The outermost layer of the architecture (bottom part of the figure) includes external information sources, such as CNNfn (cnnfn.cnn.com/news/technology/), CBC business news (cbc.ca/business/), the Globe and Mail (globeandmail.com/hubs/rob.html) and Forrester Research, whose reports we assume that the analysts download manually. This layer also includes wrappers for each source, which specify expected outputs for input queries. The information sources may be structured, semi-structured, proprietary document formats or plain text. In the prototype implementation, we wrap semi-structured (HTML) sources and plain text sources. Structured sources are easy to wrap and access, while documents in proprietary

formats (PDF, MS-Word, RTF, etc.) can be translated to a common HTML or XML format using commercial tools such as Document Navigator from Verity [Verity].

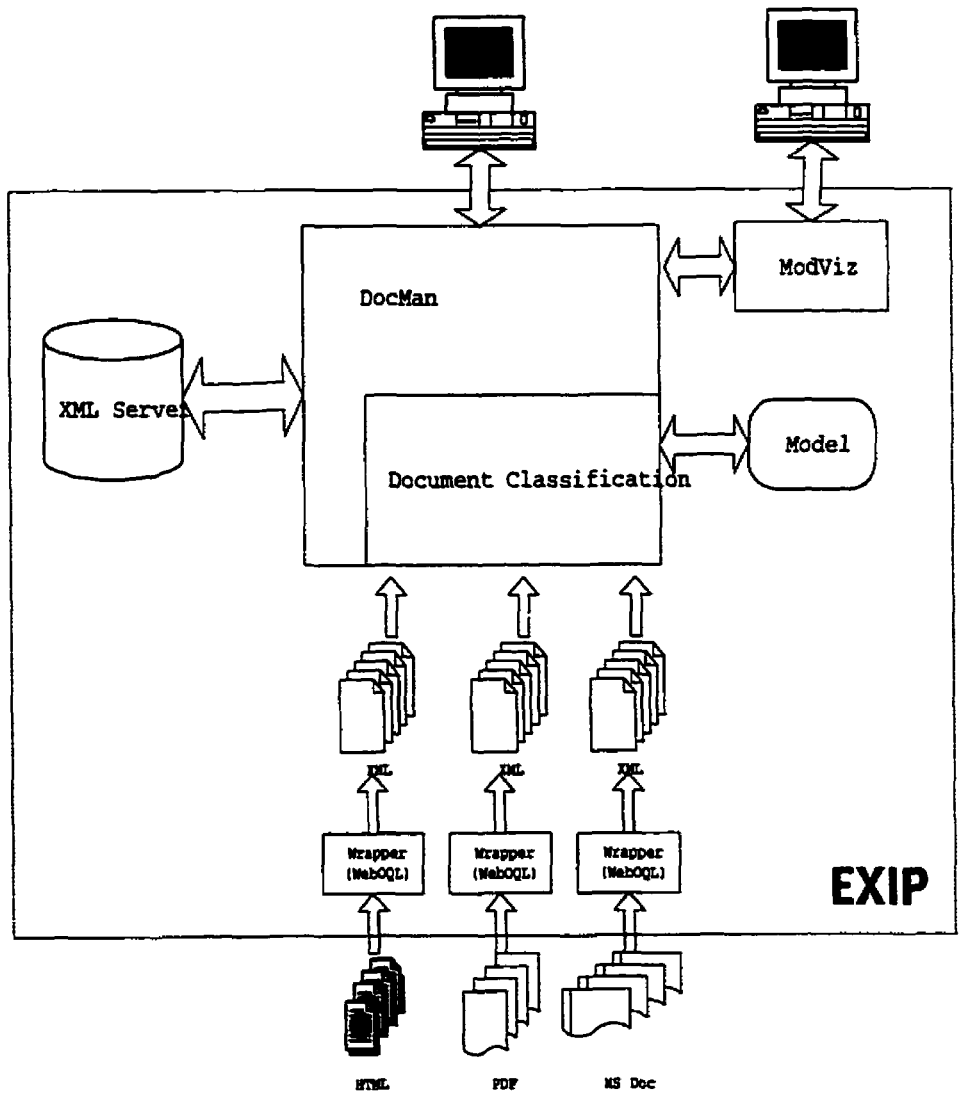


Figure 5.1: The Overall Architecture of the EXIP

Wrappers export data in XML format. One important reason for selecting XML over HTML is that XML is fast becoming the lingua franca for information exchange, and its adoption allows us to benefit from a wealth of research in this area. Moreover, XML tags

carry more useful information than HTML ones. This information can be used to better index and otherwise process retrieved documents. All exported data are translated into a common document schema, essential for the processing of documents after they have been downloaded.

The Document Classification component attempts to classify all downloaded documents by relating them to one or more elements of the semantic model. The proposed classification may be approved or overruled by the users of the system, or it may be accepted "as is" when the system is in "automatic" mode. The Document Management component, DocMan, provides support for document viewing, annotation and manipulation. DocMan uses an XML Data Server for its operations.

ModViz is the Model Visualization component. It offers support for creating, visualizing and maintaining the semantic model. Using ModViz, strategic analysts can view different parts of the semantic model, update it, or retrieve all relevant documents associated with some of its elements.

5.2. Document Management

DocMan, the document management component of the EXIP, is designed to manage thousands of documents that have multiple links among them. Furthermore, we have to support complex search operations on these documents, involving both full-text and metadata search. Moreover, the documents have to be indexed according to a multidimensional index with respect to the semantic model. Last but not least, the system has to support instant updates.

A possible platform for the implementation of DocMan is provided by information retrieval system technologies. Such technologies can handle large collections of documents (up to a range of millions). In addition, These technologies are also equipped with advance search capabilities such as fuzzy or proximity search. However, we decided

that such technologies are too "heavy weight" for our purposes. Moreover, they don't support real-time updates of documents, nor do they use in a direct way hypertext links.

At the other end of the spectrum there are document content management systems, also sometimes known as document management systems. Most of these were developed for technical publishing management, but recently they moved into the enterprise portal domain. With respect to our requirements, such systems do support hyperlinks between documents; also they have facilities for describing document metadata. Furthermore, the latest versions of such systems have moved towards the adoption of XML as their document model, meeting another one of our requirements. Their main drawback, however, is the fine granularity at which they work. We would like to manage documents at the document level, rather than the paragraph or images level. Furthermore, DCMSs do not address issues of efficient storage and retrieval for documents.

Because of the aforementioned requirements, we propose to adopt ToX (the Toronto XML Server) as the backbone of our document management system. ToX is currently being developed at the University of Toronto. The main objectives of the ToX project are to offer database-like services to XML data/documents including alternative storage methods, complex query processing, full-text and path index capabilities, as well as transaction processing. Given that ToX is in the early stage of development, for the first DocMan implementation we employ off-the-shelf software for our XML server. In particular, we use the IBM's DB2 XML-Extender [DXX]. In addition, we plan to test and evaluate other off-the-shelf XML server solutions, such as the Poet content management system [Poet] and the Microsoft SQL-Server 2000 [MS-SQL].

The basic functionality of DocMan includes registering and storing documents, as well as support for document annotation and document summary. In addition, DocMan supports primitives for document-related EXIP functions such as relationships across documents and document search (full text, metadata, and index based search.)

Once a document is registered in DocMan, it can be annotated or summarized. Annotations can be thought as "knowledge records" for our system in the sense that they facilitate the exchange of knowledge between a group of collaborating business analysts. Moreover, annotations can be used as discussion threads.

Another DocMan feature is the support for document relationships. There are many types of relationships between documents. For example, a document may support or contradict or follow-up or simply relate to another document. In the internal representation, a relationship is a relationship tag with the corresponding attribute for the relationship type.

Since annotations and summaries are in XML format, they are easy to search. Thus, we use the ToX search engine to search not only documents but also annotations and summaries. Moreover, using the XML structure we can also follow the relationships among documents.

5.3. The Semantic Model

The semantic model provides a description of the strategic objectives of an organization in terms of goals and subgoals, also the events and actors that can influence any of these goals (positively or negatively). The model can be thought as a network of relationships between goals, actors, events and documents. Thanks to the rich modeling framework, the model supports various forms of analysis. For example, analysts can visualize if and how the organization is advancing towards achieving its goals, what are the obstacles, critical events to look out for, and what are the dependencies to external organizations.

Goals

A goal represents a desirable state of affairs, such as "be the largest internet service provider (ISP) in Canada", or "increase market share by 20%". A goal can be decomposed into subgoals through two basic types of relationships. An AND-relationship relates a goal to a set of subgoals such that fulfilling all subgoals is a sufficient condition

for the fulfillment of the goal. An OR-relationship relates a goal to a set of subgoals such that fulfilling at least one of the subgoals is a sufficient condition for the fulfillment of the goal.

Events and Event Types

An event is an occurrence of an activity. For example, "Telecom A buys media company B for $1B" is an event. An event type is a generic description of a category of event. For example, "competitor buys a content provider" is an example of an event type. Both events and event types are related to goals through positive ("+") or negative ("-") links.

Documents

A document is a unit of information which is retrieved from any source and related to goals and/or events through "supports", "contradicts", "instantiates", "refers" and "describes" relationships. For example, a news document from Globe and Mail "instantiates" the event, "Telecom A buys media company B for $1B".

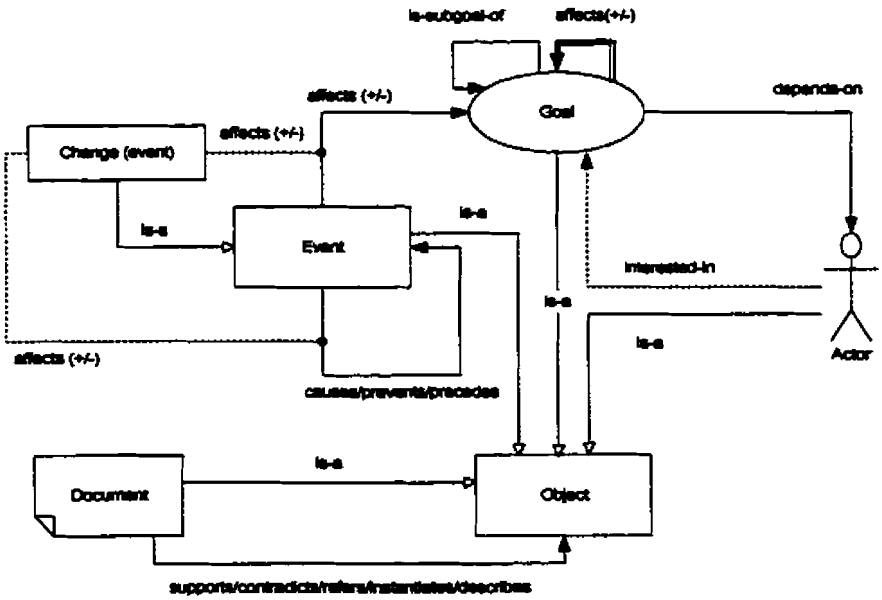


Figure 5.2: The Metamodel of the EXIP Semantic Model

Figure 5.2 shows a summary of the modeling concepts supported by the EXIP semantic model. Figure 5.3 shows an example model for an telecommunication company. For a detailed description of the semantic model, please refer Raoul Jarvis' master thesis. [R. Jarvis2001]

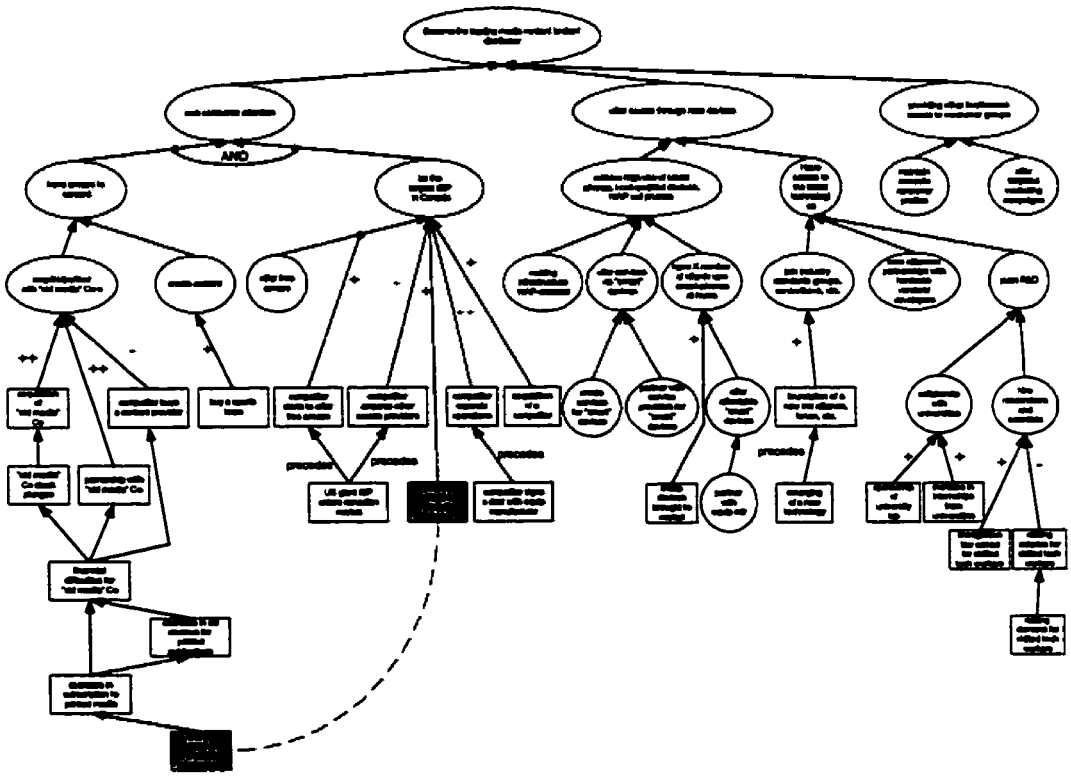


Figure 5.3: An Example Semantic Model

5.4. Classifying Documents into the Semantic Model

As indicated in Figure 5.1, a wrapper retrieves documents from an external information source, and simultaneously transforms them from a domestic format into an XML schema. The transformation is necessary because documents from different sources may have very

different formats, such as Acrobat PDF, HTML, or pure ascii. Even if they are in the same format, say HTML, they could still present information in very different ways.

The schema that has been adopted is given in Figure 5.4. The schema consists of five sections. The first is the DOCUMENT section, and it includes relevant document fields, such as TITLE, AUTHOR, CONTENT and so on. The second is the optional TOPICS section which is for generic topic classification. The topic classification organizes documents under a traditional topic taxonomy. The third is the RELEVANTNODES section which contains all the elements in the semantic model to which the document is relevant. The fourth and fifth are the SUMMARIES and ANNOTATIONS sections, accommodating summaries and annotations made by analysts.

```
<!ELEMENT EXIPDOCUMENT (TOPICS, MODEL, DOCUMENT, SUMMARIES,
                        ANNOTATIONS) >
<!ELEMENT DOCUMENT (DOCID, TITLE, SOURCE, DATE, AUTHOR, CONTENT,
                    RELEVANTDOCS) >
    <!ELEMENT DOCID (#PCDATA) >
    <!ELEMENT TITLE (#PCDATA) >
    <!ELEMENT SOURCE EMPTY >
      <!ATTLIST SOURCE NAME CDATA #REQUIRED
                       URL CDATA #REQUIRED >
    <!ELEMENT DATE (#PCDATA) >
    <!ELEMENT AUTHOR (#PCDATA) >
    <!ELEMENT CONTENT (#PCDATA) >
    <!ELEMENT RELEVANTDOCS (RELEVANTDOC+) >
    <!ELEMENT RELEVANTDOC (LINKID, LOCALPARAGRAPH, TARGET,
                           LINKTYPEID, USERID, DATE, COMMENT>
<!ELEMENT TOPICS (#PCDATA) >
<!ELEMENT RELEVANTNODES (NODE+) >
    <!ELEMENT NODE EMPTY >
        <!ATTLIST NODE ID CDATA #REQUIRED >
<!ELEMENT SUMMARIES (SUMMARY+) >
    <!ELEMENT SUMMARY (SUMMARYID, USERID, DATE, TEXT) >
        ...
<!ELEMENT ANNOTATIONS (ANNOTATION+) >
    <!ELEMENT ANNOTATION (ANNOID, POSITION, CONTRIBUTION+) >
        ...
```

Figure 5.4: The Document XML Schema

The classification component classifies documents into two types of elements in the semantic model: goals and events. To accomplish the classification task, the component first converts these model elements into a flat classification taxonomy. A category

classifier is then built for each model element to judge whether a certain document is relevant. Analysts are asked to provide a few sample documents relevant to each model element. Category classifiers train themselves with these sample documents and gain ready-to-go performance. Nevertheless, if no sample documents are provided for a model element, the corresponding category classifier will still be able to work, but the component will give less confidence to its classification results and users have to review the results more carefully and give more feedback to improve the classifier's performance. There is no need to provide nonrelevant sample documents to a model element because sample relevant documents to other elements are nonrelevant to the model element.

5.5. Using Clustering to Evolve the Semantic Model

The team of business analysts changes quickly. So do their preferred document sources and views of what is or isn't relevant. Accordingly, we expect that an organization's strategic business model will evolve continuously to reflect such changes. The evolution includes extending the semantic model by refining goals into subgoals, or decomposing an event type to several event types. Our system can give directions for such extensions. Specifically, clustering is applied when the number of documents under a goal or event is too large. This leads to smaller, more manageable clusters associated with any one node of the semantic model. Another aspect of evolution involves classifiers, which adjust themselves with feedback from users. Incremental classifiers are proposed in Chapter 4 to achieve this evolvability.

The clustering algorithm that we propose is based on the group average clustering algorithm introduced by [Cutting1992].

Let G be a document group. The group average similarity of G is

$$S(G)=\frac{1}{|G|(|G|-1)}\sum_{\alpha\in G}\sum_{\beta\in G}s(\alpha,\beta)\cdot \qquad (5\text{-}1)$$

Let Φ be a set of disjoint document groups. The algorithm searches for two different groups Γ and Δ which maximize $S(\Gamma \cup \Delta)$ over all choices from Φ. A new partition Φ' is constructed by merging Γ with Δ.

$$\Phi' = (\Phi - \{\Gamma, \Delta'\}) \cup \{\Gamma \cup \Delta\} \qquad (5\text{-}2)$$

Initially, Φ is a set of singleton groups, one for each individual document to be clustered. The iteration terminates when $|\Phi'|$ equals the desired number of clusters. The complexity of the clustering algorithm is $O(n^2)$ where n is the number of documents to be clustered. Since our system is expected to handle a maximum of a few thousand documents, such complexity is acceptable.

Chapter 6: An Example of Document Classification with respect to a Semantic Model

In order to illustrate how the chosen incremental classifier performs in an operating knowledge management environment, this chapter uses a large North American auto manufacturer as an example. Firstly, a semantic model for the auto manufacturer is introduced. Secondly, we manually collect and classify documents from Reuters-21578 in order to perform classification experiments with respect to the model. Finally, we present and analyze the experimental results.

6.1. The Semantic Model

Figure 6.1 is a fragment of an example semantic model for a large North American auto manufacturer, circa 1987. The reason for choosing this date is that our experimental data are collected from Reuters-21578 whose documents are dated in 1987.

The model fragment starts from a top-level goal "increase return on investment" which is decomposed via an AND relationship into "increase sales" and "increase profit per vehicle". These goals are in turn decomposed and this process continues until these goals that are supported, or their failure of such goals is supported, by events. Lateral relationships may occur during the goal decomposition process. Events can also be decomposed to more specific events.

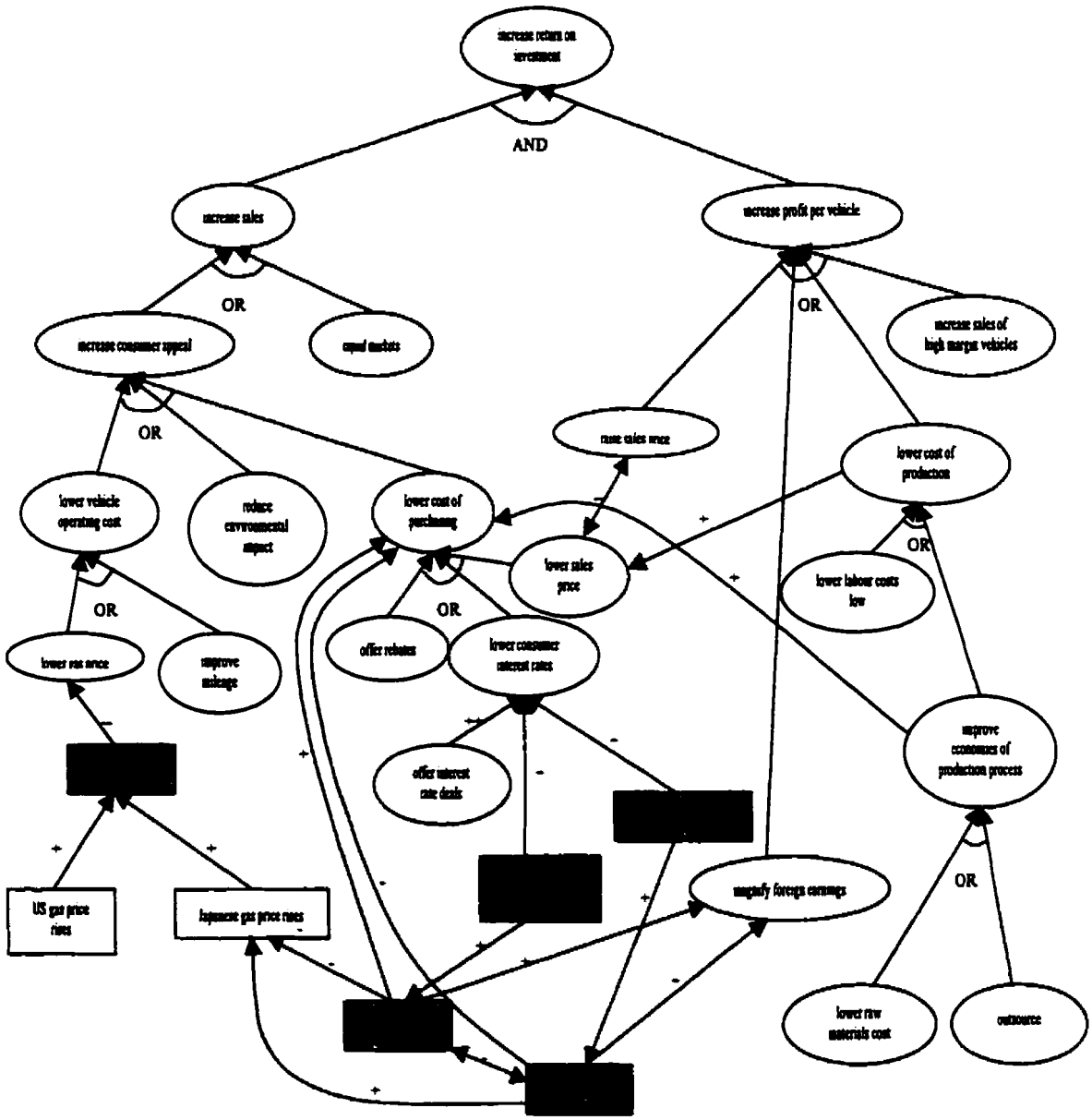


Figure 6.1: A Fragment of a Semantic Model for a North American Auto Manufacturer

6.2. The Preparation of Data

We designed experiments intended to classify incoming documents with respect to five events in the model fragment: gas price rises, US dollar rises, yen rises, US interest rates rise, Japanese interest rates rise (which are gray rectangles in Figure 6.1). Parts of the Reuters-21578 collection are scanned by us to find relevant documents. For example, we searched topics "interest", "cpi", "wpi", looking for documents relevant to the node "US Interest Rates Rise", and we searched topics "yen", "money-fx", "dlr", looking for documents relevant to the node "Yen Rises", and so on. All relevant documents are wrapped according to the XML schema in Figure 5.4. The distribution of relevant documents is as follows:

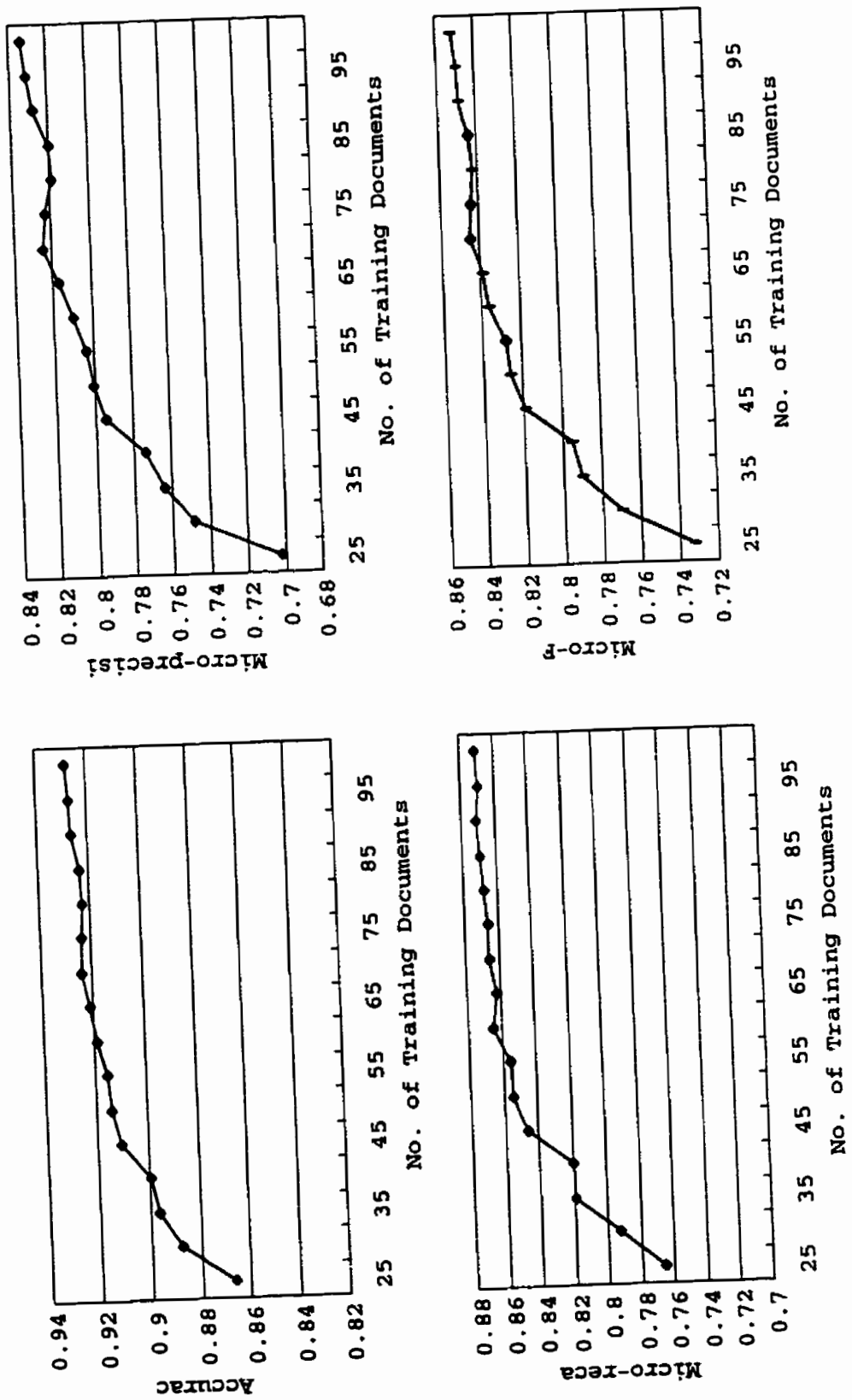
GasPrice	USDlrup	YanVal	USInt	JpnInt
100	208	92	138	31

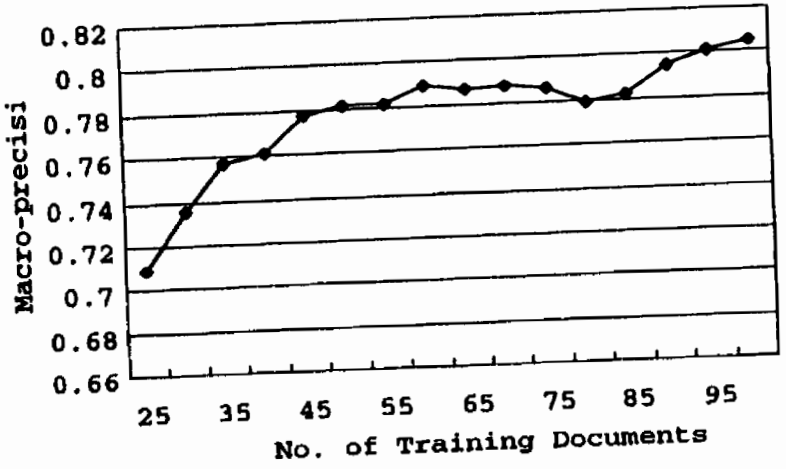
Table 6.1 The Document Distribution under the Five Model Nodes

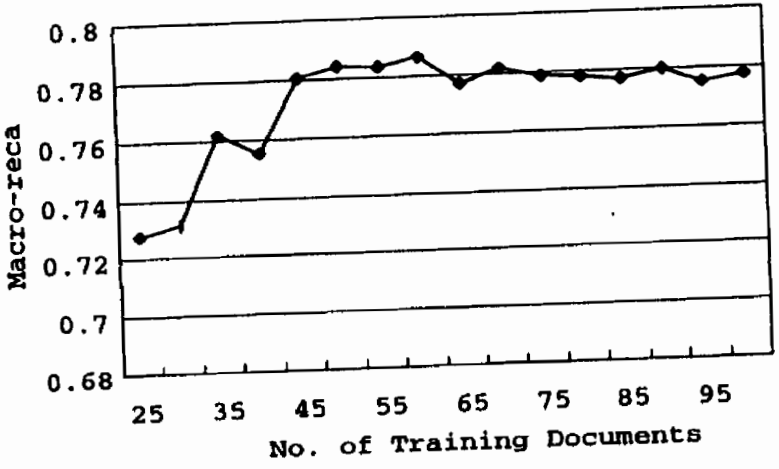
Among these documents, there are 84 documents which are relevant to 2 nodes and 4 documents which are relevant to 3 nodes. The rest are relevant to only one node.

6.3. The Experimental Results

The document generator in chapter 4 has been adjusted due to the different number of available documents. Firstly, the generator randomly selects 5 documents from each category, so the total number of initial training documents is 5×5=25. Secondly, 75 documents are selected randomly for incremental training. Finally, the rest of documents are used for testing. The generator also ensures that the three sets do not overlap.


Micro-precisi
0.84
0.82
0.8
0.78
0.76
0.74
0.72
0.7
0.68
25 35 45 55 65 75 85 95
No. of Training Documents
Micro-F
0.86
0.84
0.82
0.8
0.78
0.76
0.74
0.72
25 35 45 55 65 75 85 95
No. of Training Documents
Accurac
0.94
0.92
0.9
0.88
0.86
0.84
0.82
25 35 45 55 65 75 85 95
No. of Training Documents
Micro-reca
0.88
0.86
0.84
0.82
0.8
0.78
0.76
0.74
0.72
0.7
25 35 45 55 65 75 85 95
No. of Training Documents





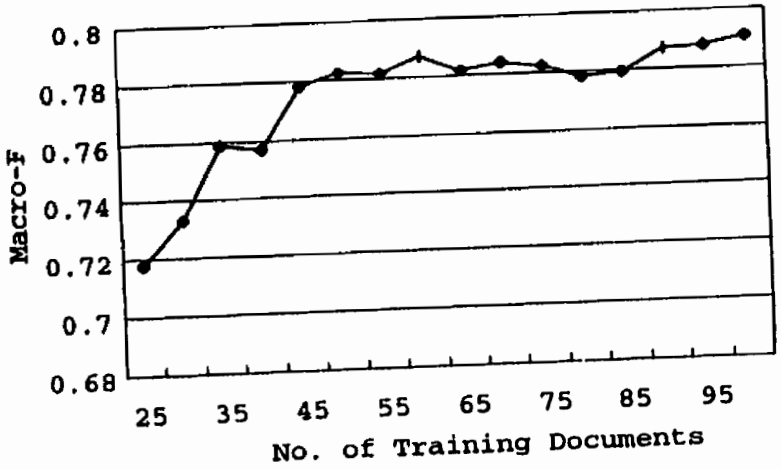


Figure 6.3 The Performance of the Incremental Classifier with respect to the Five Nodes in the Semantic Model

Figure 6.3 shows that the model classifier has accuracy of 0.866, micro-F1 of 0.732, and macro-F1 of 0.718, after it is initially trained with 5 sample documents per node. The classifier is then presented with more training documents incrementally. Accuracy, micro-F1, and macro-F1 rise to 0.928, 0.852, and 0.79, respectively, when the number of presented training documents reaches 100. There is a small performance degradation compared to the experiments on Reuters generic topics. We think the reason is that the five model elements are more fine-grained topics than the generic Reuters ones.

Chapter 7: Conclusions and Future Work

7.1. Conclusions

The thesis studies document classification in a knowledge management environment. We start with an introduction of the theoretic baseline of document classification, including the algorithms of the batch NB and kNN classifiers. We have proposed the adapted algorithms to make both the NB and kNN classifier incremental. We have suggested that the *category-classifier* architecture suits incremental document classification better than the *global-classifier* architecture does. Moreover, based on a study on feature selection for incremental classifiers, we have proposed two solutions to the feature selection problem: using local vector spaces instead of a global one, and using term frequency to limit the size of vectors. Consequently, these proposals and solutions theoretically conclude the feasibility of adapting existing batch classifiers to incremental ones.

We have designed a series of experiments to compare implementation options applicable to either the NB or kNN classifier, including term weighting scheme, similarity computation, the value of k, and feature selection. An efficient implementation has been proposed for both the incremental NB and kNN classifiers. An overall performance comparison shows that the kNN classifier outperforms the NB classifier significantly, provided that both of them are implemented optimally. The performance results are encouraging.

We have described the overall architecture of the EXIP, and proposed our solutions for integrating the document classification component. Furthermore, we build an example semantic model for an auto manufacturer in North America. Our proposed

incremental document classifier has been tested with respect to the semantic model. This series of experiments have also achieved affirmative results for the classification performance.

7.2. Future Work

First of all, we choose the NB and kNN classifiers for our study, however the study can be extended to other document classification algorithms. It is interesting to adapt more batch classifiers into incremental ones and make a broader performance comparison.

Secondly, the incremental classifiers in the thesis perform learning passively. That is the classifiers accept all training data that they are presented without any selection. Some recent studies [Schohn2000] [Tong1998] have proposed active learning. For example, [Scholhn2000] found an unusual phenomenon with the learning curves of the SVM classifier: when training examples were added via an active heuristic, performance peaked to a level above that achieved by using all available data, then slowly degraded to a level achieved by a random learning when all data had finally been added. Similarly, the kNN and NB classifiers can also improve performance by using active learning. For example, the kNN classifier can choose training documents according to their location in the training document space. An evenly distributed training document set is expected to attain better performance than a clustered training set.

Thirdly, we propose a document clustering technique intended to support evolvability of the semantic model. We have proposed the group average clustering algorithm in section 5.5, however a more thorough study of currently available techniques and an implementation of the proposed algorithm are needed to evaluate performance.

Finally, although we have performed classification experiments using an example semantic model and a number of relevant documents collected from the Reuter-21578 collection, this work needs to be tested with more complete examples of the semantic model of the EXIP and real document sets for strategic business analysts.

Bibliography

[Chakrabarti 97] S. Chakrabarti, B. Dom, R. Agrawal, P. Raghavan: *Using Taxonomy, Discriminants, and Signatures for Navigating in Text Databases*, Proc. of the 23rd Int'l Conference on Very Large Data Bases, Athens, Greece, August 1997.

[Baker1998] L.D. Baker, A.K. McCallum. *Distributional Clustering of Words For Text Classification*. Proceedings of the 21st annual international ACM SIGIR conference. 1998, Melbourne Australia

[Church89] K.W. Church, P. Hanks. *Word Association Norms, Mutual Information and Lexicography*. In Proceedings of ACL 27, pp76-83, Vancouver, Canada, 1989

[Cohen1996] W.W. Cohen, Y. Singer. *Context-sensitive learning methods for text Categorization*. In SIGIR '96: Proccdings of the 19th Annual International ACM SIGIR conference on Research and Development in Informantion Retrieval, 1996.

[Craven1998] M. Craven, D. DiPasquo, D. Freitag, et al. *Learning to Extract Symbolic Knowledge from the World Wide Web*. Proc. of 15th National Conference on Artificial Intelligence. (AAAI-98)

[Cutting1992] Cutting, D.R., Karger, D.R., Pedersen, J.O., Tukey, J.W., "Scatter/ Gather: A Cluster-based Approach to Browsing Large Document Collections," Proceedings Fifteenth Annual International ACM SIGIR Conference, pp318–329, 1992.

[Dasarathy1991] B.V Dasarathy. *Nearest Neighbor (NN) Norms : NN pattern classification techniques*. McGraw-Hill Computer science Series. IEEE Computer Society Press, Las Alamitos, California, 1991.

[Domingos1997] P. Domingos, M. Pazzani. *On the Optimality of the Simple Bayesian Classifer Under Zero-one Loss*. Machine Learning, 29:103-130, 1997.

[DXX] http://www-4.ibm.com/software/data/db2/extenders/xmlext/

[Friedman1997] N. Friedman, D. Geiger, M. Goldszmidt. *Bayesian Network Classifers*. Machine Learning, 29:131-163, 1997.

[Fuhr1991] N. Fuhr. *Air/x – a rule-based multi-stage indexing systems for large subject fields*. In 606-623, editor, Proceedings of RIAO '91, 1991.

[Hayes1990] P.J. Hayes, P.M. Andersen, I.B. Nirenburg, L.M. Schmandt. *TCS: A Shell for Content-based Text Categorization*. In Proceedings of the Sixth IEEE Conference on Artificial Intelligence Applications, pages 320-326, 1990

[Iwayama1995] M. Iwayama, T. Tokunaga. *Cluster-based Text Categorization: A comparison of Category Search Strategies*. ACM Int'l SIGIR Conference on Research and Development in Information Retrieval. pp273-281, 1995.

[Jarvis2001] R. Jarvis. "*Modeling and Analysis of Strategic Business Objectives*" Master Thesis, University of Toronto, Toronto, Canada, 2001

[Joachims1998] T. Joachims. *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*. Proceedings of the European Conference on Machine Learning, Springer, 1998.

[Lewis1994] D.D Lewis and M. Ringuette. *Comparison of Two Learning Algorithms for Text Categorization*. In Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94), 1994

[Lewis1996] D.D. Lewis. *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, 1996.

[Lieberman 97] H. Lieberman. *Autonomous Interface Agents*. ACM Conference on Human-Computer Interface [CHI-97], Atlanta, March 1997.

[MacKay] David J. C. MacKay. *A Short Course in Information Theory*. January 1995. Cavendish Laboratory, Cambridge, Great Britain. http://131.111.48.24/pub/

mackay/info-theory/course.html

[Manning1999] C.D. Manning, H. Schutze. Foundations of Statistical Natural Language Processing. The MIT Press. Cambridge, Massachusetts, 1999

[Masand1992] B. Masand, G. Linoff, and D. Waltz. *Classifying News Stories Using Memory Based Reasoning.* In 15th Annual Int'l ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'92), pp59-64, 1992.

[McCallum1998] A. McCallum and K. Nigam. A Comparison of Event Models for Naive Bayes Text Classification. AAAI-98 Workshop on "Learning for Text Categorization". 1998

[Mitchell1996] T. Mitchell. Machine Learning. McCraw Hill, 1996

[Mylopoulos2001] J. Mylopoulos, A. Barta, R. Jarvis, P. Rodriguez-Gianolli, and S. Zhou. *Managing Knowledge for Strategic Business Analysts: The Executive Information Portal.* (submitted for publication.)

[Ng1997] H.T. Ng, W.B. Goh, K.L. Low. *Feature Selection, Perceptron Learning, and a Usability Case Study for Text Categorization.* Proceedings of the 20th annual international ACM SIGIR conference, pp67-73, 1997, Philadelphia, PA USA

[Poet] http://www.poet.com/products/cms/cms.html.

[Quinlan1996] J.R. Quinlan. *Learning Decision Tree Classifiers.* ACM Computing Surveys. Vol. 28, No. 1, March 1996

[Quinlan93] J.R. Quinlan, R.M. Cameron-Jones. *FOIL: A Midterm Report.* Proc. of the 12th European Conference on Machine Learning. 1993

[Reuters] David D. Lewis. http://www.research.att.com/~lewis/reuters21578.html

[Scholm2000] G. Scholhn, D. Cohn. *Less is More: Active Learning with Support Vector Machines.* Proc. 17th International Conf. on Machine Learning, 2000

[Schutze1995] H. Schutze, D.A. Hull, J.O. Pedersen. *A Comparison of Classifiers and Document Representations for the Routing Problem.* Proceedings of the 18th annual international ACM SIGIR conference. 1995, pp229-237, Seattle, Washington

[Shilakes98] Shilakes, C., and Tylman, J., "Enterprise Information Portals," Merrill Lynch, November 16, 1998.

[Tomek76] I. Tomek. *A generalization of the k-NN Rule.* IEEE Transactions on Systems, Man, and Cybernetics, Volume SMC-6, No. 2, pp121-126, Feb. 1976.

[Tong1998] S. Tong, D. Koller. Support Vector Machine Active Learning with Applications to Text Classification. In Proceedings of the Seventeenth International Conference on Machine Learning (ICML-00), pp287-295, June 1998

[van Rijsbergen1979] C.J. van Rijsbergen. Information Retrieval. Butterworths, Second Edition, London. 1979

[Vapnik1995] Vladimir N. Vapnik. *The Nature of Statistical learning Theory.* Springer, New York, 1995.

[Verity] http://www.verity.com/products/docnav/index.html.

[Wiener1995] E. Wiener, J.O. Pedersen, A.S. Weigend. *A Neural Network Approach to Topic Spotting.* Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval

[Wilbur1992] J.W. Wilbur, K. Sirotkin. *The automatic identification of stopwords.* Journal of Information Science, Vol.18, pp45-55, 1992

[Yang1994] Y. Yang, C.G. Chute. *An Example-based Mapping Method for Text Categorization and Retrieval.* ACM Transaction on Information Systems (TOIS), 12(3): pp252-277, 1994

[Yang1995] Y. Yang. *Noise Reduction in a Statistical Approach to Text Categorization.*

In Proceedings of the 18th Annual Int'l ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'95), pp256-263, 1995.

[Yang1997] Yang, Y., Pedersen J.P. *A Comparative Study on Feature Selection in Text Categorization*. Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97), 1997.

[Yang1999] Yiming Yang and Xin Liu. *A re-examination of text categorization methods*. Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), 1999, pp 42--49.

[Young1974] T.Y. Young, T.W. Calvert. *Classification, Estimation and Pattern Recognition*. Elsevier, 1974.