A New Term Weighting Method for Text Categorization

By

Man Lan

Submitted For The Degree Of

Doctor of Philosophy

at

Department of Computer Science

School of Computing

National University of Singapore

3 Science Drive 2, Singapore 117543

September, 2006

Name: Man Lan
Degree: Doctor of Philosophy
Department: Department of Computer Science
Thesis Title: **A New Term Weighting Method for Text Categorization**
Abstract: Text representation is the task of transforming the content of a textual document into a compact representation of its content so that the document could be recognized and classified by a computer or a classifier. This thesis focuses on the development of an effective and efficient term weighting method for text categorization task. We selected the single token as the unit of feature because the previous researches showed that this simple type of features outperformed other complicated type of features.

We have investigated several widely-used unsupervised and supervised term weighting methods on several popular data collections in combination with SVM and kNN algorithms. In consideration of the distribution of relevant documents in the collection and analysis of the term's discriminating power, we have proposed a new term weighting scheme, namely $tf.rf$. The controlled experimental results showed that the term weighting methods show mixed performance in terms of different category distribution data sets and different learning algorithms. Most of the supervised term weighting methods which are based on information theory have not shown satisfactory performance according to our experimental results. However, the newly proposed $tf.rf$ method shows a consistently better performance than other term weighting methods. On the other hand, the popularly used $tf.idf$ method has not shown a uniformly good performance with respect to different category distribution data sets.
Keywords: Text Categorization, Term Weighting Method, Support Vector Machine, kNN.

To my parents and my husband.

ACKNOWLEDGEMENT

I would first thank my advisors Prof. Chew Lim Tan and Dr. Hwee Boon Low for their deep insights and dedication to guide and help me through this thesis research. Without their creative, valuable supervision, this work would have encountered a lot of difficulties.

I also sincerely appreciated the suggestions and insights I obtained from my former academic advisors: Professor Sam Yuan Sung for his suggestions on my preliminary thesis report in the Center for Information Mining and Extraction(*CHIME*) lab of School of Computing, National University of Singapore; Dr. Ah Hwee Tan currently with Nanyang Technology of University for giving me many useful suggestions during my working in the Text Mining lab of A-STAR Institute for Infocomm Research; Prof. Kang Lin Xie, in Shanghai Jiao Tong University, for encouraging me to further my education and research.

The former staff in the *CHIME* lab of School of Computing, National University of Singapore, Dr. Ji He, helped me with discussions, cooperations, encouragement, and making the research life in Singapore a very interesting and exciting experience.

Last but not least, to my loving parents and my husband, for their support and encouragement through all these years in the Ph.D. program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Motivation

In recent decades, with the explosive growth of textual information available in the World Wide Web, the ensuing needs of organizing and accessing these documents in flexible ways also increased. Text categorization (TC) is such one solution to this problem, which classifies natural language documents into a predefined set of semantic categories.

Automatic text categorization plays a crucial role in many applications to sort, direct, classify, and provide the proper documents in a timely and correct manner. It is a basic building block in a wide range of contexts, ranging from document indexing, to document filtering, word sense disambiguation, population of

hierarchical Web catalogues, and in general any application requiring document organization or selective and adaptive document dispatching.

In this thesis, we generally treat the older term *document categorization* and the newer terms *text categorization* and *text classification* as synonymous which are different from *text clustering*. The term "*text categorization*" or "*text classification*" is also called *supervised text classification* which has known label of training data set in advance and automatically assigns the documents to a predefined set of categories. This is the main topic of this thesis. However, the term "*text clustering*" is called *unsupervised text classification* and it performs without any known labelled data set. Therefore, aside from the meaning of text categorization, the term "*text clustering*" has also been used to mean the automatic identification of such a set of categories *and* the grouping of documents under them.

Generally, building an automated TC system consists of two key subtasks. The first task is text representation which converts the content of documents into compact format so that they can be further processed by the text classifiers. Another task is to learn the model of a text classifier which is used to classify the unlabelled documents.

The algorithms which have been applied to TC task have been studied extensively in recent decades and most of them are usually borrowed from the traditional machine learning (ML) domain, such as Support Vector Machines (SVMs), kNN, Decision Tree (DT), Naïve Bayes (NB), Neural Network (NN), Linear Regression (LR), etc. As a relatively new algorithm, SVM has a better performance than other methods due to its ability to efficiently handle relatively high dimensional and large-scale data sets without decreasing classification accuracy. In essence,

kNN makes prediction based on the k training patterns which are closest to the unlabelled (test) pattern. It is very simple and effective but not efficient in the case of high dimensional and large-scale data sets. The DT algorithm is sometimes quite effective but the consequent overfitting problem is intractable and needs to be handled manually case by case. The NB method assumes that the terms in one document are independent even this is not the case in the real world. The NN method, usually used in artificial intelligence (AI) field has shown lower classification accuracy than other machine learning methods.

The textual information is stored in many kinds of machine readable form, such as PDF, DOC, PostScript, HTML, XML and so on. Before the computer applies the text classifier to label the unknown document, the content of a document must be transformed into a compact and interpretable format so that it can be further recognized and classified by a computer or a classifier. This *indexing* procedure is called *text representation*. Apart from the inductive learning algorithms, text representation also has a crucial influence on how well the text classifiers can generalize. Moreover, once given a learning algorithm, choosing the text representation becomes the central modelling tool of building text classifier for several reasons. First, excellent algorithms are few. Second, since the rationale is inherent to each algorithm, the method is usually fixed for one given algorithm. Third, tuning the parameters of algorithm shows lower improvement than one might expect for the complexity of the algorithm. Furthermore, not all algorithms have such parameters to tune. Therefore, the focus of this thesis is on the text representation for text categorization.

In the traditional *vector space* model, the content of a document d is represented

as a vector in the term space, $d = (w_1, ..., w_{|T|})$, where $|T|$ is the size of the terms (sometimes called *features*) set and the value of w_k between $(0, 1)$ represents how much the term w_k contributes to the semantics of the document d. Thus, there are two important issues for text representation: (1) what a term should be and (2) how to weigh a term. As the basic indexing units for representing the content of documents, terms can be at different levels, such as sub-word level (syllables), word-level (single token), multi-word level (phrases, sentences), etc. Different terms have different importance in a text, thus an important indicator w_k (i.e. term weight, usually between 0 and 1) associated with each term represents how much the term w_k contributes to the semantics of document for TC. The widely-used term weighting methods are borrowed from traditional information retrieval (IR) field, such as *tf.idf*, *binary*, *term frequency* and so on.

As we just mentioned, among the promising approaches to TC, SVM has a better performance than others ([DPHS98], [Joa98], [LK02], [YL99], [DP03]). Generally, SVMs are classified into two categories, i.e. linear and non-linear based on the different kernel functions. Usually, the kernel functions of SVMs are crucial to the classifier's performance. However, for TC task, Leopold [LK02] points out that it is text representation schemes which dominate the performance of TC rather than the kernel functions of SVM. Therefore, choosing an appropriate text representation is more important than choosing and tuning kernel functions of SVM for TC. However, even given these previous studies, we could not definitely draw a conclusion as to which term weighting method is better than others for SVM-based text classifier, "Because we have to bear in mind that comparisons are reliable only when based on experiments performed ... under carefully controlled conditions" [Seb02]. In this case, various "background conditions" such as,

different data preparation (stemming, stop words removal, feature selection, different term weighting methods, etc), different benchmark data collections, different classifiers with various parameters, and even different evaluation methods (micro- and macro-averaged precision, recall, accuracy, error, break-even point, ROC, etc), have been adopted by different researchers. Therefore, a question surfaces here : "Among the various term weighting methods, which is the best term weighting method for SVM-based text classifier?"

The traditional term weighting methods, such as *binary*, $tf.idf$ and its variants, are usually borrowed from the IR domain. In contrast to IR, TC is a form of *supervised* learning as it makes use of prior information on the membership of training documents in predefined categories. This known information is effective and has been widely used in the step of feature selection [YP97] and the supervised learning of text classifier. Recently, researchers proposed to combine this prior information into term weighting methods. Since these supervised term weighting methods take the document distribution into consideration, they are naturally expected to be superior to the unsupervised (traditional) term weighting methods. However, not much work has been done on their comprehensive comparison with unsupervised term weighting methods. Although there are partial comparisons in [DS03] and [SM05], these supervised term weighting methods have shown to give mixed results with respect to $tf.idf$ and in most cases they have no superiority over $tf.idf$. On the other hand, another work [DTY$^+$04] has replaced idf factor with χ^2 factor as term weighting method and drawn a conclusion that χ^2 is better than idf, which is quite contrary to the finding in [DS03]. Therefore, two fundamental questions arise here, i.e. "Are supervised term weighting methods based on known information able to lead to better performance than unsupervised ones for text

categorization?" and "Can we propose a new effective and efficient term weighting method by making use of this prior information given by the training data set?"

One goal of this thesis is to propose a new effective and efficient term weighting method for TC task by using the prior information given by training data set. Meanwhile, we also give an analytical explanation of terms' discriminating power with empirical observation and explore the classification performance of several widely-used term weighing methods for SVM-based text classifier. Another goal is to examine the superiority of supervised term weighting methods and investigate the relationship between term weighting methods and learning algorithms. There are three critical questions that this thesis will address:

1. How can we propose a new effective term weighting method by using the important prior information given by the training data set?

2. Among the various term weighting methods, which is the best term weighting method for SVM-based text classifier?

3. Are supervised term weighting methods able to lead to better performance than unsupervised ones for text categorization? What kinds of relationship can we find between term weighting methods and the two widely-used learning algorithms, i.e. kNN and SVM, given different benchmark data collections?

To address these three questions, this thesis is divided into three subtasks:

- First, we will analyze and investigate the terms' discriminating power with different term weighting methods and propose a new term weighting method

to improve the performance of TC. From these investigations and analysis, we will gain insights into a better understanding concerning the intuitive idea of our newly-proposed supervised term weighting method.

- Second task is to explore term weighting methods for SVM-based text classifier and provide insights into the difference between various traditional term weighting methods and their variants for TC task. The empirical answer to this question is definitely interesting to researchers who would like to choose an appropriate term weighting method for SVM-based TC.

- Finally, we will examine the superiority of supervised term weighting methods and investigate the relationship between different term weighting methods and different learning algorithms on more general experimental conditions. This work will answer the third question with empirical evidence and give a practical guidance on how to choose term weighting methods in terms of different learning algorithms. Moreover, we will also extend our study to a new application domain, i.e. biomedical literature classification.

1.2 Structure of the Thesis

Each of the remaining chapters of the thesis captures different aspects of the work, including approaches to text categorization, text representation and term weighting methods, analysis of term's discriminating power and details of the research infrastructure. Below is a roadmap of the remaining chapters of this thesis.

Chapter 2 provides a review of techniques for the task of TC including the definition, its relationship with IR and ML, its taxonomy, applications, and most

important, the widely-used approaches to learning text classifiers. This chapter sets the context for the background of this thesis research.

Chapter 3 reviews the techniques for text representation and proposes our new term weighting method. To illustrate the differences among these term weighting methods, we analyze the terms' discriminating power from qualitative aspect and then propose a new term weighting method, i.e. $tf.rf$. From the further quantitative comparison of their discriminating power in the real cases, we gain an insight into a better understanding regarding the basic idea behind $tf.rf$. This chapter provides a detailed qualitative analysis and explanation of our newly proposed supervised term weighting method and partial answers to the first question that this thesis study will address from qualitative analysis aspect.

Chapter 4 lays out the methodology of research for the experiments in this thesis including the inductive learning algorithms, the benchmark data collections, text preprocessing, performance evaluation and statistical significance tests. This chapter provides a detailed description of all the experimental settings in this thesis.

Chapter 5 presents a series of experiments to investigate various widely-used traditional and the state-of-the-art supervised term weighting methods on various experimental conditions. The purpose of this chapter is to seek answers to the three questions with more general experimental evidence. To accomplish this, we build a fixed universal platform to compare a variety of traditional and supervised term weighting methods with our $tf.rf$ using a cross-classifier, cross-corpus and even cross-domain validation. This chapter serves an important role in this thesis since it not only examines the performance of various term weighting methods

with more experimental evidence, but also provides us with deeper insights and a practical guidance on choosing term weighting methods in terms of different learning algorithms and corpora.

Chapter 6 summarizes the contributions of this thesis and outlines some possible directions for future research.

In this study, we only focus on the the study of text representations rather than the improvements of inductive learning for TC. The default language on which we studied is English. Whether this new term weighting method is able to lead to different results for different languages is not clear. In addition, when we focus on the study on term weighting schemes for TC, we only change the term weighting schemes by using the bag-of-words approach, while the remaining background conditions such as data preparation, classifier and evaluation measures remain unchanged.

The results of this study could be useful for researchers to choose an appropriate term weighting method for TC; to find the relationship between term weighting methods and various widely-used learning algorithms; and as such finally to improve the performance of automatic TC from the text representation aspect.

CHAPTER 2
A BRIEF REVIEW OF TEXT CATEGORIZATION

This chapter presents background knowledge of the TC task and the techniques for building text classifiers. We begin by introducing the definition of TC. Generally, TC is considered as a field at the crossroads of machine learning (ML) and information retrieval (IR) since it shares a number of characteristics with these two fields. This chapter gives a broad overview of the TC system from the ML aspect. In the next chapter, we will discuss text representation from the IR aspect.

This chapter is organized as follows. Section 2.1 describes the concept of TC. Section 2.2 discusses its relationship with IR and ML. Sections 2.3 reviews various subcases of TC tasks. Section 2.4 introduces its most important applications.

Finally Section 2.5 reviews several most popular inductive learning algorithms in TC.

2.1 A Definition of Text Categorization

Text categorization is the task of assigning unlabelled documents into predefined categories. Assume D is a domain of documents and $C = \{c_1, c_2, ..., c_{|C|}\}$ is a set of predefined categories. Then the task is, for each document $d_j \in D$, to assign a decision to file d_j under c_i or a decision not to file d_j under c_i ($c_i \in C$) by virtue of a function Φ, where the function Φ is also called the *classifier* (or *model*, or *hypothesis*, or *rule*).

The TC task we will discuss in this thesis relies only on the semantics of a document. Hence, we will assume in this thesis that:

- The category label is just symbolic, and no additional knowledge of the meaning is available.
- Only endogenous knowledge extracted from the semantics of documents is available. Other exogenous knowledge, such as, publication date, document type, publication source and other metadata, is inaccessible.

Moreover, other types of TC tasks that are not dependent on semantics only will not be discussed in this thesis. For example, text sentiment classification is a task to classify a document according to the positive or negative polarity of its opinion (favorable or unfavorable, see [PLV02]). Another example is text genre classification, which differs from text classification as it discriminates between the

styles of the documents as opposed to the latter which discriminates between the topics of the documents (see [LM02]).

2.2 Relationship With Information Retrieval and Machine Learning

The study of TC dates back to the early '60s, but only in the recent decade it did become a major subfield of the information systems discipline. Generally, TC is considered as a discipline at the crossroads of machine learning (ML) and information retrieval (IR) as it shares a number of characteristics with these two fields.

Since TC is a content-based document management task, it heavily relies on the basic machinery of IR and shares the characteristics with IR in the following steps:

1. IR-style *indexing* which is performed on the training documents and on those to be classified during the later inductive learning step;
2. IR-style *induction* which is used to construct the inductive text classifier;
3. IR-style *evaluation* which performs to assess the performance of the classifier.

Dating back to the '80s, the most popular approach to building automatic document classifiers was composed of manually constructing an expert system capable of taking text classification decisions by means of *knowledge engineering* (KE) techniques. The most famous example of this approach is the CONSTRUE

system [HANS90], built by Carnegie Group for the Reuters news agency. Table 2.1 shows a sample rule of the type used in CONSTRUE system; key words are indicated in *italic*, categories are indicated in SMALL CAPS.

Table 2.1: A Rule-based classifier for the WHEAT category of Reuters Corpus in CONSTRUE system.

if	((*wheat* & *farm*)	**or**
	(*wheat* & *commodity*)	**or**
	(*bushels* & *export*)	**or**
	(*wheat* & *tonnes*)	**or**
	(*wheat* & *winter* & ¬ *soft*))	**then** WHEAT **else** ¬ WHEAT

Clearly, the drawback of this approach is the *knowledge acquisition bottleneck* which is a well known built-in problem from the expert system, that is, the rules must be manually defined by a knowledge engineer with the aid of a domain expert (an expert in the membership of documents in the chosen set of categories). Thus, if the set of categories is updated, these two professionals must intervene again, and if the classifier is transported to a completely different domain (i.e. set of categories), a different domain expert is needed to intervene and the work has to be repeated from scratch.

Since the early '90s, the machine learning approach to TC has gained popularity and has eventually become the dominant one. In this approach, a general inductive process (also called the *learner*) automatically builds a classifier for a category c_i by observing the characteristics of a set of documents manually classified under c_i or under $\overline{c}_i$ by a domain expert; from these characteristics, the inductive process

gathers the characteristics that a new unseen document should have in order to be classified under c_i.

The advantages of the ML approach over the KE approach are quite evident. The KE approach puts efforts on construction of classifier with the aid of domain expert. Thus, this construction is done manually not automatically. On the other hand, the ML approach endeavors to construct an automatic classifier. This means that if an inductive learning process is available off-the-shelf, all that one needs to do is to inductively and automatically construct a classifier from a set of manually classified documents, namely, training data set. The advantage is more evident if the classifier already exists and the original set of categories is updated, or if the classifier is transported to a completely different domain.

Classifiers built by means of ML techniques nowadays achieve impressive progress of effectiveness and efficiency, making automatic classification a qualitatively viable alternative to manual classification. We will discuss several promising methods that have been most popular in TC in Section 2.5.

2.3 Various Subcases of Text Categorization Tasks

Usually, the inductive approaches to building text classifiers cannot be applied to TC directly because several constraints may be enforced on the TC tasks according to different applications. Next we describe the techniques for these two subcases of TC tasks.

2.3.1 Single-label and Multilabel Text Categorization

Since semantics is a *subjective* notion, the membership of a document in a category cannot be decided deterministically. In fact, this inconsistency happens with very high frequency in the real world when two human experts decide whether to classify document d_j under category c_i. For example, given a news article on President Bush attending a WTO conference it could be filed under **Politics**, or under **Economics**, or under both, or even under neither, depending on the subjective judgement of the expert.

Thus, the case in which exactly one category must be assigned to each document d_j is often called the *single-label* classification, while the case in which any number of categories from 0 to $|C|$ may be assigned to the same document d_j is called the *multilabel* classification. A special case of single-label text categorization is *binary* classification, in which each document d_j must be assigned either to category c_i or to its complement $\overline{c}_i$.

The binary classification is more general than the multilabel classification since an algorithm for binary classification can also be used for multilabel classification. To do this, one needs only transform the problem of multilabel classification under $\{c_1, c_2, ..., c_{|c|}\}$ into $|C|$ independent problems of binary classification under $\{c_i, \overline{c}_i\}$, for $i = 1, ..., |C|$. That is, for each given positive category c_i, when we build a classifier for c_i, all the other categories are combined together as the negative category $\overline{c}_i$. This transformation requires that these $|C|$ categories should be stochastically independent of each other, that is, for any c_m and c_n $(m, n \in [1, |C|])$, the value of the model for category c_m does not depend on the value of the model for category c_n and vice versa. Typically this is assumed to be the case. (This is not the case

in hierarchical classification which we will discuss next.)

However, the converse transformation is not true: an algorithm for multilabel classification cannot be used for either binary or single-label classification. There are two cases that need to be considered. Given a document d_j to classify, (i) the classifier might attribute $k > 1$ categories to d_j, and it might not be obvious how to choose a "most appropriate" category from them; or (ii) the classifier might attribute to d_j no category at all, and it might not be obvious how to choose a "least appropriate" category from C. Thus, it is not a typical case to assign only one and the most appropriate category to each document in the corpus.

In this thesis we also adopt this splitting techniques to deal with the binary case for two reasons, i.e. (1) many important TC applications consist of binary classification problems and (2) solution to the binary case can be extended to the multilabel case. Note that since handling multilabel classification is also a research area in its own right, choosing to naively combine binary classifiers is only one widely-adopted technique in current TC literature.

2.3.2 Flat and Hierarchical Text Categorization

Imagine that there is a hierarchy with two top-level categories, *Computers* and *Sports*, and three subcategories within each, namely, *Computers/Hardware*, *Computers/Software*, *Computer/Chat*, *Sports/Football*, *Sports/Basketball*, *Sports/Chat* as Figure 2.1 shows.

In the flat non-hierarchical classification case, a model corresponding to a positive category is learned to distinguish the target category from all the others

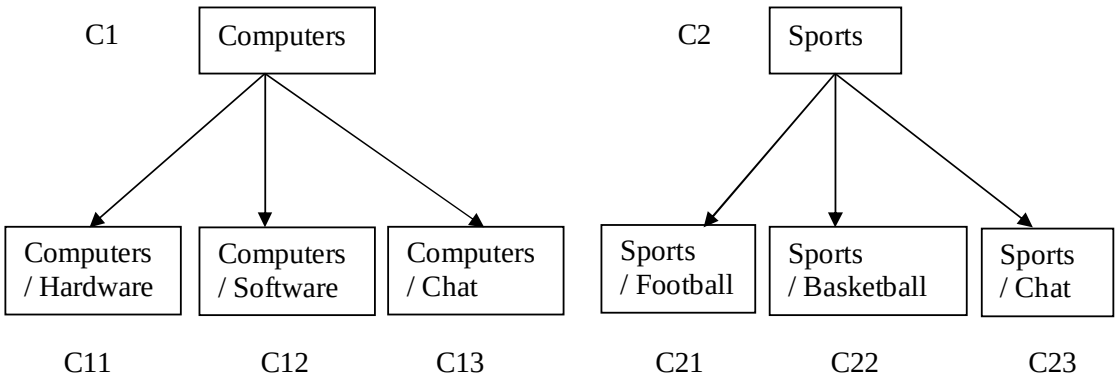

Figure 2.1: A Two-Level Hierarchy in Text Categorization

categories. However, in the hierarchical classification case, a model corresponding to a positive category is learned to distinguish the target category from other categories within the same top level. In the example shown in Figure 2.1, the text classifiers corresponding to each top-level category, *Computers* and *Sports*, distinguish them from each other. This is the same as flat non-hierarchical TC. On the other hand, the model corresponding to each second-level category is learned to distinguish the second-level category from other categories within the same top-level category. Specifically, the model built on category *Computers/Hardware* distinguishes itself from the other two categories under *Computers* category, namely, *Computers/Software* and *Computer/Chat*.

Hierarchical TC has recently aroused a lot of interest also for its possible application in automatically classifying web pages which are under the hierarchical catalogues. Since the categories in a hierarchical structure are not independent of each other, the binary classifiers discussed in the previous subsection cannot be applicable. To solve it, this hierarchical text classification problem is usually decomposed into a set of smaller problems corresponding to hierarchical splits in

the tree by using this known hierarchical structure. That is, one first learns to distinguish among categories at the top level, then lower level distinctions are learned only within the appropriate top level of the tree. Each of these sub-problems can be solved much more efficiently, and hopefully more accurately as well. Techniques exploiting this intuition in a TC context have been presented by [DC00].

2.4 A Variety of Applications of Text Categorization Technology

TC techniques have been used for a number of different applications. Although we group these applications into different cases, the borders between them are fuzzy and somehow artificial, that is, some of these cases could be considered special cases of others. Here we only discuss the most important ones. Other applications in combination with the availability of multimedia resources and/or other information extraction techniques will not be discussed in this thesis, for example, the speech categorization by means of a combination of speech recognition and TC techniques [KMW00] [SS00], the image categorization with textual title [SH00], etc.

2.4.1 Automatic Document Indexing for IR Systems

The most early research of TC techniques originated from automatic document indexing for IR systems. In this case each document is assigned one or more key words or key phrases (from a finite word set called *controlled dictionary*) describing its content. The controlled dictionary often consists of a thematic hierarchical thesaurus. For example, the NASA thesaurus for the aerospace discipline, or the

MESH thesaurus for the biomedical literature. Automatic indexing with a controlled dictionary is also closely related to *automated metadata generation*. In digital libraries, one is usually interested in tagging documents by metadata that describes them under a variety of aspects (e.g. creation date, document type, author, availability, etc.). Some of this metadata is *thematic*, that is, its role is to describe the semantics of the document by means of bibliographic codes, key words or key phrases.

Usually, this work is done by trained human indexers, and is thus a costly activity. However, if the entries in the controlled vocabulary or the thematic metadata are viewed as categories, document indexing is actually an instance of TC, and may thus be addressed by the general automatic techniques. Various text classifiers explicitly conceived for text indexing have been described in the literature, see [TH93], [RH84], [FK84].

2.4.2 Documentation Organization

Document organization may be the most general application of TC techniques to many kinds of textual information, such as ads, newspaper, emails, patents, conference papers, abstracts, newsgroup posters and so on. For example, the classification of incoming newspaper "classified" advertisements under different categories such as `Apartments or House for Rent/Sale`, `Cars for Sale`, `Job Hunting`, `Cheap Airfare`, `Vacation Packages`, the organization of patents into categories for making their search easier [Lar99], the automatic filing of newspaper articles under the appropriate sections (e.g., Politics, Home News, Lifestyles, etc.), or the automatic grouping of conference papers into sessions related to themes.

2.4.3 Text Filtering System

Text filtering is an activity of classifying a stream of incoming documents dispatched in an asynchronous way by an information producer to an information consumer (see [BC92]). One typical example is a news feed, where the producer is a news agency and the consumer is a newspaper (see [HANS90]). In this case, the filtering system should block the delivery of the documents the consumer is likely not interested in (e.g., all news not concerning `sports` in the case of a sports newspaper). In addition, a text filtering system may also further classify the documents deemed relevant to the consumer into different thematic categories. For example, all articles about `sports`, that is, the relevant documents, should be further classified according to which sport they deal with, so as to allow journalists specialized in individual sports to access only documents of prospective interest to them. Another example is junk e-mails filtering system. In recent years, junk e-mails have become an increasingly important problem with great economic impacts. Similarly, a junk e-mail filtering system may be trained to discard "spam" mails (see [AKCS00] and [HDW99]) and further classify non-spam mails into topical categories of interest to the user.

Information filtering by machine learning techniques has been widely discussed in the literature, see [AC99], [ILS+00], [KHZ00], [TKSK00] and [YL98].

2.4.4 Word Sense Disambiguation

Resolving natural language ambiguities is one important problem in computational linguistics as polysemous and homonymous words commonly exist in various types

of articles of different domains whether in English or other languages. For instance, the word, `bank` may have many different meanings in English. The two most common senses are a financial institute (as in `the Bank of National Development`) and a hydraulic engineering artifact (as in `the bank of river Thames`). Thus, identifying the meanings of words in given contexts is quite important for many linguistics applications, such as natural language processing (NLP), and indexing documents by word senses rather than by words for IR purposes. *Word sense disambiguation* (WSD) is such kind of activity of finding the sense of an ambiguous word, given the occurrence in a text of this particular word. Although a number of IE techniques have been adopted in WSD, another possible solution to WSD is to adopt TC techniques once we view word occurrence contexts as documents and word senses as categories (see [GY93] and [EMR00]) based on the assumption of one sense per discourse.

Other issues regarding resolving natural language ambiguities may all be tackled by means of TC techniques along the lines discussed for WSD, which include *context-sensitive spelling correction*, *prepositional phrase attachment*, *part of speech tagging*, and *lexical choice* in machine translation (see [Rot98] for an introduction).

2.4.5 Hierarchical Categorization of Web Pages

When documents are catalogued in this hierarchical way, a researcher may find it easier to first navigate in the hierarchy of categories and restrict his search to a particular category of interest. Therefore, many real world web classification systems have been built on complex hierarchical structure, such as Yahoo!, MeSH, U.S.Patents, LookSmart and so on. This hierarchical web page classification may

be solved by the hierarchical TC techniques we discussed in section 2.3.2. Previous research works exploiting the hierarchical structure in a TC context have been discussed by [DC00], [WWP99], [RS02], [MRMN98] and [CDAR98]. In practice, as a rich resource of information, links also have been exploited in web pages classification by [OML00], [GLF99], [F$\ddot{9}$9], [CDI98], [Att98] and experimentally compared by [YSG02].

2.5 Approaches to Effectively Learning Text Classifiers from Labelled Corpora

As we mentioned in Section 2.2, since recent decades, machine learning approaches to effectively learning text classifiers have been widely tackled in a variety of ways. In this section, we will deal only with the methods that have been most popularly applied in TC. Apart from the ML approaches, Rocchio is a unique approach which was borrowed from the traditional IR field and thus we also include it.

Usually, the construction of a classifier for each category $c_i \in C$ consists in the definition of a target function $\Phi : (D, C) \rightarrow [0, 1]$ which returns a value for a given document d_j. The returned value is usually between 0 and 1 which roughly represents the evidence for the fact that $d_j \in c_i$. Commonly, there is a preassigned *threshold* τ such that if the returned value from $\Phi(d_j, c_i) \geq \tau$, the document d_j is assigned to be positive category c_i and vice versa.

The target function of the classifier can be a *model*, a *hypothesis*, or a *rule*, which depends on the approach applied. For example, the Rocchio method builds an explicit *profile* of each category c_i which is a weighted list of the discriminative

terms whether present or absent under this category; k Nearest Neighbor is an *example-* (*sample-* or *instance-*) based classifier; the C4.5 algorithm learns *rules* by constructing a decision tree; Naïve Bayes classifier uses a *probabilistic* model of text; Support Vector Machines find the *hyperplane* which separates the positive and negative samples with the maximum margin, etc.

Note that other less standard or less popular approaches exist, such as Regression methods, Neural Networks, genetic algorithms, maximum entropy modelling, but they are not included in this thesis because in TC domain, they have no comparable performance to that of the above promising ones and/or have not been widely used in recent years. Moreover, there are several techniques that have been applied to improve the classification performance effectively and efficiently, such as majority voting (namely classifier committee), boosting, bootstrapping and so on. These techniques are not covered in this thesis either.

2.5.1 The Rocchio Method From Information Retrieval

The *Rocchio method* may be the only TC method rooted in the conventional IR field rather than in the ML field. It is used for inducing linear, profile-style classifiers, by means of an adaptation to TC of the well known Rocchio's formula for relevance feedback in the vector space model. The classifier built from the initial corpus, is in fact an explicit *profile*, that is, for each category c_i, it is a weighted list of the terms whose presence or absence is most useful for discriminating c_i. This adaption was first proposed by Hull [Hul94], and has been used by many authors since then, either as an object of research in its own right ([Joa97]), or as a baseline classifier ([Joa98], [CS96]), or as a member of a classifier committee.

The Rocchio method computes a classifier $\vec{c}_i = (w_{1i}, ...w_{|T|i})$ ($|T|$ is the term set size) for category c_i given an *initial corpus* $Tr = \{d_1, ..., d_{|Tr|}\} \subset D$ by means of the formula

$$w_{ki} = \beta \cdot \sum_{d_j \in POS_i} \frac{w_{kj}}{|POS_i|} - \gamma \cdot \sum_{d_j \in NEG_i} \frac{w_{kj}}{|NEG_i|} \tag{2.1}$$

where w_{kj} is the weight of term t_k in document d_j, $POS_i = \{d_j \in Tr | \breve{\Phi}(d_j, c_i) = True\}$, and $NEG_i = \{d_j \in Tr | \breve{\Phi}(d_j, c_i) = False\}$. In this formula, β and γ are two control parameters that allow setting the relative importance of positive and negative examples in the training data set. For instance, if β is set to 1 and γ to 0 (as in [DPHS98], [Hul94] and [Joa98]), the profile of c_i is the *centroid* of its positive training examples. Thus, the centroid-based text classifier is actually a special case of the Rocchio method. Clearly, a classifier built by means of the Rocchio method rewards the closeness of a test document to the centroid of the positive training examples, and its distance from the centroid of the negative training examples. Sometimes, the role of the negative examples is usually deemphasized by setting β to a high value and γ to a low one, e.g. [Joa97] (use $\beta = 16$ and $\gamma = 4$) and [CS96].

One issue in the application of the Rocchio formula to profile extraction is whether the set NEG_i should be considered in its entirety, or whether a well-chosen sample of it, such as the set $NPOS_i$ of near-positives (defined as "the most positive among the negative training examples"), should be selected from it. The $NPOS_i$ factor is more significant than NEG_i, since near-positives are the most difficult documents to tell apart from the positives. This method originates from the observation that, when the original Rocchio formula is used for relevance feedback in IR, near-positives tend to be used rather than generic negatives, as the

documents on which user judgements are available are among the ones that have scored highest in the previous ranking. Regarding this issue, see [SSS98], [RS99], [WWP99].

The obvious advantage of this method is interpretability, as such a *profile* is more easily understandable by a human than neural network classifiers, probabilistic classifiers or high-dimensional SVM classifiers. Another advantage is its ease of implementation and it is also quite efficient, since learning a classifier basically comes down to averaging term weights. On the other hand, in terms of effectiveness, a drawback of this method is that if the documents in the category tend to occur in disjoint clusters, such a classifier may miss most of them, as the centroid of these documents may fall outside all of these clusters. More generally, a classifier built by the Rocchio method, as all linear classifiers, has the disadvantage that it divides the space of documents linearly. Note that even most of the positive training examples would not be classified correctly by the linear classifier. Generally, the Rocchio classifier has always been considered as an underperformer and cannot achieve an effectiveness comparable to that of a state-of-the-art machine learning method. ([SSS98] improved its effectiveness comparable to that of a boosting method by using other enhancements.)

2.5.2 *k* Nearest Neighbor

k Nearest Neighbor (kNN) is a kind of example-based classifiers which do not build an explicit, declarative representation of the category c_i, but rely on the category labels attached to the training documents similar to the test documents. Other example-based methods exist, but kNN is the most widely-used one. In essence,

kNN makes its prediction based on the k training patterns that are closest to the unlabelled (testing) pattern, according to a distance metric. The commonly used distance metrics that measure the similarity between two normalized patterns include the Euclidean distance

$$Dis(p,q) = \sqrt{\sum_i (p_i - q_i)^2}, \tag{2.2}$$

the inner product

$$Dis(p,q) = \sum_i p_i q_i, \tag{2.3}$$

and the cosine similarity

$$Dis(p,q) = \frac{\sum_i p_i q_i}{\sqrt{\sum_i p_i^2}\sqrt{\sum_i q_i^2}}. \tag{2.4}$$

where p and q are two samples, p_i and q_i are attributes of the two samples respectively.

The early application of kNN algorithm for TC was in [YC94]. For deciding whether $d_j \in c_i$, kNN looks at whether the k training documents most similar to d_j also are in c_i; if the answer is positive for a large enough proportion of them, a positive decision is taken, and a negative decision is taken otherwise. The kNN used in [YC94] is actually a *distance-weighted* version. Then thresholding methods need to be used to convert the real-valued distance into binary categorization decisions. [YP97] and [YL99] used kNN based on the cosine similarity metric to measure the similarity between the two documents.

The construction of a kNN classifier also involves determining a threshold k that indicates how many top-ranked training documents have to be considered for computing the distance. [LC96] used $k = 20$, while [YL99] and [YC94] has

found $30 \leq k \leq 45$ to yield the best effectiveness. [Joa98] also achieved the best performance for kNN when $30 \leq k \leq 45$.

Unlike linear classifiers, kNN does not divide the document space linearly, and thus does not suffer from the problem discussed at the end of subsection 2.5.1. A number of different experiments have shown kNN to be quite effective. However, the most significant drawback is its inefficiency at classification time resulting from the its natural rationale in case of huge dimensional and huge-scale data sets. Unlike a linear classifier where only a dot product needs to be computed to classify a test document, kNN requires the entire training documents to be ranked for similarity with the test documents, which is much more expensive. Actually, kNN method may not be called an inductive learner as it does not have a true training (*learning*) phase and thus postpone all the computation to classification time.

2.5.3 Naïve Bayes Method

Naïve Bayes classifier is a probabilistic classifier which views the target function $\breve{\Phi}(d_j, c_i)$ in terms of the conditional probability $P(c_i|\vec{d_j})$, that is, it computes the probability that a document represented by a vector $\vec{d_j} = (w_{1j}, ..., w_{|T|j})$ of terms belongs to c_i. By an application of Bayes' theorem, this probability is given by

$$P(c_i|\vec{d_j}) = \frac{P(c_i)P(\vec{d_j}|c_i)}{P(\vec{d_j})} \tag{2.5}$$

where $P(\vec{d_j})$ is the probability that a randomly picked document has vector $\vec{d_j}$ as its representation, and $P(c_i)$ is the probability that a randomly picked document belongs to c_i.

In order to make the estimation of $P(\vec{d_j}|c_i)$ in (2.5) practical, it is common to make the assumption that any two coordinates of the document vector are statistically independent of each other when they are viewed as random variables; this *independence assumption* is encoded by the equation:

$$P(\vec{d_j}|c_i) = \prod_{k=1}^{|T|} P(w_{kj}|c_i) \tag{2.6}$$

The probabilistic classifiers that use this assumption are called *Naïve Bayes* classifiers, and account for most of the probabilistic approaches to TC in the literature, for example, [Joa98] and [Lew98].

Without the independence assumption, the estimation of $P(\vec{d_j}|c_i)$ is an impossible mission since the number of possible vectors $\vec{d_j}$ is too high. Although the *naive* character of this classifier makes the computation possible, this assumption is not verified in practice. In addition, the non-binary term weights cannot be applicable to this method[1].

To calculate $\prod_{k=1}^{|T|} P(w_{kj}|c_i)$, two models are used: one is a multi-variate Bernoulli model which is a Bayesian Network with no dependencies between words and binary word features; another is a multinomial model, that is, a uni-gram language model with integer word counts. [MN98] empirically compared their classification performance and found that the multi-variate Bernoulli performs well with small vocabulary sizes, but the multinomial model usually performs even better at larger vocabulary sizes. One prominent characteristic of the multinomial model is to relax the constraint that document vectors should be binary representation.

[1] [MN98] used a multinomial event model for Naïve Bayes text classification which can relax the constraint that document vectors should be binary-valued.

[Lew98] gives a thorough discussion about the probabilistic classifier and provides an excellent roadmap on the various directions that research on Naïve Bayes classifiers has taken, such as:

- to relax the constraint that document vectors should be binary-valued.
- to introduce document length normalization.
- to relax the independence assumption.

2.5.4 Decision Tree

Probabilistic methods are quantitative (i.e. numeric) in nature and hence they are not easily interpretable by humans. On the other hand, symbolic (i.e. nonnumeric) algorithms do not suffer from this problem. Inductive rule learners and decision tree learners are the most popular examples of symbolic algorithms.

A *decision tree* (DT) text classifier is a tree in which internal nodes are labelled by terms, branches departing from them are labelled by tests on the weight that the term has in the test document, and leaves are labelled by categories. Such a classifier categorizes a test document d_j by recursively testing for the weights that the terms labelling the internal nodes have in vector $\vec{d_j}$, until a leaf node is reached; the label of this node is then assigned to d_j. Most such classifiers use binary document representations, and thus consist of binary trees. Recall the DNF rule example shown in Table 2.1, a decision tree equivalent to that DNF rule is shown in Figure 2.2.

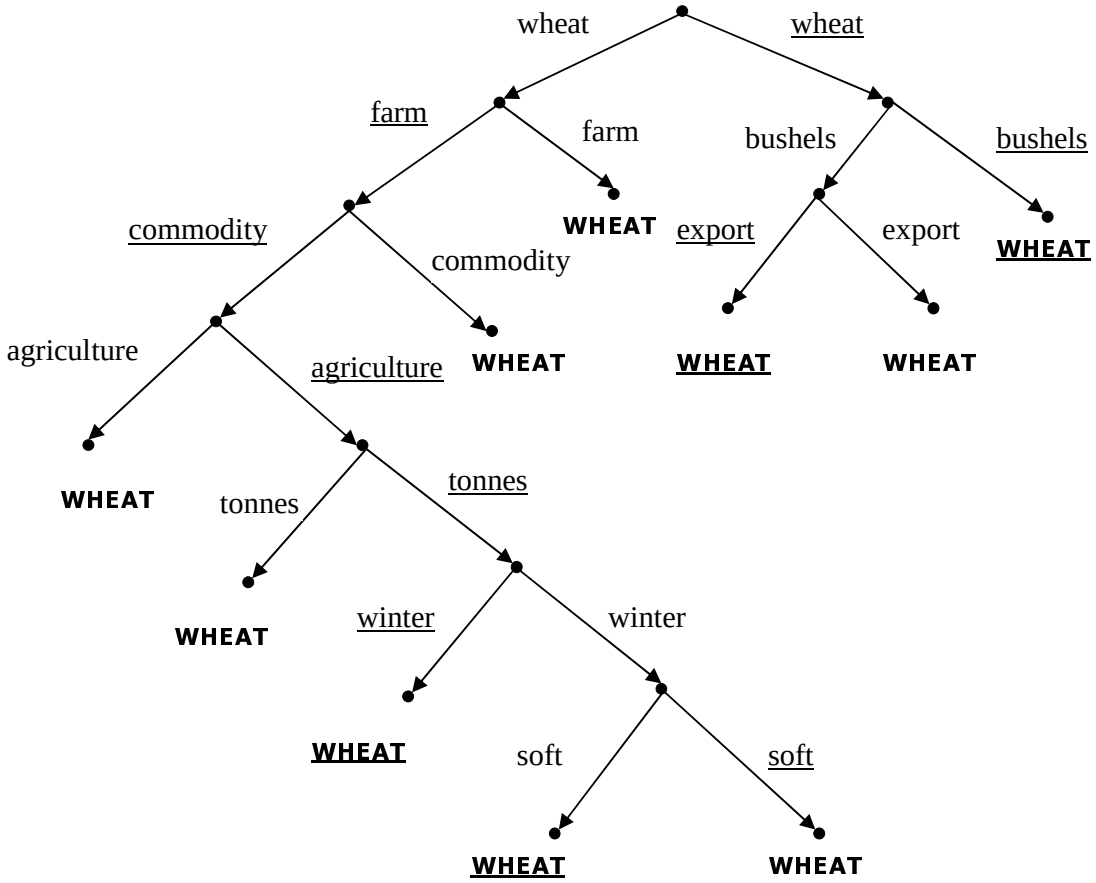

Figure 2.2: A decision tree equivalent to the DNF rule of Table 2.1. Edges are labelled by terms (underlining denotes negation) and leaves are labelled by categories (WHEAT in this example).

A possible method for learning a decision tree for category c_i consists in a "divide and conquer" strategy of:

- checking whether all the training examples have the same label (either c_i or $\overline{c}_i$);
- if not, selecting a term t_k, partitioning the training data documents into classes of documents that have the same value for t_k, and placing each such class in a separate subtree.

This process is recursively repeated on the subtrees until each leaf of the tree so generated contains training examples assigned to the same category c_i, which is then chosen as the label for the leat. The key step is the choice of the term t_k on which to operate the partition. Generally, such choice is made according to an information gain or entropy criterion.

One important issue of DT is overfitting as some branches of the "fully grown" tree may be too specific to the training data. Thus most decision tree learning methods include a method for growing and pruning the tree (i.e. removing the overly specific branches).

There are a number of standard packages for DT learning. Among them the most popular ones are ID3 (used by [FHK+91]), C4.5 (used by [CH98]) and C5 (used by [LJ98]). DT text classifiers have been used either as the main classification tool ([FHK+91]), or as baseline classifiers ([Joa98], [CS96]), or as members of classifier committees ([LJ98], [SS00]).

2.5.5 Support Vector Machines

Support vector machine (SVM) is a relatively new machine learning algorithm initially introduced by Vapnik and may have been first applied to text categorization by Joachims and Dumais [Joa98] [DPHS98]. Based on the *structural risk minimization* principle from computational learning theory, SVM seeks, among all the surfaces in $|W|$-dimensional ($|W|$ is the number of features) space that separate the training data examples into two classes, the surface (*decision surfaces*) that separates the positives from the negatives by the widest possible margin. Thus this best decision surface is determined by only a small set of training examples, known as *support vectors*, which are the only effective elements in the training data set; if all other training examples were removed, the algorithm will learn the same decision surface. This quite interesting property of SVM makes SVM theoretically unique and different from many other methods, such as kNN, Neural Network and Naïve Bayes where all the data examples in the training data set are used to optimize the decision surface [YL99]. And because of this, SVM is resilient to errors (noises or outliers), which results in its excellent performance.

SVMs are usually classified into two categories based on different kernel functions, namely, linear SVM and non-linear SVM (e.g. polynomial function, radial bases function (RBF) and etc). For example, the different kernel functions are *linear function*

$$K(x_i, x_j) = x_i^T x_j, \tag{2.7}$$

polynomial function

$$K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0, \tag{2.8}$$

and *radial based function* (RBF)

$$K(x_i, x_j) = e^{(-\gamma||x_i - x_j||^2)}, \gamma > 0 \tag{2.9}$$

for vectors (x_i and x_j), here, γ, r and d are kernel parameters.

In recent years, SVM has been extensively used in TC and has been shown better performance than other conventional machine learning algorithms to handle relatively high dimensional and large-scale training set, see [Joa98], [DPHS98], [YL99] and [LK02]. When applying SVM to TC, many researchers found that SVM with linear kernel function obtained better results than other SVMs with complicated kernel functions (See [YL99], [DPHS98]).

There are a number of off-the-shelf software packages which implement SVM with different kernel functions, such as SVM-Light, LIBSVM, TinySVM and so on, which can be applied on different operating system platforms.

2.5.6 A Summary of These Approaches

Among these popular approaches, we are quite interested to know which one is better. To evaluate a text classifier, we usually measure its effectiveness rather than its efficiency, that is, its ability to take the right classification decisions is significantly considered. Evaluation measure is the theme of Section 4.3.

To compare performance of the methods discussed, we have to bear in mind that comparisons are reliable only when based on experiments performed under carefully controlled conditions. However, various "background conditions" adopted by different authors may influence the results, which include, different benchmark

data collections, text preprocessing (stop words set, stemming, etc.), indexing, feature selection, classifier parameter values, and evaluation measures. Thus, though direct comparison makes reliable results, in most cases, only several methods are compared by the same author. Taking the relative performance of the classifiers reported by different authors into consideration, we may attempt a few conclusions:

- Support Vector Machines and kNN achieve top-notch performance. Specifically, [Joa98], [DPHS98] and [YL99] pointed out that SVM outperforms other machine learning methods compared in their experiments, including kNN.
- Rocchio and probabilistic Naïve Bayes classifiers look worst among the learning-based classifiers.
- The previous study is hardly sufficient to say anything about decision trees. The earlier literature reported unimpressive results (see [CS96], [Joa98]), while in the work by Dumais [DPHS98], a decision tree classifier was shown to perform nearly as well as the top performing system (a SVM classifier).

CHAPTER 3

TEXT REPRESENTATION FOR TEXT CATEGORIZATION

In this chapter, we discuss how to improve the performance of TC from the text representation aspect. Generally, there are two key issues involved in text representation, i.e. term type and term weight. Term can be at different levels, such as syllable, word, phrase, and other sophisticated semantic and/or syntactic representations by exploiting natural language processing knowledge. Different terms have different importance in a text and thus the term weighting methods assign appropriate weights to the terms to improve the performance of TC.

One important aspect of this chapter is to investigate various term weighting methods for TC given the bag-of-words approach. The most commonly used *tf.idf* is used for traditional IR tasks. Recently, there has been much effort spent on improving terms' discriminating power by exploiting information theory functions

or statistic metrics. Intuitively, it would seem that this knowledge with a solid theoretical basis should play an important role in improving terms' discriminating power for semantics of documents. However, many of them have not resulted in better performance. Even though some of them performed better in a few cases, the improvements did not achieve the desired level that one might hope when using such rich information sources.

Another important contribution of this chapter is to give a detailed analysis of terms' discriminating power and propose a new term weighting method, i.e. $tf.rf$. Thus, this chapter also provides an partial answer to the first question raised in this thesis with an analytical explanation, i.e."How can we propose a new effective term weighting method by using the important prior information given by the training data set?"

This chapter is organized as follows. Section 3.1 introduces the vector space model for TC. Section 3.2 presents the prerequisite techniques for text processing. Section 3.3 provides an overview of various term types in the previous study by other researchers. Section 3.4 reviews the factors for term weighting and various traditional term weighting methods from IR. Section 3.5 reviews the state-of-the-art supervised term weighting methods. Based on the analysis of terms' discriminating power in Section 3.6, we propose a new term weighting method, $tf.rf$, in Section 3.7. Furthermore, Section 3.8 compares several terms' discriminating

power with empirical observation in the real world cases.

3.1 Introduction

Even though text is already stored in machine readable form, such as HTML, PDF, DOC, PostScript, etc, it is generally not directly suitable for most learning algorithms. Therefore, before we apply one machine learning method to text categorization, the content of a textual document must be converted to a compact representation in order to be recognized and categorized by classifiers in a computer.

The vector space model (VSM) is a way of representing documents through the words that they contain. It is a standard technique in the IR field. Usually, in VSM, the content of a textual document is transformed into a vector in the term space, $d = (w_1, ..., w_{|T|})$, where $|T|$ is the term set size. The value of w_k between (0,1) represents how much the term w_k contributes to the semantics of the document d. Figure 3.1 shows one example represented in VSM. In the three term dimensions, namely, *system*, *retrieval* and *information*, there are six documents represented as six vectors in this 3-dimensional space. The number of dimensions of this space is built from all the words in all documents in the whole corpus. There is an assumption behind this model that documents that are "close together" in space are similar in meaning.

Generally, there are two issues involved for text representation:

1. What should a term be?

2. How to compute term weights?

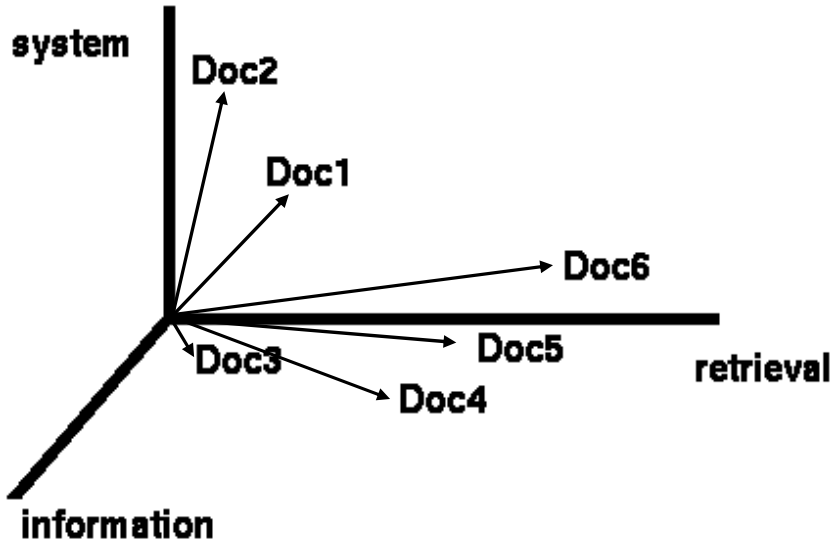

Figure 3.1: An example of vector space model

The first issue is the theme of Section 3.3. The second issue will be discussed in Section 3.4 and Section 3.5 from the traditional and the supervised methods aspects, respectively.

3.2 The Prerequisites of Text Representation

In VSM, term space consists of all the terms in all documents in the whole corpus. Therefore the number of dimensions (usually thousands or tens of thousands) is a big nightmare for the subsequent machine learning algorithms. This is known as the *curse of dimensionality*, which means that the computational cost must increase exponentially with the dimension of the problem. Therefore, before classifier induction, one often applies a pass of dimensionality reduction which is helpful to reduce the dimensionality of space and speed up the classification operation.

3.2.1 Stop Words

Consider this sentence:

```
The way to the school is long and hard when walking in the rain.
```

`The` appears three times. To save space, a search engine might replace it with a marker. The sentence would be stored like this:

```
* way to * school is long and hard when walking in * rain.
```

This explanation is simplified, but the point is that commonly occurring words are unlikely to give useful information and may be removed from the vocabulary to speed processing and save space. These frequent and uninformative words are called *stop words*. Commonly, they are topic-neutral words such as articles, prepositions, conjunctions, etc (such as, "`the`", "`a`", "`of`" and so on).

3.2.2 Stemming

In most cases, morphological variants of words have similar semantic interpretations and can be considered as equivalent for the purpose of IR applications. For example, "`computers`" and "`computer`" in one document should be represented as a single term as they share almost the same semantic interpretation.

For this reason, a number of stemming algorithms, or stemmers, have been developed to map several morphological forms of words to a common feature. Thus, the key terms of a query or a document are represented by stems rather than by the

original words. It does not usually matter whether the stems generated are genuine words or not, thus, "`computation`", "`computer`", "`compute`" might be stemmed to "`comput`" provided that (a) different words with the same "base meaning" are merged into the same form, and (b) words with distinct meanings are kept separate.

Whether stemming is suitable to TC is controversial. Sometimes it has been reported to hurt effectiveness ([BM98]), the recent tendency is to adopt it, as it reduces both the dimensionality of the term space and the stochastic dependence between terms, which results in a saving of storage space and processing time.

3.2.3 Features Selection

Feature selection is a process commonly used in ML field to reduce the dimensionality of the feature space, wherein a subset of the features available from the data are selected out. The selected features receive the highest scores according to a function that measures the "importance" of the feature (term) for TC task. The functions used to measure the "importance" are quite significant.

One simple and effective function is the *document frequency* of a term, that is, only the terms that occur in the highest numbers of documents are retained. A popular variant of this policy is to remove all terms that occur at most x times in the training data set, with popular values for x ranging from 1 to 5.

Other more sophisticated information theory functions have been used in the literature, among them are χ^2, *information gain*, *mutual information*, *odds ratio*, *relevancy score*, *GSS coefficient*, etc (see [YP97]). These functions try to capture the intuition that the best terms for category c_i are the ones distributed most differ-

ently in the sets of positive and negative examples of c_i. However, interpretations of this principle vary across different functions. Various experimental comparisons of these functions have been carried out, and most of them have improved on the results of document frequency ([YP97]). Collectively, the experiments seem to indicate that $\{$*odds ratio*, *GSS*$\} > \{\chi^2$, *information gain*$\} > \{$*mutual information*$\}$, where ">" means "performs better than". However, these results are just indicative because more general statements on the relative merits of these functions could be made only as a result of comparative experiments performed in thoroughly controlled conditions and on a variety of different situations (e.g. different classifiers, different initial corpora, etc.).

Feature selection is necessary either because it is computationally infeasible to use all available features, or because of problems of estimation when limited data samples (but a large number of features) are present. The latter problem is related to the curse of dimensionality problem.

3.3 What Should a Term Be?

As for the first issue of text representation, i.e. what should a term (feature) be, the answer is not unique. Generally, the basic units (also called *indexing terms*) for representing documents can be at different levels, such as sub-word level (syllables), word-level (single token), multi-word level (phrases, sentences), and semantic or syntactic level, etc. That is, on the word level, indexing terms refer to single words, while on the multi-word level, indexing terms refer to phrases or sentences.

3.3.1 Sub-Word Level

n-grams are the most popular representation on the sub-word level. Instead of using words as indexing terms, strings of n characters compose the basic building blocks. For example, the 2-gram (i.e. bi-gram) representation of the word "`computer`" is "_ c", "co", "om", "mp", "pu", "ut", "te", "er", "r_".

This n-gram representation can model similarity between words. For example, "`computer`" and "`computers`" are two different words but they share most of their bi-grams. However, sometimes this n-gram based similarity can also be misleading in case of "`computer`" and "`commuter`".

Clearly, the advantage of n-gram representation is that they provide some robustness against spelling errors. However, the rapidly increasing quantity of terms (for example, in 2-gram representation, the word `computer` is represented by using 9 terms as above.) is also an intractable challenge for most machine learning algorithms and degrades the efficiency of text preprocessing and the subsequent building of text classifier.

3.3.2 Word Level

Word-based representation is by far the most common way to represent the content of text. This representation (also called the *bag-of-words* approach) is quite straightforward as it is easy to decompose a text into words by splitting a string into words by white space characters for the English language. Clearly, the most advantage of this bag-of-words approach is simplicity as it ignores the text logi-

cal structure and layout. That is, only the frequency of a word in a document is recorded, while all the structure and the ordering of the words are left out.

However, due to this simplicity characteristic, it has often been criticized for its disregard of semantic relationships between words which are thought to be crucial to human understanding. But, even though it omits context information, this representation performs well in practice. The possible reasons could be because words are the elements of the language where syntax and semantics meet and they are the basic syntactic building blocks that carry their own meaning. In addition, while more expressive representation can capture more of the meaning of the document, their increased complexity degrades the quality of statistical models based on them. Therefore, the bag-of-words approach appears to be a good compromise between expressiveness and model complexity.

3.3.3 Multi-Word Level

With the development of computational linguistic tools, large quantity of text can be analyzed efficiently with respect to their syntactic structure. Some researchers have used phrases, rather than individual words, as indexing terms (see [FHK+91], [SHP95], [TH93]). The most commonly used syntactic structures are noun phrases.

The notion of "phrase" usually incorporates syntactic and/or statistical information. From the syntactic aspect, the phrase is constructed according to a grammar of the language (see [Lew92]). From the statistical aspect, the phrase is composed of a set/sequence of words whose patterns of contiguous occurrence in the collection are statistically significant (see [CMS01]).

However, the experimental results found to date have not been uniformly encouraging. In [Lew92], Lewis argued that the likely reason for the discouraging results is that, although indexing languages based on phrases have superior semantic qualities, they have inferior statistical qualities with respect to the word-only indexing languages: a phrase-only indexing language has "more terms, more synonymous or nearly synonymous terms (since synonymous terms are not assigned to the same documents), lower consistency of assignment, and lower document frequency for terms". Although his remarks are about syntactically motivated phrases, they also apply to statistically motivated ones, although perhaps to a smaller degree.

A combination of the two approaches is probably the best way to go: [TH93] obtaining significant improvements by using noun phrases obtained through a combination of syntactic and statistical criteria, where a "crude" syntactic method was complemented by a statistical filter (only those syntactic phrases which occurred at least three times in the positive examples of a category c_i were retained). The final word on the usefulness of phrase indexing in text categorization is yet to be told, and investigations in this direction are still interesting for researchers to actively pursue in future.

In general, many of the existing techniques for extracting multi-word features are based on finding phrases and multiple words that occur near each other within the text. This is based on the assumption that the closer a set of intersecting terms, the more likely they are relevant; however, in [PA98], the authors modelled the multi-word features as a set of words appearing within windows of varying sizes and the experimental results suggested that windows of larger span yield improvements

in retrieval over windows of small span. The authors gave the possible explanation that the benefit of the larger window is due to their more frequent appearance in relevant documents versus non-relevant documents. However, the experiments were conducted based on a subset of TREC topics data set rather than the widely-used data collections for TC. Further research still needs to be done.

3.3.4 Semantic and Syntactic Representations

Like two sides of a coin, the simplicity characteristic of the bag-of-words approach has the advantage of easy text processing and the disadvantage of discarding a great deal of information from the original document. For example, the bag-of-words approach will respectively make a high distinction between the words *ocean* and *sea*, but will merge the different meanings of the word *surfing* (the Internet or in the sea) as it does not take into account the synonymic and polysemic properties of human languages.

To make the text representations closer to the format of human understanding language, researchers have resorted to other kinds of representations or techniques. For example, gathering together the words in "concepts" is meant to disambiguate the cases of synonymic or polysemic use of language. However, the experimental results found to date have not been consistently encouraging. The research work in this direction is still going on.

Word Senses

As the characteristics of natural language, the problems of synonym and polysemy are not evasive. The same word may assume different meanings in different contexts. For example, as a noun, "`bank`" has 10 senses; and as a verb, it has about 8 senses. On the other hand, different words may refer to the same or similar meaning. Therefore, it is reasonable to consider that word meanings provide more information about the content of a document and the category to which it belongs than words themselves.

To test this conjecture, in [KPKF03], the authors used *word meanings* (*senses*) to represent a document for TC task. However, a series of experiments indicated that this sense-based representation did not result in any significant categorization improvement on the word-based representation. The authors stated that the possible reasons may come from two aspects: (1) the data set is quite small with 123 training documents and 59 testing documents which are divided into 7 categories; (2) in a practical classification task the step of senses disambiguation would introduce a significant error. Though this result is not encouraging, it is still quite interesting to do more research on this work, for example, increasing the data set size and human intervention to improve the accuracy of word sense disambiguation.

Term Clustering

Term clustering tries to group words with a high degree of pairwise semantic relevance, so that the groups (or their centroid, or a representative of them) may be used instead of the terms as dimensions of the vector space. Term clustering is

different from term selection, since the former tends to address terms *synonymous* (or near-synonymous) with other terms, while the latter targets *noninformative* terms.

Lewis was the first to investigate the use of term clustering in TC. In [Lew92], besides studying the phrasal indexing, he also studied the properties of clustered indexing languages on a text categorization task. He conducted comparative experiments based on word-based and phrasal indexing languages, with and without term clustering. The experimental results showed that neither traditional term clustering method nor the syntactic phrasal representation provides significantly improved text representation.

Semantic and Syntactic Relationships

In [SM99], the authors examined some alternative ways to represent text based on syntactic and semantic relationships between words, i.e. phrases, synonyms and hypernyms (hyponyms). The goal of using phrases as features is to attempt to preserve some of the information left out from the bag-of-words representation. The authors tried two kinds of different phrase-based representations, that is, noun phrases extracted by a tool named Noun Phrase Extractor (NoPE) which uses syntactic information to select a useful subset of all the available phrases in the text, and key phrases developed by Peter Turney [Tur99] who uses a statistical algorithm to extract the most *meaningful* phrases from a document. Moreover, in order to capture the semantic relationships between words ignored by using the bag-of-words representation, the author also included a hypernym-based representations (a linguistic term for the "is a" relationship. For example, a knife is a weapon,

therefore, "weapon" is a hypernym of "knife".) by virtue of WordNet. However, the experimental results showed that none of the new representations performed significantly better than the bag-of-words representation.

Latent Semantic Indexing

Latent semantic indexing (LSI) [SDH90] is a feature reconstruction technique developed in IR in order to address the problems deriving from the use of synonymous, near-synonymous, and polysemous words as dimensions of document representations. This technique compresses document vectors into vectors of a lower-dimensional space whose dimensions are obtained as combinations of the original dimensions by looking at their patterns of co-occurrence. In practice, LSI infers the dependence among the original terms from a corpus and "wires" this dependence into the newly obtained, independent dimensions. The function mapping original vectors into new vectors is obtained by applying a singular value decomposition (SVD) to the matrix formed by the original document vectors. In TC, this technique is applied by deriving the mapping function from the training set and then applying it to training and test documents alike.

One characteristic of LSI is that the newly obtained dimensions are not, unlike in term selection and term clustering, intuitively interpretable. However, they work well in bringing out the "latent" semantic structure of the vocabulary used in the corpus. A drawback of LSI, is that if some original term is particularly good in itself at discriminating a category, that discrimination power may be lost in the new vector space.

Wiener et al. [WPW95] used LSI in two alternative ways: (i) for local LSI, thus

creating several category-specific LSI representations, and (ii) for global LSI, thus creating a single LSI representation for the entire category set. Their experiments showed that the former approach (local LSI) performs better than the latter (global LSI), and both approaches perform better than simple term selection based on feature selection scores, such as χ^2, *information gain* and so on. The similar results can also be found in the recent study [LCZ+04]. For other TC works that have used LSI techniques, see [SHP95], [Hul94], [Sch98], [WWP99].

Combination Approach

Even the indexing terms used to represent a text can be at word, sub-word and multi-word levels, these terms are always used exclusively. In many pattern recognition applications, the exploitation of different information sources for the same recognition task often leads to different errors in the results, which are very often complementary. Therefore, an appropriate exploitation of these sources can effectively reduce the error rate. [Joa02] also claimed that "a high level of redundancy is a desirable property of text-classification tasks". This inspired the authors in [DP03], who proposed an approach to combine two types of indexing terms, i.e. words and 3-grams, to represent a text for TC. The preliminary experimental results showed that in most cases, appropriate combination of different types of features can work better than a single type of feature. Although the data set tested in this study is only a partial corpus, and therefore it may somewhat reduce the reliability of this result, further study along this direction can be quite interesting.

3.3.5 Other Knowledge-based Text Representations

Other researchers try to make use of some available information beyond the original corpus such as manually-made summaries (only obtained from training data set) or present more complicated representation models to capture more information and relationships from the corpus for TC. The following are two examples:

Using Keywords from Summarization

All the above literature assumed that only the original text corpus is available before training the text classifier. However, the authors in [CLL03] proposed a new method for text classification when the summaries of the texts are available. Note that only the summaries of the training data set are available and are used to for transforming the text representation in training and test documents.

For each keyword (non-stopword) in summary, the authors picked out the top 20 words with which the keyword has closer relationship from the training data sets. Based on the Naïve Bayes model, the text is mapped into a vector of probability values, each of which corresponds to a keyword. Then the text classification is conducted on the mapped vector. Actually, the author converted the original term-based text space into a keyword-based text space because the keywords in summary may be more accurate and informative than the terms in original text. The experimental results on the email messages showed significantly improvements with respect to other methods.

Theme Topic Mixture Model – A Graphical Model

Theme Topic Mixture Model actually is a kind of graphical model presented in [KB04]. This density estimation model assumes two types of relations among textual data. Topics link words to each other and themes gather documents with particular distribution over the topics. The results showed that this model failed with respect to the simple bag-of-words representation when the size of the data set becomes sufficiently large. However, only when small proportion (say 1%) of data used for training the SVMs, this density estimation model captured important information from the data and outperformed the bag-of-words representation.

3.3.6 Remarks on the Term Types

Generally, the simple bag-of-words approach performs well in practice and the complicated high level representations have not shown good performance on TC in most cases. However, it is still too early to draw a definite conclusion that the bag-of-words approach is better than the complicated representations. The current obstacle that hinders further research comes from the small annotated data corpora. To solve this issue, either more human intervention is needed to preprocess the initial data corpora or more advanced NLP techniques are needed to improve the accuracy of the representations of the data corpora.

3.4 How to Weigh a Term?

No matter which term type we adopt to represent a document, not all terms are equally useful for the semantics of documents and thus an important indicator represents how much the term contributes to the semantics of document.

Salton [SB88] elaborated three considerations of the assignment of appropriately weighted terms for IR tasks. First, term occurrences in one document appear to closely represent the content of the document and can help to improve the recall measure. Second, term frequency alone may not have the discriminating power to pick up all the relevant documents from other irrelevant documents. Therefore, the *idf* factor has been proposed to help to improve the precision measure. In general, the two factors, *tf* and *idf*, are combined by a multiplication operation and are thought to improve both recall and precision measures. Third, to take the effect of length of documents into consideration, a *cosine* normalization factor is incorporated to equalize the length of the documents.

3.4.1 Term Frequency Factor

Table 3.1 lists four widely-used term frequency components, namely, a binary weight, a normal raw term frequency, a logarithm of term frequency and an inverse term frequency.

The simplest way of text representation is the *binary* representation, which ignores the occurrences of terms in the document (1 for present and 0 for absent). It is usually used in certain machine learning algorithms such as Naïve Bayes, decision

Table 3.1: Term frequency component

1.0	Binary weight equal to 1 for terms present in a vector
tf	Raw term frequency (number of times a term occurs in a document)
$log(1+tf)$	Logarithm of the term frequency
$1-\frac{r}{r+tf}$	Inverse term frequency (ITF), usually $r=1$

tree, where a floating number format of term frequency might not be used or not be used without constraints. The most popular term frequency representation just adopts the raw term frequency(tf) in the document. Moreover, different variants have been presented, such as $\log(1+tf)$ [BSAS94], where the logarithm operation is used to scale the effect of unfavorably high term frequency in one document. Inspired by the inverse document frequency, ITF (inverse term frequency) was proposed by [LK02]. These term frequency alone factors could be used as term weighting methods without other factors.

3.4.2 Collection Frequency Factor

In 1972, Karen Sparck Jones published a paper in *Journal of Documentation* [Jon72] where she proposed a term weighting scheme now known as inverse document frequency (idf). The original idea of idf is to pick up all the relevant documents from other irrelevant documents for IR tasks and it is typically computed as $log(N/n)$, where N is the number of documents in the whole collection and n is the number of documents which contain this term in the collection.

In the traditional *probabilistic* model, the relevance properties of the documents are taken into account and thus a *term relevance* weight is derived as the proportion of relevant documents in which a term occurs divided by the proportion of non-relevant documents in which the term occurs. Due to the lack of knowledge of the occurrences of the terms in the relevant and non-relevant documents in IR, this *term relevance* can be reduced to an inverse document frequency factor of the form $log((N-n)/n)$ (see [WS81]). We also call this variant factor as *idfprob* (which means *probabilistic idf* for it was derived from the *probabilistic* language model). However, in TC, this *term relevance* factor can be calculated from the training data set in advance and it is actually a well-known statistical measure, known as *Odds Ratio*.

Table 3.2 lists three collection frequency components from IR, namely, a multiplier of 1 that ignores the collection frequency factor, a conventional inverse collection frequency factor (idf), and a probabilistic inverse collection frequency ($idfprob$), respectively.

Table 3.2: Collection frequency component

1.0	No change in weight; use original term frequency component
$log(\frac{N}{n_i})$	Multiply original tf factor by an inverse collection frequency factor (N is the total number of documents in collection, and n_i is the number of documents to which a term is assigned)
$log(\frac{N-n_i}{n_i})$	Multiply tf factor by a *term relevance* weight (i.e. probabilistic inverse document frequency)

3.4.3 Normalization Factor

To eliminate the length of documents effect, we use the *cosine* normalization to limit the term weighting range within $(0, 1)$. Specifically, the *binary* feature representation does not use any normalization since the original value is 0 or 1. Assuming that w_{kj} represents the weight of term t_k in document d_j, the final normalized term weight w_{kj} might then be defined as

$$\text{normalized } w_{kj} \quad = \quad \frac{w_{kj}}{\sqrt{\sum_k (w_{kj}^2)}} \tag{3.1}$$

3.4.4 Traditional Term Weighting Methods from IR

In this thesis, we classify the term weighting methods into two categories according to whether the method makes use of the known category membership of training documents, namely, *supervised term weighting method* (if it uses this known membership information) and *unsupervised term weighting method* (if it does not use this information).

Collectively, the traditional term weighting methods rooted from IR belong to unsupervised term weighting methods as the calculation of these weighting methods do not involve the information on the membership of training documents in categories. For instance, *binary*, $tf.idf$ and its various variants. Here the tf factor also has various variants, such as $log(tf)$, $log(tf + 1)$, $log(tf) + 1$, ITF, etc. In addition, the idf (typically computed as $log(N/n)$) factor also has a number of variants, such as $log(N/n + 1)$, $log(N/n) + 1$, *idfprob*, etc. Besides these variants, there may be other variants which we will not cover in this thesis. Their formats are basically similar with each other and they share the same idea of $tf.idf$.

Besides, there are two special term weighting methods we have to discuss: RSJ and BM25.

RSJ (**R**obertson and **S**parck **J**ones) is also known as the term relevance weight in the *probabilistic* model presented by Robertson and Sparck Jones in 1976. First we define two probabilities for term t_k:

$$p_k = P(\text{document contains } t_k|\text{document is relevant}) \tag{3.2}$$

$$q_k = P(\text{document contains } t_k|\text{document is not relevant}) \tag{3.3}$$

Then the term weight is:

$$\text{RSJ weight } w_k = log\frac{p_k(1-q_k)}{(1-p_k)q_k} \tag{3.4}$$

Assume we know which documents are relevant and which are not, as before, a total of N documents of which n contain the terms, and further R out of the N are relevant and r relevant documents contain the term, then the obvious estimates are:

$$p_k \approx \frac{r}{R} \tag{3.5}$$

$$q_k \approx \frac{n-r}{N-R}. \tag{3.6}$$

The usual modification justified (not discussed here) is to use the following estimate of RSJ weight, where the 0.5 added to each of the components can be seen as a smoothing correction:

$$w_k = log\frac{(r+0.5)(N-R-n+r+0.5)}{(R-r+0.5)(n-r+0.5)} \tag{3.7}$$

However, the assumption above is of course not only unrealistic, but also makes a nonsense of the search process. Then if we have no relevance information, based

on some reasonable assumptions and simple estimates of equations (for details see [Rob04]), the final term weight is replaced by:

$$w_k = log\frac{N+0.5}{n+0.5} \tag{3.8}$$

and since $N >> 0.5$ and $n >> 0.5$, further simplified as

$$w_k = log\frac{N}{n} \tag{3.9}$$

It is now apparent that we can regard *idf* as a simple version of the RSJ weight, applicable when we have no relevance information.

The BM25 weighting function is based on an analysis of the behavior of the full eliteness model under different values of the parameters. Its formula can be expressed thus:

$$\text{BM25 weight } w_k = f(tf_k) * w_k^{(1)} \tag{3.10}$$

where $w_k^{(1)}$ is the usual RSJ weight, $f(tf_k) = \frac{(k_1+1)tf_k}{K+tf_k}$ (K and k_1 are global parameters which are in general unknown, but may be tuned on the basis of evaluation data; tf_k is the term frequency of term t_k). Even though the default parameter values may be adopted for these parameters in BM25, the best values will vary with the documents in the specific corpus.

It is clear that the formula given in (3.10) expresses BM25 as a $tf*idf$ weighting scheme. The first component is a term frequency (tf) component and the second component is the RSJ weight, which reduces to an *idf* measure in the absence of relevance information as we discussed just now.

Both of the two methods shown above have solid theoretical basis, when applied in practice, we must simplify them due to the unknown relevance information.

Finally, they all can be reduced to a version of $tf.idf$. For this reason, we treat these two weighting methods as variants of $tf.idf$. Hence, we still grant $tf.idf$ as the most popular term weighting method rather than other variants used in IR.

3.5 Supervised Term Weighting Methods

In contrast to IR, TC is a supervised learning task because the labels of training documents are available in advance. This known information on the category membership of training documents is quite helpful not only for the feature selection step but also for the construction of text classifier [YP97]. Recently, researchers have introduced several new term weighting methods by using this prior information. We group them as *supervised term weighting methods* once their calculation involves this known information in the training data set. Generally, the state-of-the-art supervised term weighting methods use this known information in the following ways.

3.5.1 Combined with information-theory functions or statistical metrics

One approach is to weigh terms by using feature selection metrics, such as χ^2, *information gain*, *gain ratio*, *odds ratio* and so on [YP97]. These information-theoretic functions usually have been applied to feature selection to reduce the high dimensionality of the term space by picking out the most relevant and discriminating features for the classification task. The terms with higher feature selection scores

are deemed to contribute more to the TC task than those with lower scores. Consequently, these scores are believed to be helpful in assigning more appropriate weights to the terms.

In [DTY+04], Deng et al replaced the *idf* factor with χ^2 factor to weigh terms and asserted that $tf.\chi^2$ is more effective than $tf.idf$ in their experiments with a SVM-based TC. Similarly, Debole [DS03] assigned weights to terms by replacing the *idf* factor with three metrics that have been widely used for feature selection process, namely, *information gain*, χ^2 and *gain ratio*. However, these supervised term weighting methods have not been shown to have a consistent superiority over the standard $tf.idf$-based term weighting. In most cases, the $tf.idf$ method is better than these complicated supervised term weighting approaches. Specifically, $tf.idf$ outperforms $tf.\chi^2$ in [DS03], which is contrary to the conclusion of [DTY+04]. With respect to the two studies, the only difference is that Deng et al [DTY+04] used different metrics for feature selection and subsequent term weighting operation while Debole et al [DS03] adopted the same metric for the two steps, i.e. feature selection and term weighting.

Beside the TC task, the idea of using feature selection metrics to weigh terms has been adopted in other text mining tasks. For example, in document summarization, Mori [Mor02] adopted *gain ratio* as a term weighting method to compare with a *tf*-based summarization system and the result showed that the *gr*-based term weighting method is very effective in summarization.

3.5.2 Based on Statistical Confidence Intervals

In [SM05], the authors introduced a new term weighting method called `ConfWeight` based on statistical confidence intervals. As before, let N be the number of documents in the whole collection and n is the number of documents which contain the term in this collection. They estimate the proportion of documents containing term t_k to be:

$$\tilde{p} = \frac{n + 0.5z^2_{\alpha/2}}{N + z^2_{\alpha/2}} \tag{3.11}$$

where $\tilde{p}$ is the Wilson proportion estimate and $z^2_{\alpha/2}$ is a value such $\Phi(z_{\alpha/2}) = \alpha/2$, Φ is the t-distribution (Student's law) function when $N < 30$ and the normal distribution one when $N \geq 30$. So when $N \geq 30^2$, $\tilde{p}$ is:

$$\tilde{p} = \frac{n + 1.96}{N + 3.84} \tag{3.12}$$

Thus, its confidence interval at 95% is:

$$\tilde{p} \pm 1.96\sqrt{\frac{\tilde{p}(1-\tilde{p})}{N + 3.84}} \tag{3.13}$$

For a given category, we get $\tilde{p_+}$ and $\tilde{p_-}$ by applying equation (3.12) to the positive and negative category in the training data set respectively. Then, we label *MinPos* as the lower range of the confidence interval of $\tilde{p_+}$ and label *MaxNeg* as the higher range of that $\tilde{p_-}$ according to formula (3.13). Let *MinPosRelFreq* be:

$$MinPosRelFreq = \frac{MinPos}{MinPos + MaxNeg} \tag{3.14}$$

Now the strength of term t_k for category c_i is defined as:

$$str_{t_k,c_i} = \begin{cases} log2MinPosRelFreq & \text{if } MinPos > MaxNeg \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

Then by using a global policy technique, the weight is defined as the maximum strength of t_k:

$$maxstr(t_k) = \max_{c \in c_i}(str^2_{t_k,c}) \tag{3.16}$$

Thus, the final ConfWeight of term t_k in document d_j is defined as:

$$ConfWeight_{t_k,d_j} = log(tf + 1) * maxstr(t_k) \tag{3.17}$$

The first component in formula 3.17 is actually a logarithmic operation of term frequency. The second component of `ConfWeight` determines the weight of a particular term. Thus, although the authors assigned weight to terms based on statistical confidence, to calculate the `ConfWeight` value for each term with respect to each category c_i, the prior knowledge of labelled training data set is required.

The experimental results showed that *ConfWeight* generally outperformed $tf.idf$ and *gain ratio* on three benchmark data collections. However, their experiments failed to show that supervised weighting methods are generally superior to the unsupervised ones.

3.5.3 Interaction with Linear Text Classifier

This approach is to weigh terms in the interaction with a text classifier. Similar to the idea of using feature selection functions as term weighting factors, the text classifier selects the positive documents from negative documents by assigning different scores to the documents and thus these scores are believed to be effective in weighting terms for TC.

In [HKK01], terms are weighted using an iterative approach involving the kNN text classifier at each step. For each iteration, the weights are slightly modified and the categorization accuracy is measured using an evaluation set (a split from the training set). Convergence of weights should provide an optimal set of weights. However, this method is generally much too slow to be used, particularly for broad problems (involving a large vocabulary).

Strictly speaking, another literature combining the term weighting methods with supervised linear learning algorithms is used for feature selection tasks rather than for term weighting methods. But due to the potential usage in term weighting methods, we include it in this literature review as a supervised term weighting method. In [MBGMF04], Mladenic compared three term weighting methods, i.e. *Odds Ratio*, *information gain* and weights from linear models (linear SVM and Perceptron). Their results showed that feature scoring using weights from linear SVMs yields better classification performance than other term weighting methods when combined with three algorithms, namely, Naïve Bayes, Perception and SVM. In addition, they postulated that it is the sophistication of the term weighting method rather than its apparent compatibility with the learning algorithm that improves classification performance.

3.6 Analysis of Term's Discriminating Power

Whether the traditional unsupervised or the state-of-the-art supervised term weighting methods, the goal is to assign appropriate weights to terms to improve the terms' discriminating power for TC. In this section, we give an analysis of the terms' discriminating power for TC task.

For multi-label classification problem, the benchmark on each corpus is generally simplified into multiple independent binary classification problems. That is, in each experiment, a chosen category is tagged as the positive category and the other categories in the same corpus are combined together as the negative category. Therefore all collection frequency factors are specified "locally" to a specific positive category c_i. We adopt this local policy to assign term weights in this thesis.

Term frequency represents a close relationship between the term and the content of the documents which contain this term. However, it is observed that if high frequency terms are not concentrated in a few particular documents, we may not retrieve the relevant documents from the whole collection. Consequently, the traditional *idf* factor and its variants are introduced to improve the discriminating power of terms in the traditional IR field. However, in the field of TC, it may not be the case. To illustrate the difference between them, we take Figure 3.2 for example to analyze the terms' discriminating power to TC. Figure 3.2 depicts the distributions of documents which contain six terms, $t1, t2, t3, t4, t5$ and $t6$, given one chosen positive category on one data collection.

In this figure, each column represents the documents distribution in the corpus for each term. The height of one column is the number of documents in the corpus. The horizontal line divides these documents into two categories, the positive (above) and the negative (below). The heights of the columns above and below the horizontal line denote the number of documents in the positive and negative categories respectively. The height of the shaded part is the number of documents which contain this term. We use a, b, c and d to denote the different numbers of

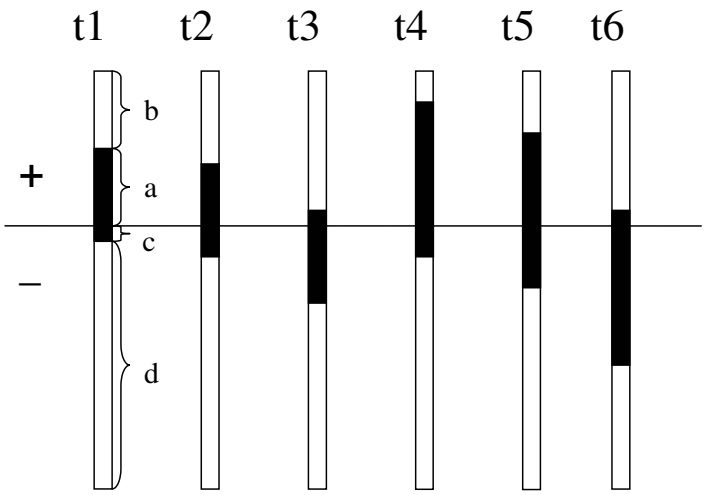

Figure 3.2: Examples of document distributions with respect to six terms in the whole corpus

documents as below:

- a : the number of documents in the positive category which contain this term
- b : the number of documents in the positive category which do not contain this term
- c : the number of documents in the negative category which contain this term
- d : the number of documents in the negative category which do not contain this term

Thus, N, the number of documents in the whole collection, is the sum of a, b, c and d. In general, $d >> a, b, c$.

By using the notations in Figure 3.2, several widely-used collection frequency factors and information theory functions are represented as follows:

$$idf = log\frac{N}{a+c} \tag{3.18}$$

$$idfprob = log\frac{b+d}{a+c} \tag{3.19}$$

$$\chi^2 = N * \frac{(a*d-b*c)^2}{(a+c)(b+d)(a+b)(c+d)} \tag{3.20}$$

$$\begin{aligned} ig &= \sum_{c\in c_i,\bar{c}_i}\sum_{t\in t_k,\bar{t}_k} P(t,c) * log\frac{P(t,c)}{P(t)*P(c)} \\ &= \frac{a}{N} * log\frac{a*N}{(a+c)*(a+b)} + \\ &\quad \frac{b}{N} * log\frac{b*N}{(b+d)*(a+b)} + \\ &\quad \frac{c}{N} * log\frac{c*N}{(a+c)*(c+d)} + \\ &\quad \frac{d}{N} * log\frac{d*N}{(b+d)*(c+d)} \end{aligned} \tag{3.21}$$

$$gr = \frac{ig}{-\frac{(a+b)}{N} * \log\frac{(a+b)}{N} - \frac{(c+d)}{N} * \log\frac{(c+d)}{N}} \tag{3.22}$$

$$\begin{aligned} or &= \frac{P(t_k|c_i)*(1-P(t_k|\bar{c}_i))}{(1-P(t_k|c_i))*P(t_k|\bar{c}_i)} \\ &= \frac{a*d}{b*c} \end{aligned} \tag{3.23}$$

We assume the six terms have the same term frequency (tf) and $N = 1000$. The first three terms, i.e. $t1$, $t2$ and $t3$, have the same idf, which we will call $idf1$, and the last three terms, i.e. $t4$, $t5$ and $t6$, share the same idf, which we will call $idf2$. It is clear to find that $idf2 < idf1$ as the traditional idf factor gives more weights to the first three terms with less document frequency in the corpus than the last three terms. Let us consider the first three terms, i.e. $t1$, $t2$ and $t3$. They share the same $idf1$ value but have different ratio of a and c illustrated in Table 3.3.

Table 3.3: The first three terms which share the same *idf* but have different ratio of a and c

Term	Sum(a,c)	a : c	*idf*
$t1$	100	10 : 1	log(N/100) = 3.322
$t2$	100	1 : 1	log(N/100) = 3.322
$t3$	100	1 : 10	log(N/100) = 3.322

The membership of a document in a category is pretty much as the relevance of a document to an information need in IR [Sar75]. In IR, the relevance information cannot be available in advance (otherwise it makes a nonsense of the search process), while in TC, the labelled training data set is available in advance. Thus, considering the examples in Figure 3.2, we cannot know where this horizontal line is in IR. However in TC, due to the availability of the training data set, we can draw out this line. This difference between IR and TC comes from the availability of training data sets.

It is quite clearly observed that these six terms contribute differently to the classification task. Intuitively, it is considered that the more concentrated the terms are in the positive category than in the negative category, the more contribution they may make to separate the positive samples from the negative examples. Thus, in the above examples in Figure 3.2, term $t1$ contributes more than $t2$ and $t3$ to distinguish the positive from the other negative categories. Similarly, term $t4$ contributes more than term $t5$ and $t6$.

However, by using *idf* and *idfprob* (listed in Equation 3.19) factors, we cannot see any difference among the first three terms. The similar observation can be found among the last three terms. Let us discuss the following two cases in which

the *idf* and *idfprob* factors fail to show their discriminating power in TC.

case 1: Consider the first three terms, i.e. $t1$, $t2$ and $t3$. We can easily find that these three terms make different contributions to TC. Specifically, $t1$ contributes more power to discriminate the positive documents from the negative documents than $t2$ and $t3$. Thus, $t1$ should be weighted more than $t2$ and $t3$. However, by using *idf* or *idfprob* factor, the three terms will equalize their discriminating power for text categorization. The same observation can be found for the last three terms. Therefore, in this case, the *idf* and *idfprob* factors fail to assign appropriate weights to terms based on their different contributions to text classification tasks.

case 2: Consider the two terms $t1$ and $t4$. Note the two have different *idf* values and $idf2\ (t4) < idf1\ (t1)$. However, it is clearly observed from this figure that $t4$ contributes more than $t1$ and thus $t4$ should be weighted more than $t1$. Thus, in this case, the *idf* and *idfprob* factors has a reverse effect on expressing the terms' discriminating power for text categorization.

In the above two cases, the *idf* and *idfprob* factors have no or even decrease the discriminating power for terms in TC. An obvious explanation for the failure of *idf* and *idfprob*in TC is that they do not consider the document distribution in the positive and negative category.

Consequently, the other supervised factors listed from Equations 3.20 to 3.23, i.e. χ^2, *Odds Ratio*, *information gain* and *gain ratio*, should, in theory, outperform *idf* and *idfprob* as they make use of this prior information from the training data set. The basic idea of using these functions is that the best terms for c_i are

the ones distributed most differently in the positive and negative examples of c_i. However, these functions may have different interpretations of this principle. For instance, in the experimental sciences χ^2 is used to measure how the results of an observation differ (i.e. are independent) from the results expected according to an initial hypothesis (here, the lower values means lower dependence). In TC, it is used to measure how independent term t_k and category c_i are. The term t_k with the lowest value for $\chi^2(t_k, c_i)$ are thus the most independent from c_i; since we are interested in the terms which are not, the terms for which $\chi^2(t_k, c_i)$ is highest are considered to contribute the most for classification task.

To discuss the differences in terms' discriminating power between the above supervised factors and *idf*, we take the two factors χ^2 and *or* for example in the following cases.

case 3: Consider the first three terms in Figure 3.2 again. The three are assigned differently in terms of *idf*, χ^2 and *or* factors. Specifically, in *idf* value, $t1 = t2 = t3$; in χ^2 value, $t1 = t3 > t2$; in *or* value, $t1 > t2 > t3$. This observation indicates that the three show inconsistent discriminating power with respect to each other. Moreover, even though χ^2 and *or* both consider the document distribution in the positive and negative category, the two assign different weights for the same term. This difference arises from their different theoretical bases or rationales.

case 4: Consider the two terms $t1$ and $t4$. Although the *idf* factor unreasonably results in $t1 > t4$, χ^2 and *or* correct this obvious mistake and appropriately makes $t4 > t1$. This case shows that χ^2 and *or* would assign more appropriate weights to terms than *idf*.

case 5: Consider the two terms $t2$ and $t5$. Let $a = c$ for each term (i.e. the same number of documents which contain this term in the positive and negative category). Clearly, $t2 > t5$ in *idf* value while $t5 > t2$ in both χ^2 and *or* values. This is inconsistent with our intuitive consideration that $t2$ and $t5$ should contribute equally because each term occurs with the same frequency in the positive category as that in the negative and thus they both have the same ratio of a and c. For details about our intuitive consideration, refer to the next Section 3.7.

Apparently, these supervised factors might assign more reasonable weights to terms than *idf* and *idfprob* as they consider the distribution of the documents in the positive and negative category from the training data set. However, we must note that in this section this is only a a qualitative analysis of terms' discriminating power. These supervised factors might show various degrees of discriminating power when weighting terms in the real world cases. To illustrate their different discriminating power from quantitative analysis aspect, we will take some real case examples in Section 3.8.

3.7 A New Proposed Supervised Term Weighting Scheme — *RF*

The basic idea of our intuitive consideration is quite simple: the more concentrated a high frequency term is in the positive category than in the negative category, the more contributions it makes in selecting the positive samples from among the negative samples.

Although the above supervised term weighting factors take the document distribution into account, they are not always consistent with the above intuitive consideration. Specifically, several supervised term weighting factors discussed in the previous subsection are symmetry in terms of positive and negative categories. That is, given one category c_i, if the distributions of term $t1$ in the positive and negative categories are the same as those of term $t3$ in the negative and positive categories respectively, they will be weighted equally in χ^2, *information gain* and *gain ratio* value. But, as we mentioned in the previous subsection, our intuitive consideration is that $t1$ contributes more than $t3$ and thus $t1$ should be weighted more than $t3$. Why we would like to assign more weights to $t1$ rather than $t3$? For the multilabel multiclass TC task, when we build a text classifier for each category, we usually label this category as positive category and group all the other categories as negative category. Thus, the documents in the positive category concentrate on one topic or several topics close to each other while the documents in the negative category are spread over a wide range of topics since all non-positive categories are grouped together as negative. Thus, the high frequency terms concentrated in the positive category are good discriminators to select the positive samples from among the various negative samples. This idea of favoring positive terms can also be observed in [NGL97] which tried to select such "positive" words via a metric (the square root of chi-square) as opposed to "negative" words indicative of negative documents. Therefore, unlike the above three supervised factors, we would like to assign more weight to term $t1$ than $t3$.

On first sight, the *or* factor seems to be consistent with our consideration. But we may recall *case 5* listed in the previous section where two terms $t2$ and $t5$ have the same ratio of a and c. In our consideration, they would have the

same contribution to TC no matter what value of b and d. That means, slightly increasing or decreasing the value of b and d (i.e. whether adding or deleting the documents in the positive or negative category which do not contain this term) should not have an impact of terms' discriminating power. However, it is not the case in *or* value. The difference between *or* and our consideration is that we consider the terms' discriminating power to be imposed by the number of relevant documents which contain this term only, i.e. a and c. Since $d >> a, b, c$ in general case, the *or* value would vary drastically because it involves the d value.

Based on these above intuitive considerations and the analysis in the previous section, we propose a new supervised weight factor rf (*relevance frequency*) to capture this basic idea. Its formula is expressed as:

$$rf = log(2 + \frac{a}{c}) \tag{3.24}$$

When combined with tf by a multiplication operation, the weight of term t_k is defined as:

$$tf.rf = tf * log(2 + \frac{a}{c}) \tag{3.25}$$

We assign the constant value 2 in the rf formula because the base of this logarithmic operation is 2. Without the constant 2+, the formula (3.24) would have the effect of giving a number of other terms zero weight. Let us consider two extreme cases in formula (3.25). In one extreme case, if $a = 0$ and then $rf = 1$, the final weight of a term in a document, i.e. $tf.rf$, at least equals to term frequency alone. In another extreme case, if $c = 0$, to avoid zero divisor, we set the minimal denominator as 1, which is reasonable whatever the value of a is. Thus the rf

formula is replaced by:

$$rf = log(2 + \frac{a}{\_max(1, c)}) \tag{3.26}$$

and, the final term weight is replaced by:

$$tf.rf = tf * log(2 + \frac{a}{\_max(1, c)}) \tag{3.27}$$

Compared with the other supervised factors, the rf factor does not involve the d value. This is based on the observation that $d >> a, b, c$ and thus d will lessen the significance of a and c in expressing the term's discriminating power for TC.

We name it rf (**r**elevance **f**requency) because only the frequency of relevant documents (i.e. those which contain this term) are considered in this formula. That is, only the ratio of a and c exerts an influence on the formula. Table 3.4 lists the rf values with different a and c values.

Table 3.4: The rf values with different a and c values

rf		**a**						
		0	1	2	3	4	5	6
c	(0,1)	1.00	1.58	2.00	2.32	2.58	2.81	3.00
	2	1.00	1.32	1.58	1.81	2.00	2.17	2.32
	3	1.00	1.22	1.42	1.58	1.74	1.87	2.00
	4	1.00	1.17	1.32	1.46	1.58	1.70	1.81

Clearly, rf function captures the idea that the best discriminators for c_i are the ones distributed more in the positive examples of c_i than in the negative ones.

Therefore, by using rf, we weight $t1$ more than $t2$ and $t3$ in the above examples. Similarly, $t4$ is weighted more than $t5$ and $t6$ by using rf factor.

3.8 Empirical Observation of Term's Discriminating Power

The previous two sections only presented an analytical explanation of the six widely-used term weighting factors (Equations 3.18 to 3.23) and our newly proposed rf factor. To evaluate their discriminating power from a quantitative aspect, we apply them to real world cases. To accomplish it, we choose four terms from the Reuters News Corpus. Table 3.5 and 3.6 list the six different factors' values for the selected four terms in terms of two categories, namely, *00_acq* and *03_earn* respectively.

Table 3.5: Comparison of six weighting values of four features in category *00_acq*

Feature	Category: *00_acq*					
	idf	rf	χ^2	*or*	*ig*	*gr*
acquir	3.553	4.368	850.66	30.668	0.125	0.161
stake	4.201	2.975	303.94	24.427	0.074	0.096
payout	4.999	1.000	10.87	0.014	0.011	0.014
dividend	3.567	1.033	46.63	0.142	0.017	0.022

Based on the literal meaning, the first two terms, *acquir*[1] and *stake*, are closely related to the content of category *00_acq* while the last two terms, *payout* and *dividend*, are closely related to the content of category *03_earn*. However, as the

[1]Here the *acquir* is a common root of several morphological forms of words and the original words can be *acquire*, *acquired*, *acquires*, *acquiring*, *acquirement*, etc.

Table 3.6: Comparison of six weighting values of four features in category *03_earn*

Feature	Category: *03_earn*					
	idf	rf	χ^2	*or*	*ig*	*gr*
acquir	3.553	1.074	81.50	0.139	0.031	0.032
stake	4.201	1.082	31.26	0.164	0.018	0.018
payout	4.999	7.820	44.68	364.327	0.041	0.042
dividend	3.567	4.408	295.46	35.841	0.092	0.095

idf factor neglects the category information of the training set, each of these four terms is weighted equally by the *idf* factor in terms of the two categories. On the contrary, the supervised term weighting factors assign different weights to different terms for different categories. It is observed from the two tables that these supervised factors might show various degrees of discriminating power in weighting terms.

Let us recall Salton's consideration in Section 3.4 regarding the two factors composing a term's weight in a document, i.e. term frequency factor and collection frequency factor. Term frequency factor is considered to appropriately express the content of the document and also indicates how close this term is to the semantics of this document. On the other hand, the various collection frequency factors are used to impose discriminating power on terms to separate the positive samples from the negative samples. They both are considered to make a contribute for text classification. But, which factor is more important?

Our consideration is that both factors are equally important in contributing to text classification and their good combination has a crucial effect on expressing the discriminating power of a term in a document. Thus, de-emphasizing or

emphasizing either factor might result in inappropriate weights to terms. Usually, although the frequency of a term in a document varies according to the natural property of documents, it is quite a common case that a term occurs in zero, or several times rather than tens or even hundreds of times in a document.

Let us continue to discuss the supervised factors' different efforts to assign the terms' discriminating power.

Consider the χ^2 factor first. The χ^2 value of these four terms are quite large. Take term *acquir* for example, its χ^2 value is 850.66 and 81.50 in the two categories respectively. Since this value is significantly much larger than the common value of tf, it results in suppressing the impact of the term frequency factor in contributing the discriminating power to this term. The similar observation can be found for the *or* factor. Moreover, the values of the *or* factor for the four terms vary more drastically than the χ^2 factor and thus the *or* factor may suppress the term frequency factor's power even more.

On the contrary, the effects of the ig and gr factors are significantly outweighed by the term frequency factor.

Finally, it is interesting to find that the rf factor has a comparable impact to term frequency factor for each of the four terms. Thus the $tf.rf$ method will be able to balance the impact of both the term frequency factor and collection frequency factor on expressing their discriminating power.

So far, only the analytical explanation and the empirical observation based on several selected samples are given. To experimentally compare the performance

of these factors and confirm the effectiveness of our intuitive consideration of the $tf.rf$ method, we will conduct a series of experiments under various circumstances. That is what Chapter 5 will do. But before we start by conducting experiments to empirically address these problems, we will first establish the experimental settings for this thesis and describe them in next chapter.

CHAPTER 4
METHODOLOGY OF RESEARCH

This chapter establishes the experimental methodology of this thesis. Section 4.1 describes the learning algorithms applied in this thesis. Section 4.2 presents the benchmark data collections on which the experiments have been conducted. Section 4.3 introduces evaluation metrics to measure the performance of different methods. Section 4.4 describes the statistical significance tests.

4.1 Machine Learning Algorithms Applied in This Thesis

SVM and kNN achieve top-notch performance among the widely-used machine learning algorithms (see [YL99], [Joa98] and [DPHS98]). That is the main reason we choose them. Another reason is that although other algorithms such as Decision Tree and Naïve Bayes are also widely used, they are not included because the real

numbers format of term weights could not be used except for the *binary* text representation. Even the real numbers could be used with various constraints[1] ([MN98]), other learning algorithms have been shown worse performance than the two promising algorithms. Finally, the two state-of-the-art algorithms scale to large classification problems with thousands of features and examples.

4.1.1 Support Vector Machines

Due to their ability to efficiently handle relatively high dimensional and large-scale data sets without decreasing classification accuracy, SVMs have been confirmed to show better performances than other methods by many researchers [DPHS98] [Joa98] [LK02].

According to different kernel functions from computational learning theory, SVMs are classified into two categories, namely, linear and non-linear. Specifically, our benchmark adopted the linear SVM rather than non-linear SVM. The reasons why we chose linear kernel function of SVM in our experiments are listed as follows. First, linear SVM is simple and fast [DPHS98]. Second, our preliminary experimental results showed that the linear SVM performs better than the non-linear models, even at the preliminary optimal tuning level the accuracy achieved with RBF kernel is lower than that of linear (0.8 vs 0.9). This result contradicts our anticipation of a better performance by a more sophisticated kernel when dealing with numerous dimensional features but it is also consistent with the findings in [DPHS98] and [YL99]. Third, this result might be considered preliminary, but

[1]For example, [MN98] presented a model for Naïve Bayes text classification which can relax the constraint that document vectors should be binary-valued.

our current focus is on the comparison of term weighting schemes rather than how to tune the parameters of kernel functions. Following the established practice in TC, throughout this thesis we used an SVM with a linear kernel as the benchmark classifier algorithm. The SVM software we used is LIBSVM-2.6 and its later updated version, LIBSVM-2.8 [CL01].

4.1.2 k Nearest Neighbors

The kNN algorithm is very simple and effective. However, the most important drawback is its inefficiency in the case of high dimensional and large-scale data sets. This drawback comes from the nature of this "lazy" learning algorithm as it actually does not have a true learning phase and thus incur a high computational cost at the classification time.

The performance of kNN is dependent on the unique parameter k. That is, once the k value is fixed, the performance of kNN is also unchangeable. In order to estimate the unique parameter k for kNN, we need a validation data set which is different from the training data set. Usually, a subset of training data set is separated to tune this parameter and the remaining training data set is used to train the model of text classifier. [LC96] used $k = 20$. The previous work in [YL99] set k as $30 - 45$ since they found that in that range the kNN yields stable effectiveness. Similarly, [Joa98] also achieved the best performance for kNN when $30 \leq k \leq 45$. Following the above two attempts, we explored the k values where $k \in \{1, 15, 30, 45\}$ for the kNN classifier. The results for the parameter k with the best performance on the test set are reported in the results sections in the next chapter.

4.2 Benchmark Data Collections

To make the comparisons between our results and the published results meaningful, we conduct experiments on standard widely-used benchmark data collections. We first discuss the general text preprocessing steps for these corpora.

4.2.1 Text Preprocessing

The stop word list we used consists of 292 or 513 functional or connective words that assumed to have no information content. Appendix I lists the 513 stop words. After removing stop words and punctuation characters, the Porter's stemming was performed to reduce words to their base forms according to [Por80].

We use χ^2 metric for feature selection in this thesis. [YP97] found χ^2 and *information gain* are the most effective feature selection metrics in their experiments. We have thus followed her study to use χ^2 for feature selection. However, since SVM has the capability to deal with high dimensional features, and the previous works showed that feature selection does not improve or even slightly degrades the SVM performance [LK02] and [DL04], we also conducted experiments by inputting the full words (after remove stop words, stemming and set minimal term length[2] as 4) without feature selection.

[2] i.e. the term with only 3 or less letters will be removed.

4.2.2 Reuters News Corpus

The most widely used corpus is the Reuters corpus, consisting of a set of news stories classified under categories related to economics. The Reuters collection accounts for most of the experimental work in text categorization so far. Thus, conducting experiments on this popular corpus have meaningful comparison with the existing works.

In this study, we selected the documents from the top ten largest categories of the Reuters-21578 document collection. We adopted the bag-of-words approach for the documents. According to the widely-used ModApte split (it is now a standard split in Reuters-21578 corpus), the 9,980 news stories have been partitioned into a training set of 7,193 documents and a test set of 2,787 documents. Stop words (292 stop words), punctuation and numbers were removed. The Porter's stemming was performed to reduce words to their base forms [Por80]. The threshold of the minimal term length is 4 (i.e. each term has at least 4 letters.). Null vectors (i.e. vectors with all attributes valued 0) were removed from the data set. The resulting vocabulary has 15937 words (terms or features).

By using the χ^2 statistics ranking metric for feature selection, the top p features per category were selected from the training sets. In our experiments, we set p ={25, 50, 75, 150, 300, 600, 900, 1200, 1800, 2400, All} respectively.

One noticeable issue of Reuters corpus is the skewed category distribution problem. Among the top ten categories which have 7193 training documents, the most common category has a training set frequency of 2877 (40%), but 80% of the categories have less than 7.5% instances.

4.2.3 20 Newsgroups Corpus

Another benchmark data corpus is the 20 Newsgroups corpus[3], which is a collection of approximate 20,000 newsgroup documents nearly evenly divided among 20 discussion groups and each document is labelled as one of the 20 categories corresponding to the name of the newsgroup that the document was posted to. Some newsgroups are very closely related to each other. For example, the posts in category of *comp.sys.ibm.pc.hardware* are very similar to those in category of *comp.sys.mac.hardware*. However, others are highly unrelated, for example, the category of *misc.forsale* and category of *soc.religion.christian*. After removing duplicates and headers, the remaining 18846 documents are sorted by date and are partitioned into 11314 training documents (about 60%) and 7532 test documents (about 40%). After this partition, the training and test documents still remain nearly evenly distribution on the 20 topics. Therefore, compared with the skewed category distribution in the Reuters corpus, the 20 categories in the 20 Newsgroups corpus are of approximately uniform distribution.

The resulting vocabulary, after removing stop words (513 stop words) and words that occur less than 3 and 6 times in the positive and negative categories respectively, has 50088 words. According to the χ^2 statistics metric, the top $p \in \{5, 25, 50, 75, 100, 150, 200, 250, 300, 400, 500\}$ features are selected.

[3]The 20 Newsgroups corpus can be freely downloaded from http://people.csail.mit.edu/jrennie/20Newsgroups/.

4.2.4 Ohsumed Corpus

The MEDLINE database is the largest component of PubMed (http://pubmed.gov). The Ohsumed collection is a subset of clinically-oriented MEDLINE from year 1987 to year 1991, consisting of 348,566 references out of a total of over 7 million, covering all references from 270 medical journals over a five-year period. The Ohsumed in year 1991 includes 74337 documents but only 50216 of which having abstracts. Joachims [Joa98] used the first 10000 documents for training and the second 10000 documents for testing. This data set is available at ftp://medir.ohsu.edu/pub/ohsumed. Hence, we can clearly find the relationship among these databases is as follows:

$$\text{Joachims' Ohsumed} \subset \text{OHSUMED} \subset \text{MEDLINE} \subset \text{PubMed}$$

Specifically, we use the Ohsumed corpus adopted by Joachims in [Joa98] because this data set is a widely-used benchmark data collection and the comparison between our experiments and other published experiments makes sense.

This corpus including medical abstracts from the MeSH categories are related to 23 cardiovascular diseases and each disease corresponds to one category label. After selecting such category subset, the unique abstract number becomes 13,929 (6,286 for training and 7,643 for testing). The resulting vocabulary, after removing stop words (513 stop words) and words that occur less than 3 and 6 times in the positive and negative categories respectively, has 19501 words. According to the χ^2 metric, the top $p \in \{10, 25, 50, 75, 100, 200, 300, 400, 600, 800, 1000, 1200\}$ features are tried.

4.2.5 18 Journals Corpus

From the digital library center in National University of Singapore (NUS), we chose 20 journals with the top largest impact factor in the subject categories of `Biochemistry and Molecular Biology`. Due to access limitation, two journals are not accessible[4]. The resulting data collection named 18 Journals Corpus consists of $7,903$ documents from a latest two-year span (year 2004 - year 2005) of 18 journals in PubMed. Table 4.1 lists the statistical information of these 18 journals.

After deleting the duplicates and removing the documents with blank abstract and/or blank MeSH keywords[5], the ultimate corpus has $5,417$ documents and 933 MeSH keywords. Each of the 933 MeSH keywords is viewed as a category label and a document belongs to a category if it is indexed with at least one such keyword from these 933 keywords.

Due to a large number of category labels (i.e. keywords), most categories contain only 1-2 documents. Then we select 8421 document and category pairs from the top 132 categories each of which has at least 10 articles. For each category, the first half of documents are used to train the text classifier model and the last half of documents are used to test as unlabelled samples. After removing the 513 stop words, the whole vocabulary is 19018 words in all 132 categories. For this corpus, we also conducted experiments on three subsets with different sizes as shown in Table 4.2

[4]The two inaccessible journals are "Biochimica ET Biophysica ACTA-Reviews On Cancer" and "Reviews Of Physiology Biochemistry And Pharmacology".

[5]We also remove the articles which keywords are *Research* or *Research Support by Government* because these keywords are too general and not biomedical keywords.

Table 4.1: Statistical information of the 18 Journals Corpus

Abbreviated Journal Title	Impact Factor	# of Articles
Annual review of biochemistry	31.538	78
Nature medicine	31.223	773
CELL	28.389	844
Molecular cell	16.811	688
Trends in biochemical sciences	14.112	229
PLoS biology	13.868	525
Annual review of biophysics & biomolecular structure	13.462	40
Nature structural & molecular biology	12.000	491
Current biology : CB	11.901	1396
The Plant cell	11.295	571
The EMBO journal	10.492	915
Genome research	10.382	501
Cytokine & growth factor reviews	9.926	98
Current opinion in structural biology	9.821	184
Progress in lipid research	8.810	39
Advances in microbial physiology	8.667	11
Current opinion in chemical biology	8.623	183
Cell death and differentiation	8.192	337

Table 4.2: Statistical information of three subsets of the 18 Journals corpus

Top Categories	# of Document-Category Pairs	# of Feature Set Size
10	4974	15103
50	7060	17894
132	8241	19018

Generally, compared with the Ohsumed corpus which involved domain experts in grouping the documents into 23 categories regarding cardiovascular diseases, the 18 Journals corpus does not require domain experts and all documents are grouped based on the indexed MeSH keywords. In some sense, the 18 Journals corpus is more difficult than the Ohsumed corpus because the data are more "noisy". That is, the word/category correspondences are more "fuzzy" in the 18 Journals corpus. Consequently, the categorization is more difficult to learn for a classifier. It will be a harder work for the 18 Journals corpus than the Ohsumed corpus.

4.3 Evaluation Methodology

With respect to TC systems, the experimental evaluation of classifier usually measures its *effectiveness* rather than its efficiency, that is, its ability to take the *right* classification decisions.

4.3.1 Precision and Recall

Usually, classification effectiveness is measured by using *precision* and *recall*. *Precision* (p) is the proportion of truly positive examples labelled positive by the system that were truly positive and *recall* (r) is the proportion of truly positive examples that were labelled positive by the system.

For obtaining estimates of *precision* and *recall*, two different methods may be adopted:

micro-averaged: where the *precision* and *recall* are obtained by summing over all individual decisions;

macro-averaged: where *precision* and *recall* are first evaluated "locally" for each category, and then "globally" by averaging over the results of the different categories;

These two methods may give quite different results, especially if the different categories have very different generality. That is, the ability of a classifier to behave well also on categories with low generality (i.e. categories with few positive training instances) will be emphasized by macro-averaged and much less so by micro-averaged. Thus it is clear to be observed that the two methods will be equalized on the uniform category distribution data sets. Whether one or the other should be used obviously depends on the application requirements.

Neither *precision* nor *recall* makes sense in isolation from each other as it is well known from the IR practice that higher levels of *precision* may be obtained at the price of low values of *recall*. Thus, a classifier should thus be evaluated by means of a measure which combines *precision* and *recall*. To accomplish this, several measures have been proposed. Among them, the two most widely-used measures adopted by TC are, F_1 function and *breakeven point*.

4.3.2 F_1 Function

A F_β function is computed as:

$$F_\beta = \frac{(\beta^2 + 1) * p * r}{\beta^2 * p + r} \tag{4.1}$$

where β may be seen as the relative degree of importance attributed to *precision* and *recall*. If $\beta = 0$ then F_β coincides with *precision*, whereas if $\beta = \infty$ then F_β coincides with *recall*. Usually, a value $\beta = 1$ is used, which attributes equal importance to *precision* and *recall*. Thus, the F_1 function is computed as:

$$F_1 = \frac{2 * p * r}{p + r} \tag{4.2}$$

Similarly, the F_1 function also can be estimated from two ways, i.e. micro-averaged (where the *precision* and *recall* are obtained by summing over all individual decisions) and macro-averaged (where *precision* and *recall* are first evaluated "locally" for each category, and then "globally" by averaging over the results of the different categories). Moreover, these two methods also give quite different results, especially if the different categories have very different generality.

4.3.3 Breakeven Point

The *breakeven point* is the value at which *precision* equals *recall*. To obtain the *breakeven point*, a plot of *precision* as a function of *recall* is computed by repeatedly varying the threshold ρ; thus the *breakeven point* value is the value of *precision* or *recall* for which the plot intersects the $precision = recall$ line. This idea relies on the fact that, by decreasing the parameter ρ, *recall* always increases monotonically from 0 to 1 and *precisions* usually decreases monotonically from a value near 1 to lower. If for no values of the ρ *precision* and *recall* are exactly equal, the ρ is set to the value for which *precision* and *recall* are closest, or an interpolated *breakeven* is computed as the average of the values of *precision* and *recall*. We must note that

there may be no parameter setting that yields the *breakeven*; in this case the final *breakeven* value obtained by interpolation is artificial.

Compared with the F_1 function, [YL99] showed that the *breakeven point* of a classifier is always less or equal than its F_1 value.

4.3.4 Accuracy

Although *accuracy* is commonly used in the machine learning literature, it is not widely used in TC. The reason is that, as [YL99] pointed out, the large value of documents in the whole corpus makes them much more insensitive to variations in the number of correct decisions than *precision* and *recall*. This makes the classifier behave very much like a trivial rejector.

Once an effectiveness measure is chosen, a classifier can be tuned by setting thresholds and other parameters so that the resulting effectiveness is the best achievable by that classifier. Tuning a parameter ρ is normally done experimentally. This means performing repeated experiments on the validation set with the values of the other parameters ρ_k fixed and with different values for parameter ρ. The value that has yielded the best effectiveness is chosen for ρ. In general, our experiments adopted the *micro-averaged precision/recall break-even point* and F_1 functions as the measures of performance.

4.4 Statistical Significance Tests

To verify the impact of the difference in data on the performance variation of these term weighting methods and further evaluate whether there is significant difference between different methods, we employed the McNemar's significance tests [Die98] based on the evaluation measures chosen from the previous section, i.e. micro-averaged F_1 or *break-even point*. McNemar's test is a χ^2-based significance test for goodness of fit that compares the distribution of counts expected under the null hypothesis to the observed counts. The McNemar's test can be summarized as follow.

Two classifiers f_A and f_B based on two different term weighting schemes were performed on the test set. For each example in test set, we recorded how it was classified and constructed the following contingency table (Table 4.3). The null

Table 4.3: McNemar's test contingency table

$n00$: Number of examples misclassified by both classifiers f_A and f_B	$n01$: Number of examples misclassified by f_A but not by f_B
$n10$:Number of examples misclassified by classifiers f_B but not by f_A	$n11$: Number of examples misclassified by neither f_A nor f_B

hypothesis for the significance test states that on the test set, two classifiers f_A and f_B will have the same error rate, which means that $n10 = n01$. Then the statistic χ is defined as

$$\chi = \frac{(|n01 - n10| - 1)^2}{n01 + n10} \tag{4.3}$$

where $n01$ and $n10$ are defined in Table 4.3.

Dietterich showed that under the null hypothesis, χ is approximately distributed as χ^2 distribution with 1 degree of freedom, where the significance levels 0.01 and 0.001 corresponded to the two thresholds $\chi_0 = 6.64$ and $\chi_1 = 10.83$ respectively. Given a χ score computed based on the performance of a pair of classifiers f_A and f_B, we compared χ with threshold values χ_0 and χ_1 to determine if f_A is superior to f_B at significance levels of 0.01 and 0.001 respectively. If the null hypothesis is correct, then the probability that this quantity is greater than 6.64 is less than 0.01. Otherwise we may reject the null hypothesis in favor of the hypothesis that the two term weighting schemes have different performance when trained on the particular training set.

After setting up the whole experimental conditions for this thesis, to address the three questions raised in the beginning of this thesis, we will conduct comprehensive experiments under various circumstances in the next chapter.

CHAPTER 5
EXPERIMENTAL RESEARCH

This chapter presents a series of experimental research work of this thesis. The main purpose of this chapter is to address to the three questions raised in Chapter 1. To accomplish this, we first explore various widely-used traditional term weighting methods from IR domain on a SVM classifier. Then we investigate several supervised term weighting methods and their relationship with different machine learning algorithms. Finally, we extend the experiment on a new application domain, i.e. biomedical literature classification.

Another purpose of this chapter is to provide empirical evidence to examine the effectiveness of $tf.rf$ under a more general experimental circumstances, i.e. various traditional and the state-of-the-art supervised term weighting methods,

different machine learning algorithms, different benchmark corpora and even different application domains.

This chapter is organized as follows. Section 5.1 describes the experiment on exploring various (in most cases traditional) term weighting methods for SVM-based classifier. Section 5.2 compares supervised and unsupervised term weighting methods and also investigates their relationship with different machine learning algorithms given different benchmark corpora. Section 5.3 presents a further study of these term weighting methods on a new domain, i.e. biomedical literature.

5.1 Experiment Set 1: Exploring the Best Term Weighting Method for SVM-based Text Categorization

The goal of this experiment is to address the second question raised in Chapter 1, i.e. "which is the best term weighting method for SVM-based text classifier?".

Another purpose of this experiment is to make a comprehensive comparative study on various traditional term weighting methods and their variants since no such a comprehensive comparison has been made in TC so far.

To accomplish this, we build a fixed universal platform to explore various widely-used traditional term weighting methods from IR and our newly-proposed $tf.rf$ based on a SVM text classifier. Based on this universal platform, the comprehensive comparative study among a variety of term weighting methods is more meaningful. So we only change the term weighting methods based on the bag-of-words approach, while the remaining background conditions such as, data preparation, classifier and evaluation measures remain unchanged.

5.1.1 Term Weighting Methods

The ten term weighting methods involved in this experiment is listed in Table 5.1. Most of them are traditional term weighting methods which have been widely used in IR and borrowed by TC and/or have shown good performance in practice.

Table 5.1: Summary of 10 term weighting methods studied this experiment set 1

Name	**Description**
binary	binary feature representation
tf	tf only
$logtf$	$log(1+tf)$
ITF	$1-1/(1+tf)$
idf	*idf* only
$tf.idf$	traditional $tf.idf$
$logtf.idf$	$log(1+tf).idf$
$tf.idfprob$	probabilistic *idf*, actually is the approximate $tf.term\ relevance$
$tf.chi$	$tf.\chi^2$
$tf.rf$	$tf.relevance\ frequency$ is our new weighting scheme

The first four term weighting methods are actually different variants of tf factor. Specifically, *binary* is the simplest and popularly used representation in all learning algorithms, especially in which the real number term weights can not be input. The raw term frequency(tf) and one of its log operation variants, $\log(1+tf)$ (used in [BSAS94]) are included, where the logarithm operation is used to scale

the effect of unfavorably high term frequency in a document. Moreover, inspired by the inverse document frequency, a ITF (inverse term frequency) was presented by [LK02] for a SVM-based text classifier.

The next four methods are different variants of $tf.idf$ method. Note that we do not include RSJ and BM25 rooted from IR because the two can be simplified as $tf.idf$ or a variant of $tf.idf$ when applied to TC tasks (for details see subsection 3.4.4). There may be other variants which we do not cover here because they share the same idea of $tf.idf$ and their formats are basically similar with each other.

The $tf.chi$ scheme is included for two reasons. First, it is a typical representation which combines tf factor with one feature selection metric (here is χ^2). Another reason is in [DTY+04], the authors stated that $tf.\chi^2$ performs better than $tf.idf$ for SVM-based text classifier.

The final weighting representation is our newly proposed scheme based on the analysis of term's discriminating power in Section 3.7 in order to improve the terms' discriminating power for TC.

Other weighting schemes exist, but these ten methods were chosen due to their reported superior classification results or their typical representation when using SVM. For example, ITF representation proposed by [LK02] is included because the experimental results show that when combined with linear SVM it needs the minimum of *support vectors* (i.e. best generalization).

5.1.2 Results and Discussion

We conduct comparative experiment on Reuters Corpus and 20 Newsgroups Corpus by using these ten term weighting schemes in combination with linear SVM classifier. Specifically, as for the 20 Newsgroups Corpus, we randomly select the first 200 training samples and the first 100 testing samples per category. On a chosen category, 200 positive training samples and 3800 negative training samples evenly distributed in the other 19 categories are used for the classifier.

Experimental Results

Figure 5.1 depicts the micro-averaged break-even point performance on the Reuters corpus. The performance of different term weighting schemes at a small vocabulary size cannot be summarized in one sentence but the trends are distinctive that the break-even points of different term weighting schemes increase as the number of the features grows. All term weighting schemes reach a maximum of break-even point at the full vocabulary. Among these, the best break-even point 0.9272 is reached at the full vocabulary by using our newly proposed scheme $tf.rf$. The $tf.rf$ scheme has always been shown a significantly better performance than others when the number of features is larger than 5000. The following significance tests results support this observation.

Table 5.2 summarizes the statistical significance tests results on the Reuters corpus at different numbers of features, where the term weighting schemes with insignificant performance differences are grouped into one set, "<" and "<<" denotes worse than at significance level 0.01 and 0.001 respectively.

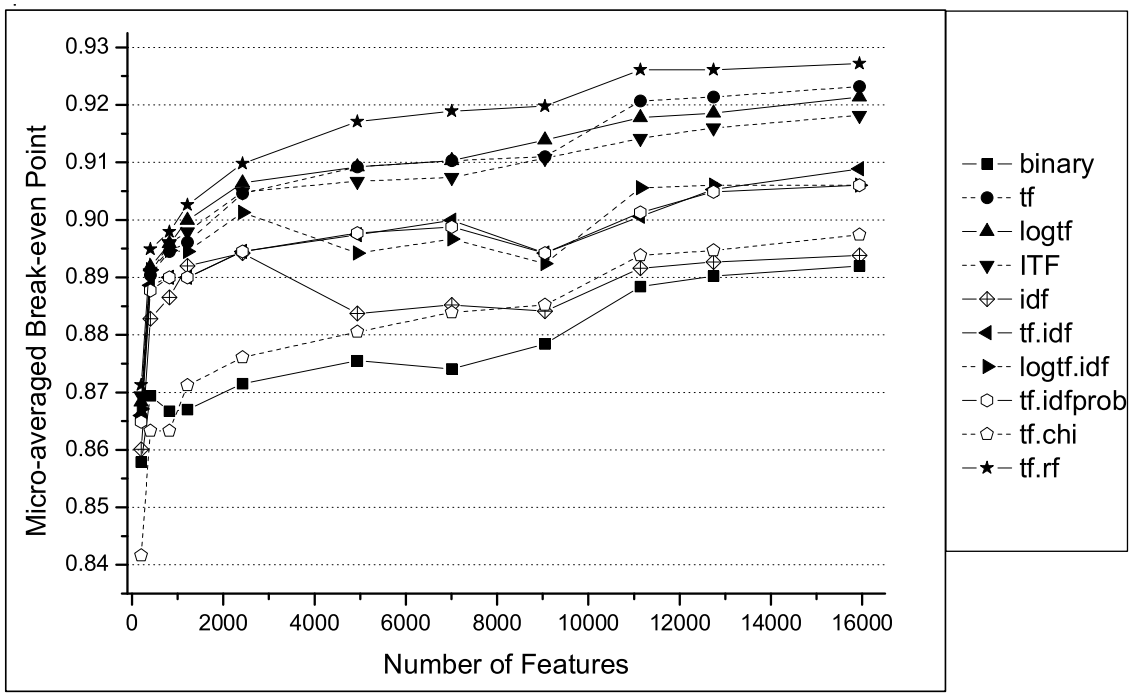

Figure 5.1: Micro-averaged break-even points results for the Reuters-21578 top ten categories by using ten term weighting schemes at different numbers of features.

Table 5.2: Statistical significance tests results on Reuters-21578 at different numbers of features.

#_Features	McNemar's Test
200	$\{tf.chi\} << \{\text{all the others}\}$
400 - 1500	$\{binary, tf.chi\} << \{\text{all the others}\}$
2500	$\{binary, tf.chi\} << \{idf, tf.idf, tf.idf\text{-}prob\} < \{\text{all the others}\}$
5000 - All	$\{binary, idf, tf.chi\} << \{tf.idf, logtf.idf, tf.idf\text{-}prob\} << \{tf, logtf, ITF\} < \{tf.rf\}$

Figure 5.2 shows the micro-averaged break-even point results on the subset of the 20 Newsgroups corpus. Unlike the trends on the Reuters data set, the perfor-

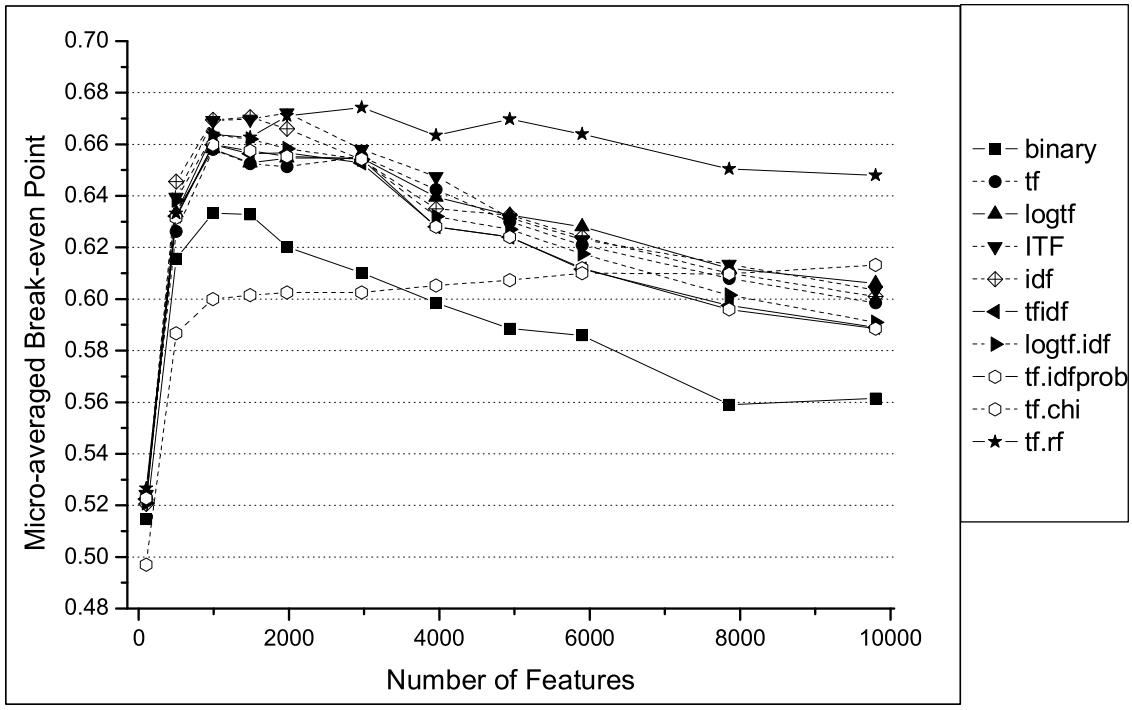

Figure 5.2: Micro-averaged break-even points results for the subset of 20 Newsgroups corpus by using ten term weighting schemes at different numbers of features.

mance curves on the 20 Newsgroups are not monotonically increasing. All term weighting schemes reach their maximum break-even point at a small vocabulary range from 1000 to 3000. The best break-even point 0.6743 is also achieved by using our newly proposed scheme $tf.rf$ at a vocabulary size of 3000.

Table 5.3 summarizes the statistical significance tests results on the subset of the 20 Newsgroups corpus at different numbers of features, where the term weighting schemes with insignificant performance differences are grouped into one set, "$<$" and "$<<$" denotes worse than at significance level 0.01 and 0.001 respectively.

Table 5.3: Statistical significance tests results on the subset of 20 Newsgroups at different numbers of features.

#_Features	McNemar's Test Result
100 - 500	$\{tf.chi\} << \{\text{all the others}\}$
1000	$\{tf.chi\} << \{binary\} << \{\text{all the others}\}$
1500	$\{tf.chi\} << \{binary\} < \{\text{all the others}\} < \{ITF, idf, tf.rf\}$
2000	$\{binary, tf.chi\} << \{\text{all the others}\} < \{ITF, tf.rf\}$
3000 - 5000	$\{binary, tf.chi\} << \{\text{all the others}\} < \{tf.rf\}$
6000 - 10000	$\{binary\} << \{\text{all the others}\} << \{tf.rf\}$

Effects of Feature Set Size on Performance

To achieve high performance in terms of *break-even point*, the vocabularies used in the two data sets are quite different. The categories in the Reuters data set often consist of diverse subject matters which involve overlapping vocabularies. For example, the documents in the same *acquisition* category may involve diverse subjects about acquisition. In this case, large vocabularies are required for adequate classification performance. Hence, for the Reuters data set, the full vocabulary are required to achieve the best break-even point. However, in the 20 Newsgroups data set, all documents in a category are about a single narrow subject with limited vocabulary. Thus, $50 - 100$ vocabularies per category are sufficient for best performance for the 20 Newsgroups data set.

Effects of Term Weighting Methods on Performance

Even though the best performances are reached at different numbers of vocabularies, these term weighting schemes have been shown consistent performance compared with others on the two different data sets.

Firstly, our newly proposed scheme $tf.rf$ has been shown to give a significantly better performance than the others in the two different data sets based on different category distributions. Both of the best break-even points are achieved by using the newly proposed $tf.rf$ scheme on the two data sets. This result also verifies our analysis in Section 3.7 that the $relevance frequency$ scheme does improve the term's discriminating power for text categorization.

Secondly, there is no observation that *idf* factor adds discriminating power when combined with tf factor together. In the Reuters data set, the three term weighting schemes related with term frequency alone, tf, $logtf$ and ITF achieve higher break-even points than these schemes combined with *idf* factor, $tf.idf$, $logtf.idf$ and $tf.idfprob$. In the 20 Newsgroups data set, the differences between these schemes related with tf alone or with *idf* or both are not significant. This result shows that the *idf* factor does not increase any discriminating power or even decrease the discriminating power of the features. Moreover, it also shows that the variants of term frequency have no significant difference.

Thirdly, compared with other schemes, the *binary* and $tf.chi$ schemes show consistently worse performance even when they achieve the best break-even point performance. The *binary* weighting scheme ignores the information of term frequency which is crucial to the representation of the content of the document. This

might be the reason why these schemes related with term frequency have been shown to give drastically better performance than *binary* scheme. The $tf.chi$ scheme, as a representative of term weighting schemes combined with a feature selection measure, though taking the collection distribution into consideration, has been shown to give even worse performance than *binary* representation at a small vocabulary size in the two data sets. As we analyzed in Section 3.7, the d value dominates the χ^2 value and the resulting term weighting value cannot express the term's discriminating power as appropriate as the $tf.rf$. Although $tf.chi$ has been shown to exhibit a slow increasing trend in the 20 Newsgroups data set and gets the higher performance at larger number of vocabularies, its best break-even point performance is still worse than that of the others. Specifically, the $tf.chi$ scheme has been shown no significantly different or even worse performance than the $tf.idf$ scheme. This finding contradicts with the previous result in [DTY+04].

Generally, the ITF scheme has a comparable good performance in the two data sets as other schemes related with term frequency alone, such as tf and $logtf$ factor, but still worse than the $tf.rf$ scheme.

We would like to point out that all the observations are supported by the McNemar's significance tests.

5.1.3 Concluding Remarks

With respect to the second question, i.e. "Among the various term weighting methods, which is the best term weighting method for SVM-based text classifier?", the answer is:

So far, the newly proposed $tf.rf$ method consistently performs the best.

This experiment set 1 not only addresses the second question but also provides an experimental evidence to confirm the effectiveness of the newly proposed $tf.rf$ method and thus replies to the first question under particular circumstances.

Besides, the following interesting conclusions with empirical evidence also are drawn:

- There is no significant difference among the methods related with term frequency alone,i.e. tf, $logtf$, ITF; these three variants of term frequency have been shown rather good performance but still not as good as the $tf.rf$ scheme.
- There is no significant difference between *tf.idf* and its two variants, namely, $logtf.idf$ and $tf.idfprob$.
- The *idf* and χ^2 factor, taking the collection distribution into consideration, do not improve or even decrease the term's discriminating power for text categorization.
- The *binary* and $tf.chi$ methods significantly underperform in comparison with the other term weighting methods.

Note that the observations above are made in combination with linear SVM in terms of micro-averaged break-even point measure.

5.2 Experiment Set 2: Investigating Supervised Term Weighting Methods and Their Relationship with Machine Learning Algorithms

Even thought in the previous experiment set, our newly proposed $tf.rf$ has been shown a better performance than other traditional (unsupervised) term weighting methods and χ^2 scheme in a SVM-based classifier, it is interesting to see if we can observe the similar results when including other supervised term weighting methods (such as *information gain*, *gain ratio*, *Odds Ratio* and etc.) on more general experimental conditions.

The main purpose of this experiment is to address the third question raised in Chapter 1. Another purpose is to provide empirical evidence to examine the effectiveness of $tf.rf$ under a more general experimental circumstances, i.e. different supervised methods, different learning algorithms. To accomplish this, we explore a number of widely-used supervised term weighting methods and unsupervised term weighting methods (selected based on the results in experiment set 1) and investigate their relationship with different learning algorithms, i.e. SVM and kNN on two full benchmark corpora.

5.2.1 Methodology

Term Weighting Methods Involved

We carefully pick out eight different supervised and unsupervised term weighting methods listed in Table 5.4. These eight weighting methods are chosen due to

their reported superior classification results or their typical representation in TC (For details about their formulae, see Section 3.6).

Table 5.4: Summary of 8 supervised and unsupervised term weighting methods

Methods	Denoted by	Description
Unsupervised term Weighting	*binary*	0 for absence or 1 for presence
	tf	term frequency alone
	$tf.idf$	classic *tf.idf*
Supervised term weighting	$tf.rf$	our newly proposed scheme
	rf	rf factor alone, i.e. $binary * rf$
	$tf.\chi^2$	$tf.chi^2$
	$tf.ig$	$tf.information\ gain$
	$tf.or$	$tf.Odds\ Ratio$

The first three are unsupervised term weighting methods which are chosen based on the experimental results in Experiment Set 1. The tf alone method and its two variants, i.e. $logtf$ and ITF, have been shown satisfactory results and the previous experiment also showed that there is no significant difference among them. As a result, we will only include the *tf* alone and *binary* for further research in this section. In addition, the experimental results in Experiment Set 1 indicate there is no significant difference among $tf.idf$ and its variants, $\log(1+tf).idf$ and *tf.idfprob*, and thus only the most popular $tf.idf$ method serves as a baseline in this experiment and its variants are not included here.

The last five are supervised term weighting methods. Besides our newly proposed $tf.rf$ method, the rf alone method (which is actually the product of *binary*

factor and rf factor), is included to explore the effects of rf alone factor on the classification task. The $tf.\chi^2$ and $tf.ig$ methods are two typical representatives which are based on information-theory and the two functions have been widely used for feature selection. We also investigate the $tf.gr$ (*gain ratio*) method. Given the local policy for term weighting, the result of *gain ratio* is identical with that of *information gain* since they only differ in a constant multiplicative factor. The $tf.or$ method is derived from the probabilistic model and it was actually the *term relevance* factor in IR.

Inductive Learning Algorithms

In this experiment, we choose two top-notch inductive learning algorithms, namely, kNN and SVM. The two classifiers also differ statistically, i.e. SVM is based on the structural risk minimization principle while kNN is an example-based classifier. Hence, to investigate the performance of different term weighting methods, an evaluation using both classifiers should reduce the possibility of classifier bias in the results.

5.2.2 Results and Discussion

We conduct comparative experiments on the eight supervised and unsupervised term weighting methods in combination with SVM and kNN on two benchmark corpora. The experimental results are reported from Figure 5.3 to Figure 5.10. Each figure reports the performances of eight term weighting methods with respect to micro- or macro-averaged F_1 measure in combination of one algorithm and one

corpus, and each curve in the figures shows the performance of each term weighting method as the number of features grows.

Results on the Reuters Corpus using SVM

Figure 5.3 and 5.4 depict the micro- and macro-averaged F_1 performance of different term weighting methods on the Reuters corpus using the SVM algorithm.

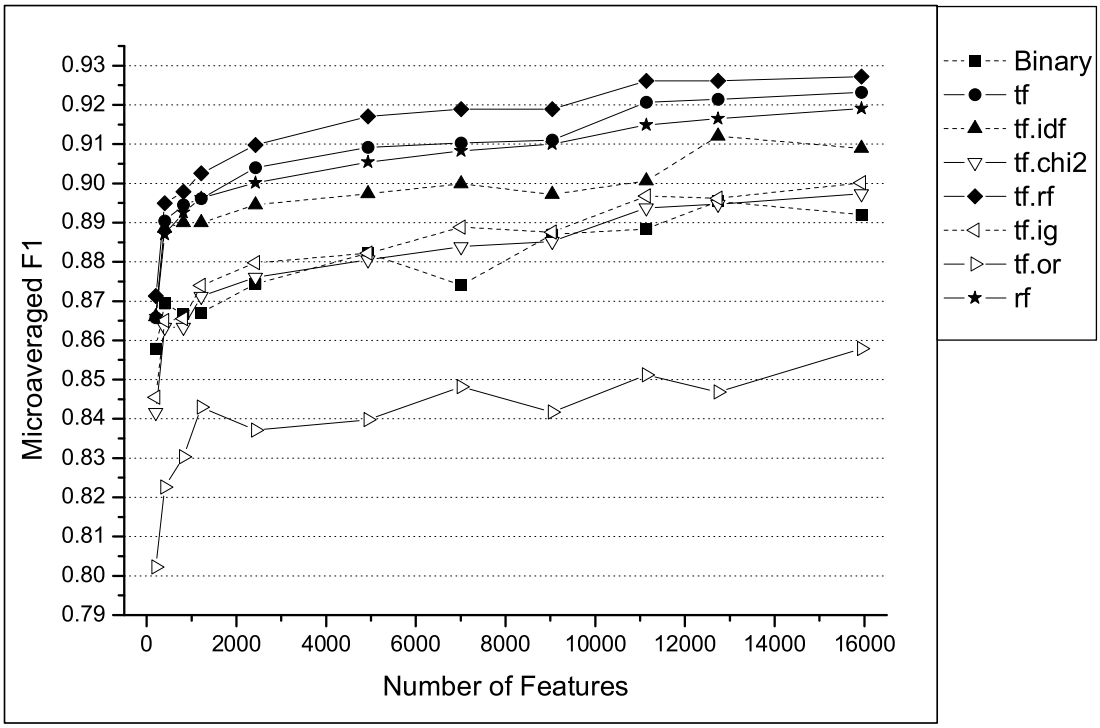

Figure 5.3: Micro-averaged F_1 measure of the eight unsupervised and supervised term weighting approaches on the Reuters-21578 top ten categories using linear SVM algorithm with different numbers of features

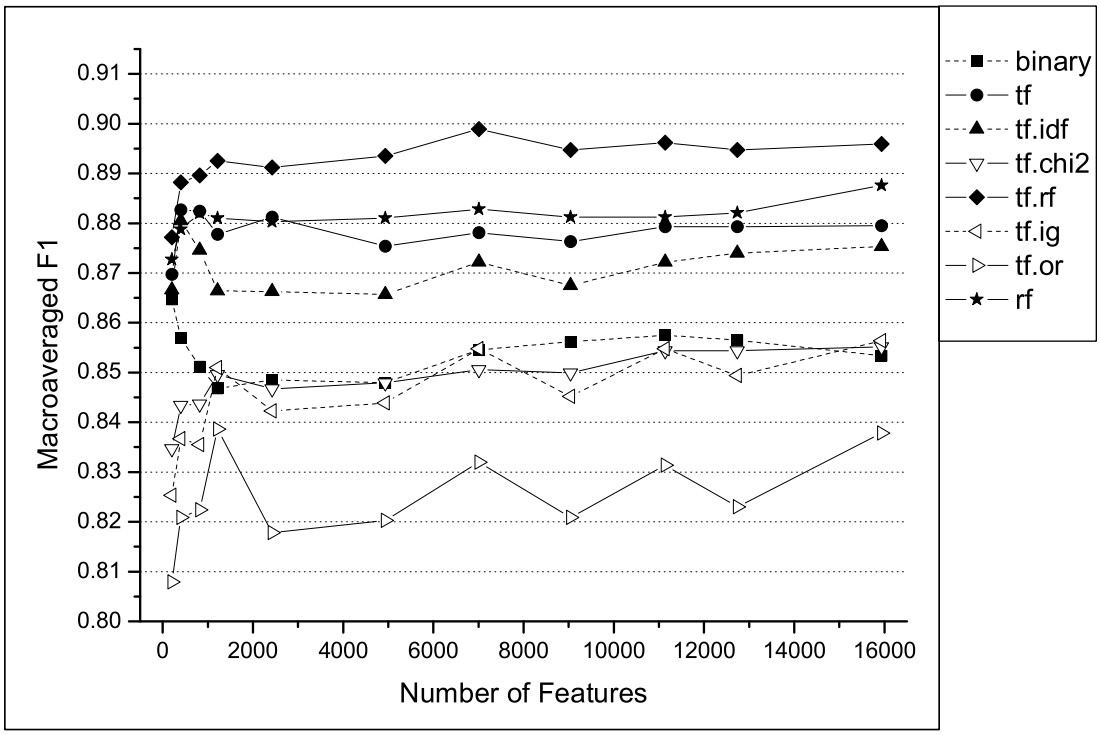

Figure 5.4: Macro-averaged F_1 measure of the eight unsupervised and supervised term weighting approaches on the Reuters-21578 top ten categories using linear SVM algorithm with different numbers of features

The performance of different term weighting methods at a small vocabulary size cannot be summarized in one sentence but the trends are distinctive in that the micro-averaged F_1 points of different term weighting methods generally increase as the number of features grows. All term weighting methods reach a maximum micro-averaged F_1 point at the full vocabulary. Among these, the best three micro-averaged F_1 points 0.9272, 0.9232 and 0.9191 are reached at the full vocabulary, using $tf.rf$, tf and rf, respectively. The $tf.rf$ method has always been shown to perform better than others when the number of features is larger than 5000. In

contrast to the performance in terms of micro-averaged F_1 measure, the performance of these methods in terms of macro-averaged F_1 measure does not increase significantly as the number of features grows. However, these methods show consistent performance in macro-averaged F_1 and the $tf.rf$ method is still the best among these methods.

Results on the 20 Newsgroups Corpus using SVM

Figure 5.5 and 5.6 depict the micro- and macro-averaged F_1 performance of different term weighting methods on the 20 Newsgroups corpus using the SVM algorithm.

The trends of the curves are similar to those in Figure 5.3. That is, the micro-averaged F_1 points of different term weighting methods show a tendency to increase as the number of the features grows. However, these curves approach a plateau when the number of features exceeds 5000. Finally, almost all the term weighting schemes reach a maximum of micro-averaged F_1 point at the full vocabulary. Among them, the best three micro-averaged F_1 points 0.8081, 0.8038 and 0.8012 are reached at the full vocabulary, using rf, $tf.rf$ and $tf.idf$, respectively. Moreover, the performance of these methods in macro-averaged F_1 measure is quite similar to that in micro-averaged F_1 measure on this nearly uniform category distribution corpus.

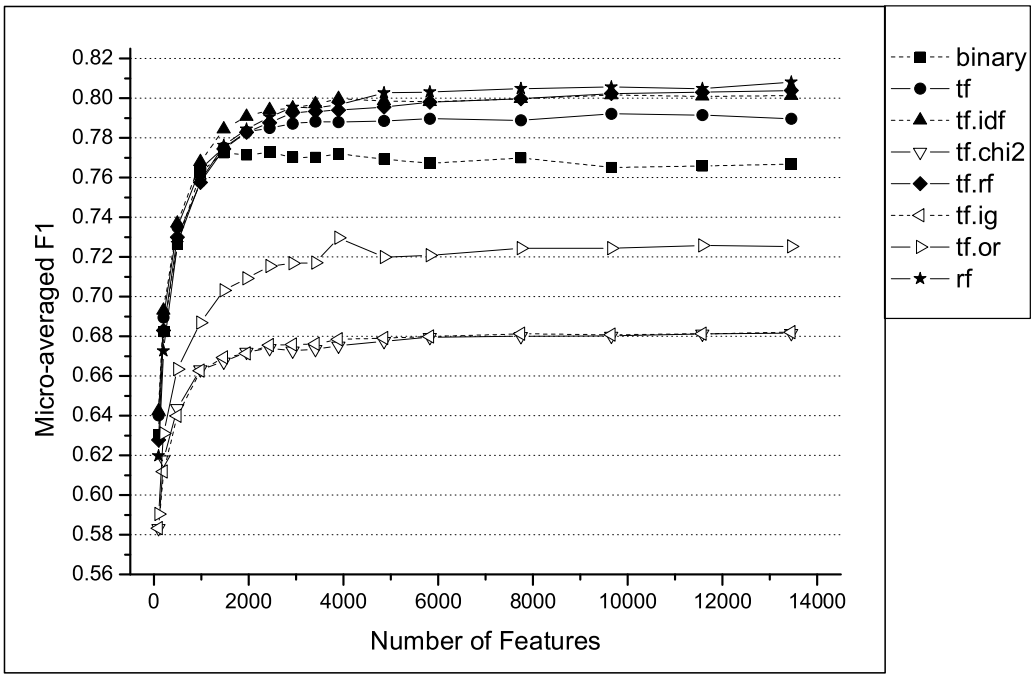

Figure 5.5: Micro-averaged F_1 measure of the eight unsupervised and supervised term weighting approaches on 20 Newsgroups Corpus using linear SVM algorithm with different numbers of features

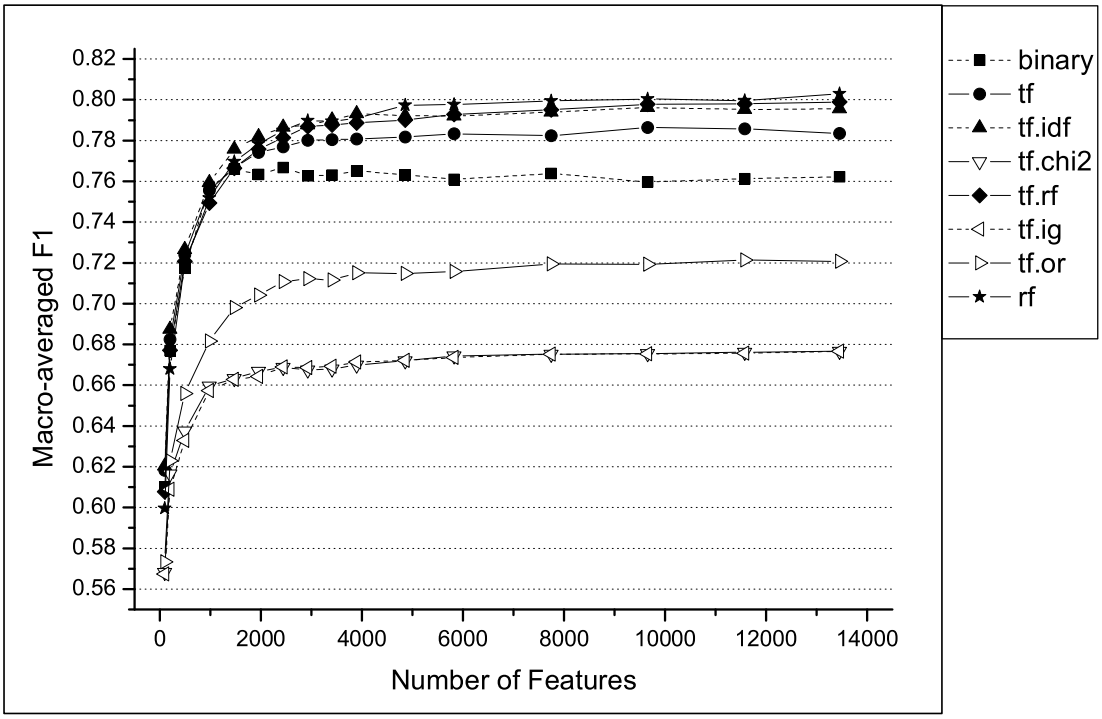

Figure 5.6: Macro-averaged F_1 measure of the eight unsupervised and supervised term weighting approaches on 20 Newsgroups Corpus using linear SVM algorithm with different numbers of features

Results on the Reuters Corpus using *k*NN

Figure 5.7 and 5.8 depict the micro- and macro-averaged F_1 performance of different term weighting methods on the Reuters corpus using the kNN algorithm.

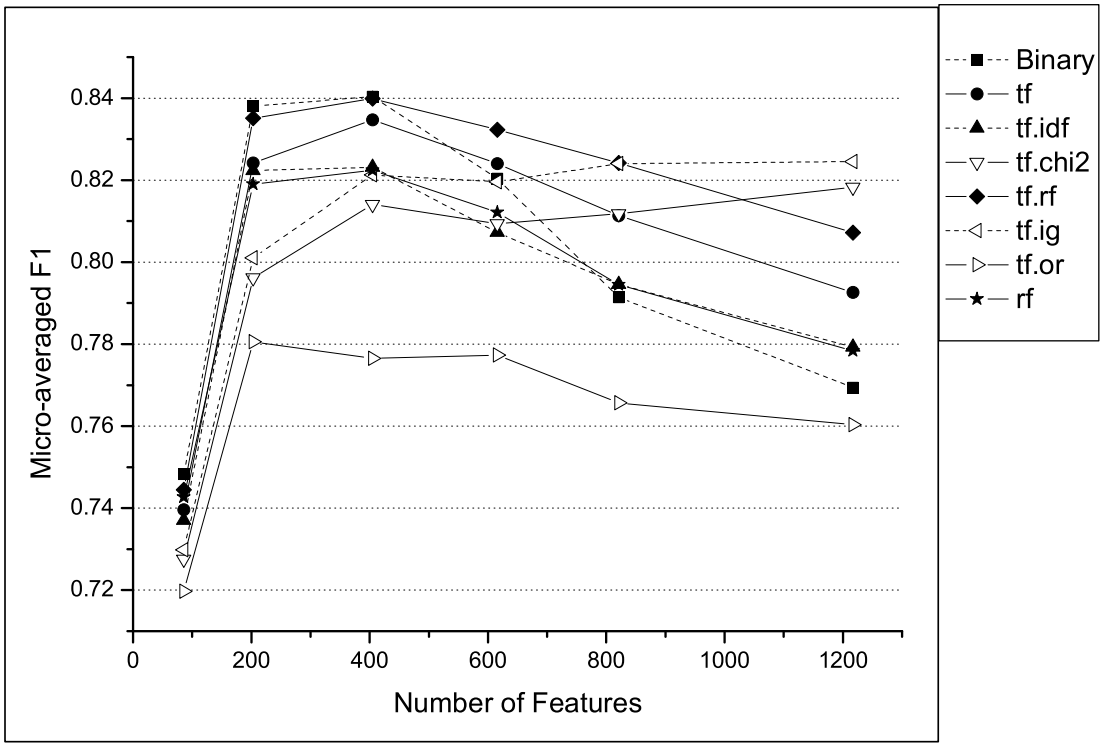

Figure 5.7: Micro-averaged F_1 measure of the eight unsupervised and supervised term weighting approaches on the Reuters-21578 top ten categories using kNN algorithm with different numbers of features

Unlike the shape of curves on the same corpus (Reuters) for the SVM algorithm, the performance of each term weighting method reaches a peak at a small feature set size around 400 in terms of micro-averaged F_1 performance and around 200 in

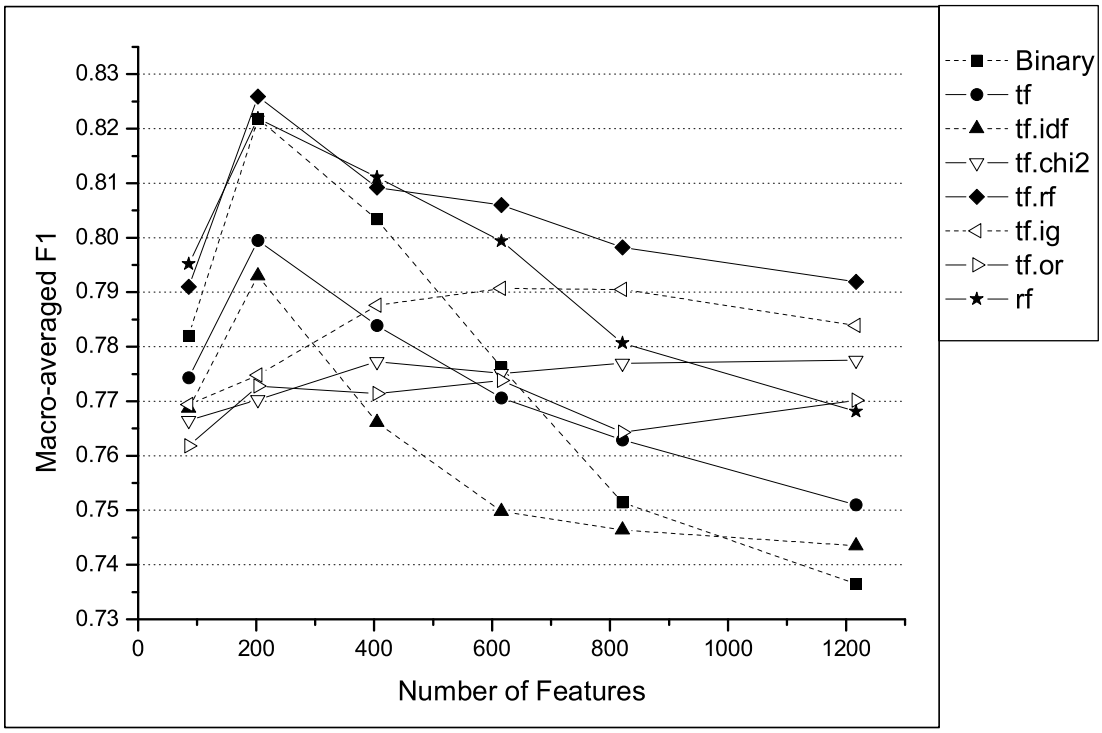

Figure 5.8: Macro-averaged F_1 measure of the eight unsupervised and supervised term weighting approaches on the Reuters-21578 top ten categories using kNN algorithm with different numbers of features

terms of macro-averaged F_1 performance respectively. As the number of features grows, the performance of all methods declines except for $tf.ig$ and $tf.\chi^2$. The best two micro-averaged F_1 points 0.8404 and 0.8399 are achieved by using *binary* and $tf.rf$ methods at the feature set size of 405. Similarly, the best three macro-averaged F_1 points 0.8259, 0.8219 and 0.8218 are achieved at the feature set size of 203 by using $tf.rf$, rf and *binary* methods, respectively. When the number of features is more than 1000, the computation inefficiency of kNN algorithm at the classification time is the most important drawback which prevented us from conducting further experiments.

Results on the 20 Newsgroups Corpus using *k*NN

Figure 5.9 and 5.10 depict the micro- and macro-averaged F_1 performance of different term weighting methods on the 20 Newsgroups corpus using the kNN algorithm.

The trends of curves are generally similar to those based on the Reuters and kNN (Figure 5.7). Almost all the curves reach a peak at a small features size around 500 except for $tf.rf$ and rf. The curves of these two methods show a tendency to increase slowly as the number of features grows. The best two micro-averaged F_1 points 0.6913 and 0.6879 are achieved by using $tf.rf$ and rf when the number of features is around 2000, respectively. Similarly, we did not conduct experiments at a larger vocabulary size due to the built-in inefficiency problem of the kNN algorithm. Moreover, the performance of these methods in macro-averaged F_1 measure is quite similar to the micro-averaged F_1 measure on this nearly uniform category distribution corpus.

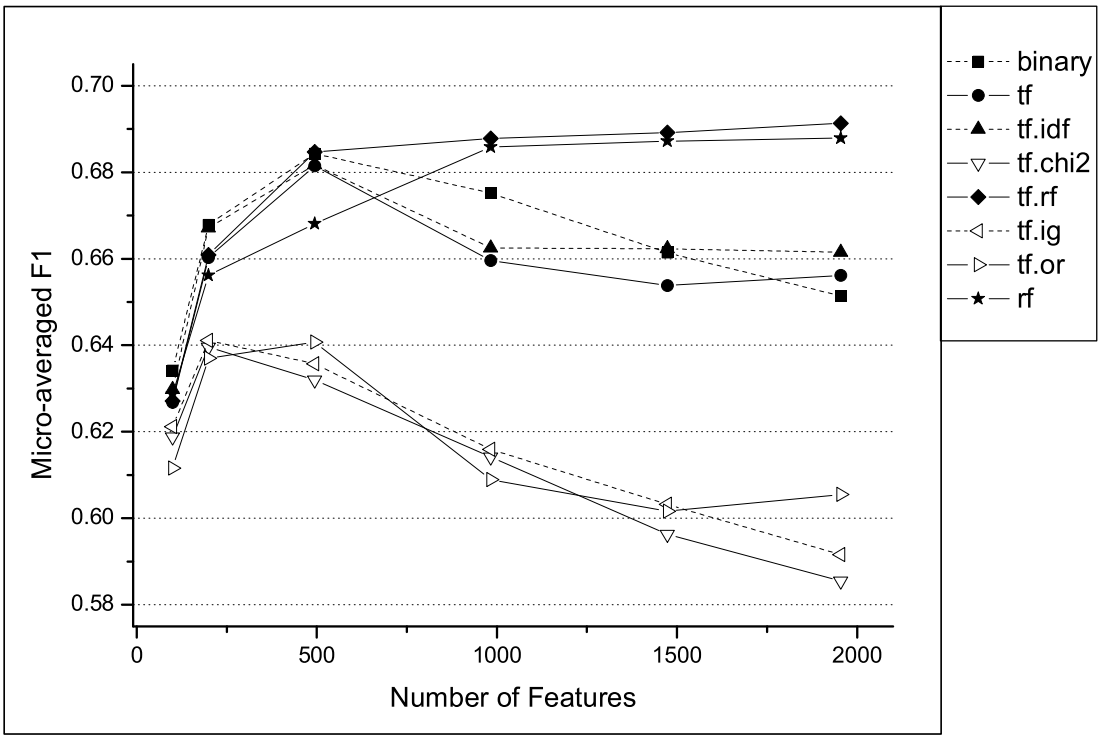

Figure 5.9: Micro-averaged F_1 measure of the eight unsupervised and supervised term weighting approaches on 20 Newsgroups Corpus using kNN algorithm with different numbers of features

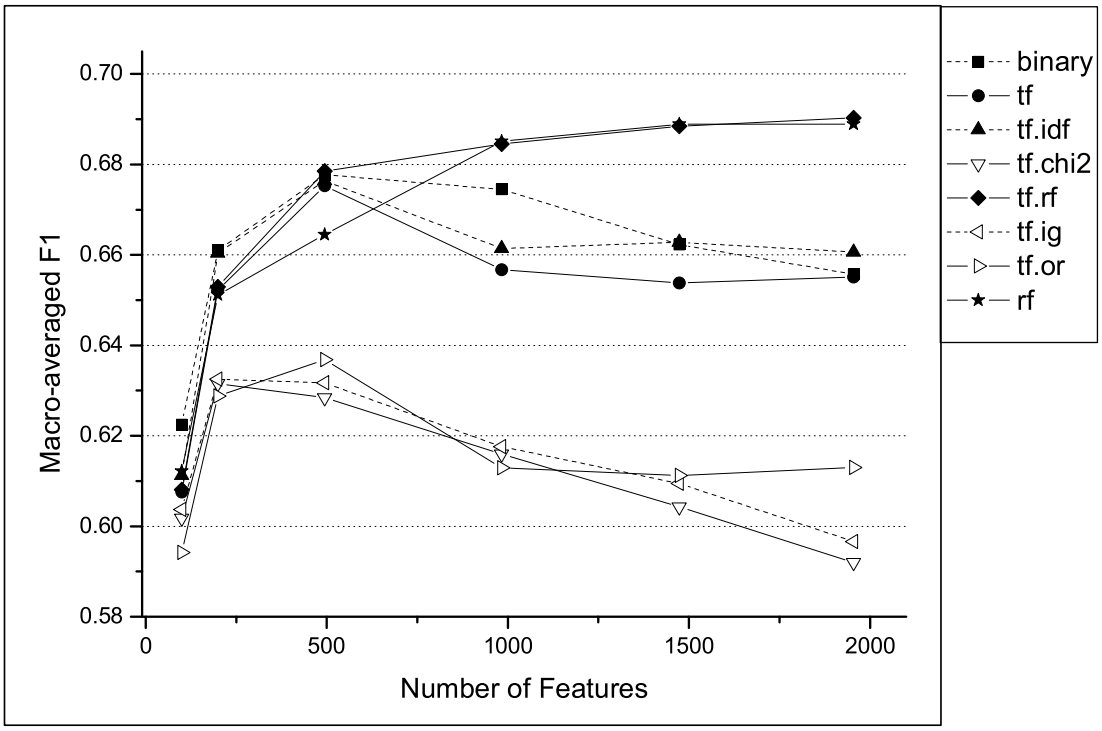

Figure 5.10: Macro-averaged F_1 measure of the eight unsupervised and supervised term weighting approaches on 20 Newsgroups Corpus using kNN algorithm with different numbers of features

Results of McNemar's significance tests

We use the McNemar's tests [Die98] to validate if there is any significant difference between two term weighting methods in terms of the micro-averaged F_1 performance analysis. Table 5.5 summarizes the statistical significance tests results on the two data sets and two learning algorithms at a certain feature set size where most of the methods reach their best performance. The term weighting methods with insignificant performance differences are grouped into one set and ">" and ">>" denote better than at significance level 0.01 and 0.001 respectively.

Table 5.5: Statistical significance tests results on the two data corpora and two learning algorithms at certain numbers of features in terms of the micro-averaged F_1 measure.

Algorithm	Data Corpus	#_Features	McNemar's Test Results
SVM	Reuters	15937	$(tf.rf, tf, rf) > (tf.idf) > (tf.ig, tf.\chi^2, binary) >> tf.or$
SVM	20 Newsgroups	13456	$(rf, tf.rf, tf.idf) > tf >> binary >> tf.or >> (tf.ig, tf.\chi^2)$
kNN	Reuters	405	$(binary, tf.rf) > tf >> (tf.idf, rf, tf.ig) > tf.\chi^2 >> tf.or$
kNN	20 Newsgroups	494	$(tf.rf, binary, tf.idf, tf) >> rf >> (tf.or, tf.ig, tf.\chi^2)$

Summary of different term weighting methods

We state that the performance of different term weighting methods is closely related to the learning algorithm (SVM or kNN) and the property of the data corpus (skewed or uniform category distribution). That is, the comparison of supervised and unsupervised term weighting methods should be made in conjunction with the text classifier and the property of the data corpus.

Although these methods achieve their best performance at different numbers of features, the results in Table 5.5 shows approximate ranks of these methods since most of them show consistent performance with respect to each other as the number of features grows given specific data corpus and learning algorithm. Moreover, based on the Figures 5.3 to 5.9, several findings can be found as follows.

Generally, these supervised and unsupervised term weighting methods have not been shown a universally consistent performance on the two corpora and the two different algorithms.

Specifically, these supervised term weighting methods have been shown to give two extremes results in all experiments. On the one hand, our $tf.rf$ method consistently shows the best performance with respect to the two different algorithms and text corpora. Moreover, the rf alone method which ignores the term frequency also shows a comparable performance to $tf.rf$ in most experiments except for Reuters data set using the kNN algorithm (Figure 5.7). On the other hand, the three typical supervised methods based on the information theory, i.e. $tf.\chi^2$, $tf.ig$ and $tf.or$, are the worst methods among these eight methods. In most cases, they are inferior to the unsupervised ones, i.e. tf, $tf.idf$ and $binary$, and also inferior

to the two newly presented special supervised methods, i.e. $tf.rf$ and rf. For example, in Figure 5.3, $tf.ig$ and $tf.\chi^2$ outperform *binary* representation on Reuters data collection using the linear SVM, while the *binary* method outperforms $tf.or$. For another example, in Figure 5.7, $tf.ig$ is comparable to rf and $tf.idf$, and again $tf.or$ is the worst method of all. These findings indicate that these sophisticated methods based on information theoretic functions have no superiority over the simpler unsupervised ones. This is contrary to our original expectation that the supervised term weighting methods which consider the document distribution in the training documents should always be better than the unsupervised ones.

Moreover, the performance of the unsupervised term weighting methods, i.e. tf, $tf.idf$ and *binary*, is dependent on the special data corpus and the learning algorithm in use. That is, the three have no consistently performance with respect to each other. For example, for the SVM-based text classifier (Figure 5.3 and Figure 5.5), *binary* is the worst among these three methods. However, in Figure 5.3, tf is better than $tf.idf$ and but it is the other way round in Figure 5.5. On the other hand, different behaviors are noted for the kNN-based text classifier. For example, in Figure 5.7, *binary* is the best and $tf.idf$ is the worst among the three unsupervised methods while in Figure 5.9 the performance of the three methods are comparable to each other.

Therefore, several general observations on the unsupervised methods can be made here. First, the popularly-used $tf.idf$ method performs well or even comparable to $tf.rf$ on the 20 Newsgroups corpus using SVM and kNN. This could be explained by the observation that on the skewed corpus the *idf* factor has lost its discriminating power for terms while on uniform corpus it has retained such

power. Recall the two examples in Figure 3.2, $t2$ and $t4$, in skewed categories, even though $t2$ has larger *idf* value (small sum of a and c) than $t4$, $t4$ contribute more than $t2$. This indicates that the property of data corpus has a great impact on *idf*. It is the same case for other *idf* type factors. Second, *binary* performs well or even comparable to $tf.rf$ using kNN on both corpora. However, on SVM-based model, *binary* has rather bad performance. This indicates that kNN favors *binary* while SVM does not. Third, one advantage of tf is its good robustness. Although tf does not have a comparable performance to $tf.rf$ in all experiments, it outperforms many other methods consistently and significantly. This can be verified from Figure 5.3 to Figure 5.9.

5.2.3 Further Analysis

Effects of Feature Set Size on Algorithms

We state that the performance of term weighting methods is closely related to the learning algorithms and data collections. However, the size of the feature set at which each term weighting method reaches the peak is closely dependent upon the learning algorithm in use rather than the term weighting method itself and the benchmark data collection.

Specifically, for the SVM-based text classifier, almost all term weighting methods achieve their best performance when inputting the full vocabulary. The findings in Figure 5.3 and Figure 5.5 indicate this. Since SVM also has the capability to handle thousands or even tens of thousands of features, the traditional feature selection can be omitted. These findings are entirely consistent with those reported

in previous studies (see [DS03] and [Joa98]).

However, for the kNN-based text classifier, all term weighting methods achieve their best performance at a small feature set size. For example, in Figure 5.7, most methods reach a peak at the feature set size of 400 or so, except for $tf.ig$ and $tf.or$. Similarly, in Figure 5.9, most methods attain their maximum performance at the feature set size of 500 except for $tf.rf$ and rf which increase slowly with increasing feature set size.

The possible explanation for this difference lies in their different theoretical rationale. When the size of the input feature set increases, the number of noisy or irrelevant features also increases. However, SVM algorithm is resilient to noise because only the *support vectors* are effective for the classification performance [YL99]; if all other examples (vectors) except support vectors are removed, the model learned will not change. This property makes SVM theoretically different from other algorithms. On the other hand, kNN is an example-based learning algorithm, i.e.,it uses all features equally in computing similarities, which results in the well-known problem with high dimensional spaces [Mit97]. That is, in high dimensional spaces, almost all sparse vectors are equally far apart, thus kNN achieves good performance on classification in terms of small feature set; however, kNN is prone to be fooled by noisy or irrelevant features when the size of the input feature set grows. That is the reason why kNN peaks its performance in the small number of features in the experiments.

Effects of Data Corpora and Algorithms on Term Weighting Methods

To further investigate these methods, we explore the performance of these methods on each category. To do so, we choose four representative methods, namely, $tf.rf$, $tf.\chi^2$, $tf.idf$ and *binary*, and investigate their performance in combination with different algorithms with respect to F_1 measure on each category.

The results are shown from Figure 5.11 to Figure 5.14. Each curve represents a different term weighting method.

Note that these methods achieve their best performance at different feature set size. We analyze the performance of these methods at a certain feature set size where most of the methods achieve their best performance. Even though not all the methods may achieve their best performance, it is still interesting to compare their performance with respect to each other.

1. Reuters Corpus and Linear SVM Algorithm

Figure 5.11 depicts the F_1 measure of four term weighting methods on each of the 10 categories of Reuters-21578 corpus using the SVM-based classifier at the full vocabulary (15937 features).

All the four methods achieve almost the same F_1 on the largest category, i.e. category 3 (*earn*, 39%) while for the other 9 categories there are significant differences among these methods. For example, the maximal difference of F_1 between $tf.rf$ (0.8303) and $tf.\chi^2$ (0.6278) on category 5 (*interest*, 4.7%) is 0.2025.

Among these four methods, $tf.\chi^2$ is the worst on 8 of the 10 categories except

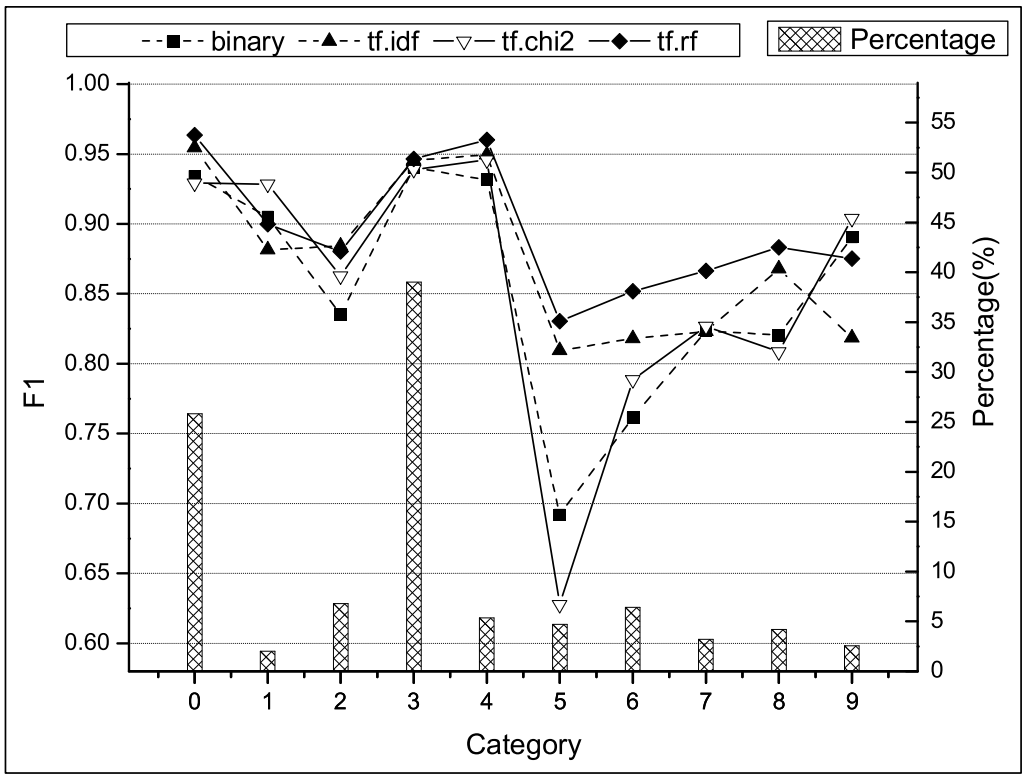

Figure 5.11: F_1 measure of the four term weighting methods on each category of Reuters-21578 corpus using SVM algorithm at the full vocabulary

that it is the best method on the two smallest categories, i.e. category 1 (*corn*, 2%) and category 9 (*wheat*, 2.5%). The two categories are only a small percentage of the whole corpus and thus $tf.\chi^2$ is generally the worst among the four methods.

On the other hand, $tf.rf$ has the best performance on 7 of the 10 categories and thus has contributed the best performance for the whole corpus. This finding shows that $tf.rf$ is quite effective for the skewed category distribution in the Reuters Corpus.

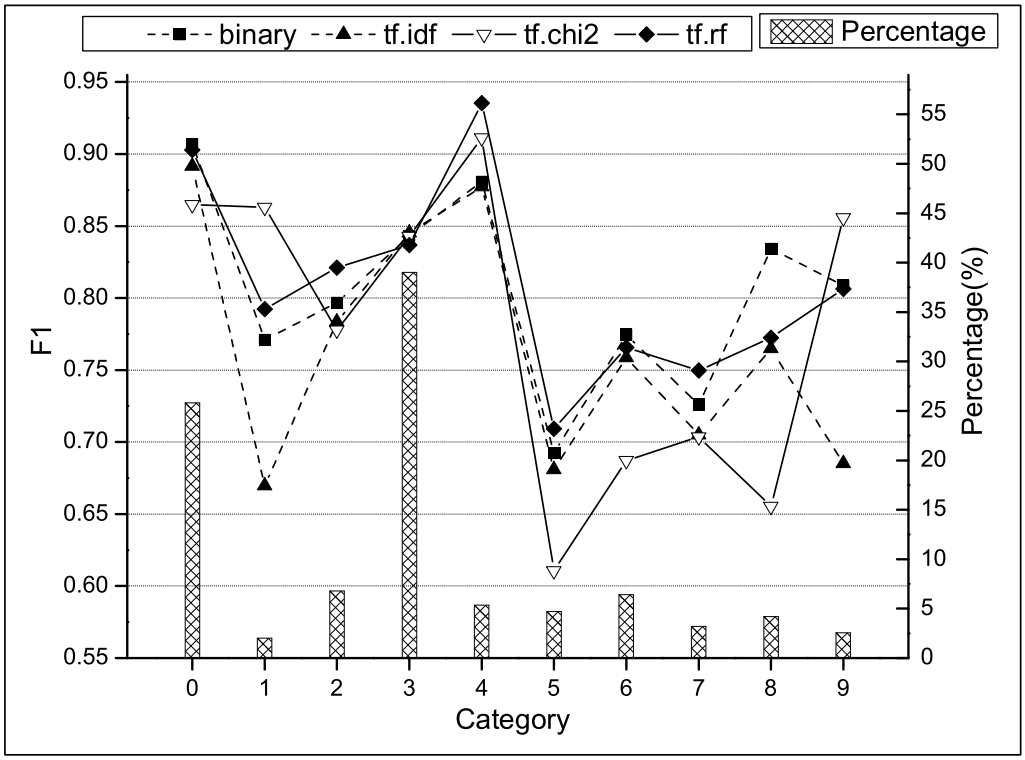

Figure 5.12: F_1 measure of the four term weighting methods on each category of Reuters-21578 corpus using kNN algorithm at a feature set size of 405

2. Reuters Corpus and kNN Algorithm

Figure 5.12 shows the F_1 measure of the four term weighting methods on each of the 10 categories of Reuters-21578 corpus using the kNN-based classifier at a feature set size of 405.

Similar to Figure 5.11, there are no differences among the four methods on the largest category, i.e. category 3 (*earn*, 39%) while on the other 9 categories there are significant differences among these methods.

However, unlike the findings in Figure 5.11, *binary* is the best on 3 of the

10 categories while $tf.rf$ is the best on 4 of the 10 categories. Moreover, the differences between them on most categories are less drastic. Therefore, *binary* and $tf.rf$ are the best two methods of all for the whole corpus with respect to micro-averaged F_1 measure. The findings in Figure 5.11 and Figure 5.12 also show that kNN favors *binary* while SVM does not.

Again, similar to Figure 5.11, $tf.\chi^2$ is the worst method on 6 of the 10 categories except that it is the best method on the two smallest categories, i.e. category 1 (*corn*, 2%) and category 9 (*wheat*, 2.5%), and thus $tf.\chi^2$ is still generally the worst of the four methods.

3. 20 Newsgroups Corpus and Linear SVM Algorithm

Figure 5.13 shows the F_1 measure of the four term weighting methods on each category of the 20 Newsgroups corpus using the SVM classifier at a feature set size of 13456.

Unlike the results on Reuters Corpus that has a skewed category distribution (Figure 5.11 and Figure 5.12), there are significant differences among the four term weighting methods on each of the 20 categories of the 20 Newsgroups corpus. It is clear that $tf.\chi^2$ is the worst method on all the 20 categories.

Similar to Figure 5.11 and Figure 5.12, $tf.rf$ has been shown to perform very well on each category. Furthermore, the traditional $tf.idf$ has also been shown to give the same good performance as $tf.rf$ on most of the 20 categories. Both $tf.rf$ and $tf.idf$ are consistently better than the other two methods, i.e. *binary* and $tf.\chi^2$. The good performance of $tf.idf$ may be attributed to the uniform category

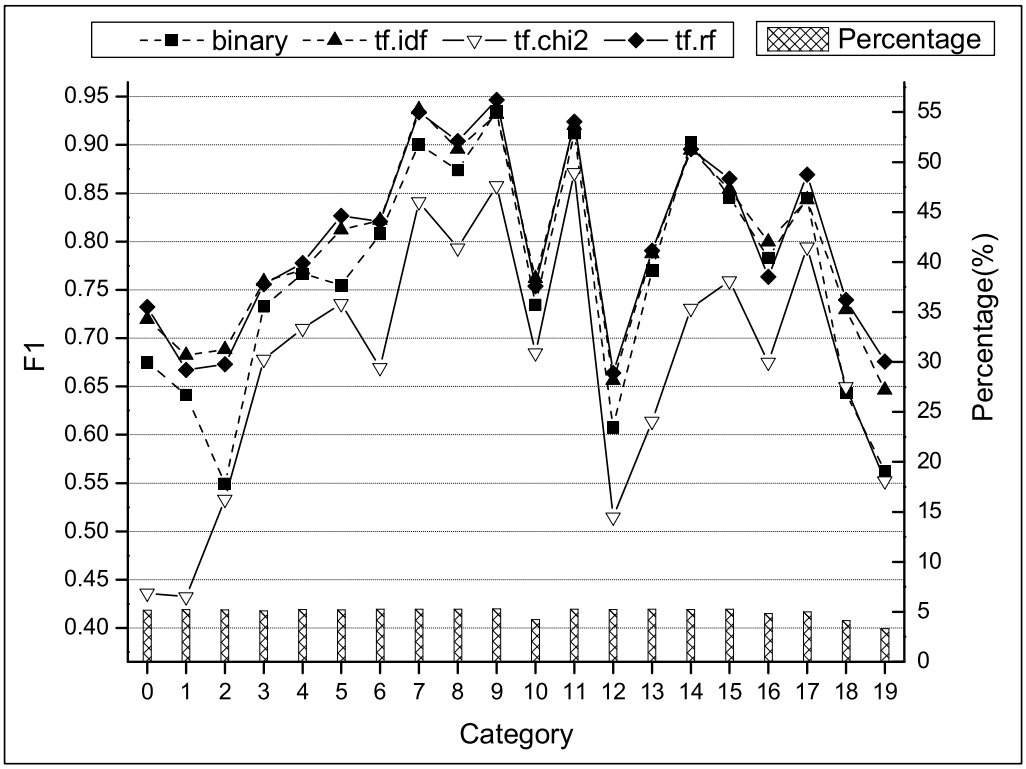

Figure 5.13: F_1 measure of the four term weighting methods on each category of 20 Newsgroups using SVM algorithm at a feature set size of 13456

distribution.

4. 20 Newsgroups Corpus and kNN Algorithm

Figure 5.14 shows the F_1 measure of the four term weighting methods on each category of 20 Newsgroups corpus using the kNN classifier at a feature set size of 494.

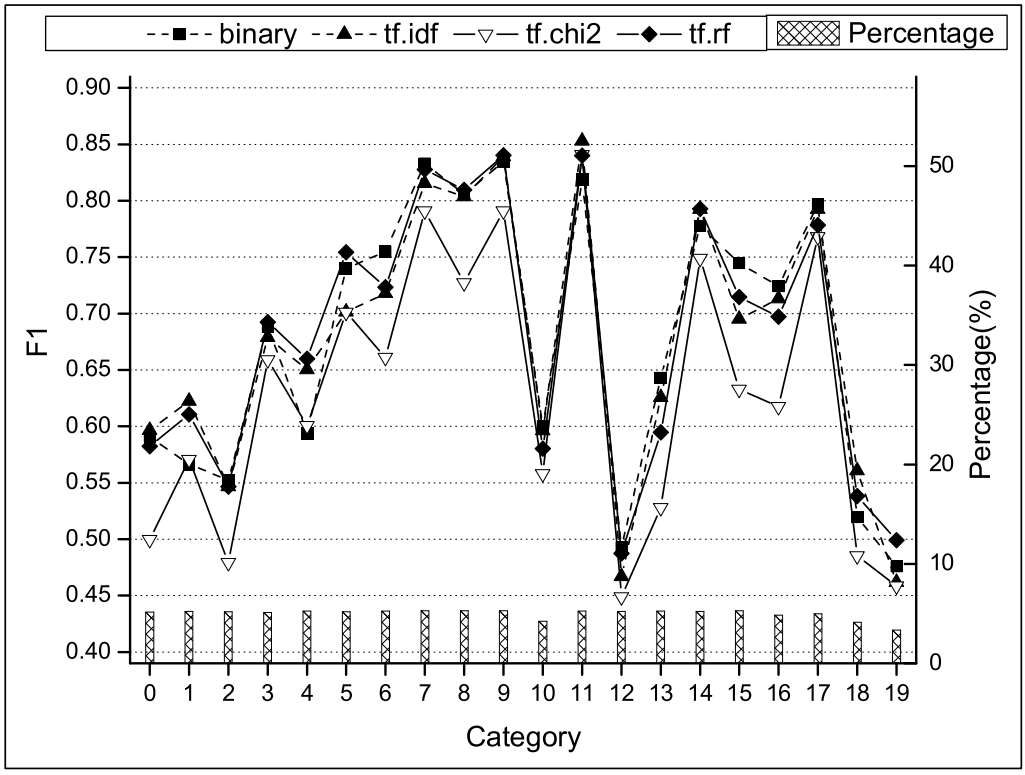

Figure 5.14: F_1 measure of the four term weighting methods on each category of 20 Newsgroups using kNN algorithm at a feature set size of 494

Clearly, $tf.\chi^2$ is the worst method on 18 of the 20 categories.

In terms of F_1 value, *binary*, $tf.rf$ and $tf.idf$ are the best on 10, 7 and 3 of the 20 categories, respectively. From the whole corpus, *binary* and $tf.rf$ are comparable to each other and both are better than $tf.idf$. These findings once again indicate that, first, *binary* performs well on kNN algorithm; second, $tf.rf$ has good robustness on either skewed or uniform category distribution and on either the SVM-based or the kNN-based text classifier; third, *idf* factor is more effective on uniform corpus than on skewed corpus. The difference of $tf.idf$ between different corpora could be explained by the fact that on a skewed corpus the *idf* factor has lost its discriminating power for terms while on uniform corpus it has retained such power.

5.2.4 Concluding Remarks

The following conclusions with empirical evidence address the third question in this thesis including two sub-questions, i.e., "Are supervised term weighting methods based on known information able to lead to better performance than unsupervised ones for text categorization?" and "What kinds of relationship can we find between term weighting methods and the two widely-used learning algorithms, i.e. kNN and SVM, given different benchmark data collections?"

The answer to the first sub-question is: Not always. That is, not all supervised term weighting methods are superior to unsupervised methods. Actually, these supervised term weighting methods are the two extremes in terms of performance. On the one hand, the three supervised methods with solid theoretical bases, i.e. $tf.\chi^2$, $tf.ig$ and $tf.or$, have the worst performance in all experiments. On the other hand, our proposed supervised method $tf.rf$, consistently achieves the best

performance and outperforms other methods substantially and significantly.

The answer to the second sub-question is, the performance of the term weighting methods, especially, the three unsupervised methods, has close relationships with the learning algorithms and data corpora. Their relationships with algorithms and data corpora can be summarized as follows:

- *tf.rf* performs consistently the best in all experiments.
- *tf.or*, $tf.\chi^2$ and *tf.ig* perform consistently the worst in all experiments.
- *tf.idf* performs well or even comparable to $tf.rf$ on the uniform category corpus (i.e. 20 Newsgroups corpus) either using SVM or kNN. This indicates that the property of data corpus has a great impact on idf.
- *binary* performs well or even comparable to $tf.rf$ on both corpora using kNN while rather bad on SVM-based text classifier. This shows that kNN favors *binary* and SVM does not.
- tf has no clear relationship with algorithm or data corpus. But although tf does not perform as well as $tf.rf$, it performs consistently well in all experiments.

The study in this experiment also addresses the first question in this thesis from a more general experimental circumstance. That is, the best performance of $tf.rf$ has been confirmed by all experiments. Given the cross-method comparison (various supervised and unsupervised), and cross-classifier (SVM and kNN) and cross-corpus validation (the Reuters and 20 Newsgroups corpora), we are convinced

that the observed consistently best performance of $tf.rf$ is general rather than corpus-dependent or classifier-dependent.

We suggest that $tf.rf$ should be used as the term weighting method for TC task as it performed consistently well on either skewed or uniform category distribution data benchmark and on either SVM-based or kNN-based text classifiers.

5.3 Experiment Set 3: Application to Biomedical Data Collections

The previous two experiments investigated the effectiveness of our newly proposed supervised term weighting method, i.e. $tf.rf$, cross various term weighting methods comparison and cross different learning algorithms validation on two benchmark corpora in newswire domain.

In this experiment, we extend the $tf.rf$ method to a totally new application domain, that is, biomedical literature domain. The purpose of this experiment is to validate whether $tf.rf$ can improve the performance of biomedical text classification. To accomplish it, we will examine its effectiveness on two biomedical data collections, one is a widely-used benchmark Ohsumed corpus, and another is an in-house manually-collected data corpus related to the Biochemistry and Molecular Biology subject category, i.e. 18 Journals Corpus.

5.3.1 Motivation

With the rapid growth of biomedical research, the volume of published biomedical articles is expanding at an increasing rate. PubMed (http://pubmed.gov) is a U.S. National Library of Medicine's (NLM) database, which has more than 16 million biomedical citations and abstracts that are searchable on the web at no cost. As the largest component of PubMed, the MEDLINE database covers over $4,800$ journals published in the United States and more than 70 other countries primarily from 1966 to the present. The volume of the MEDLINE database is currently growing at the rate of $500,000$ new citations per year. Since year 2002, between $1,500$ - $3,500$ completed references are added each day Tuesday through Saturday. Over $571,000$ records are added during year 2004.

Therefore, the needs for automatically organizing and classifying these biomedical articles increase. The work on biomedical literature classification has been intensively studied only in recent years. In *Knowledge Discovery in Databases* (KDD) Challenge Cup 2002 ([ASYM03]), there is a task to determine whether the paper from the FlyBase data set should be curated given the presence of experimental evidence of *Drosophila* gene products. Two best performing approaches were presented by manually constructed rules and manually chosen "keywords" in [RFLF+02] and [KOS+02] which achieve an F-score of 78% and 73% on determining whether to curate a paper based on the presence of experimental evidence, respectively. [RFLF+02] used a set of manually constructed rules based on POS tagging, a lexicon, and semantic constraints determined by examining the training documents. Moreover, they explored the information on figure captions which were found to be useful. In [KOS+02], authors manually picked out "keywords"

and computed the distance between keywords and gene names. The two best approaches are based on domain knowledge and domain experts, which is very expensive and does not provide a general solution. That is, once the documents are updated, the domain experts are required to intervene this manual work to construct new rules or new keywords. Another automatic approach to this KDD task used regular expressions to find patterns of words and then used a SVM to classify the papers [MMG03] but achieved a worse performance than the two previous manual intervention approaches (an F-score of 58% on determining whether to curate a paper based on the presence of experimental evidence).

Furthermore, other researchers explored biomedical literature classification in various ways. In [DMdBW03], Donaldson used the "bag-of-word" approach with an SVM classifier trained on the words in the MEDLINE abstracts to distinguish abstracts containing information on protein-protein interactions, prior to curating this information into their BIND database. Another investigation [CMS04] in this classifying area used extraneous web sources and thus unlike the other previous works in the KDD task, it does not involve expensive domain knowledge. Based on the following three widely-used web databases:

- MeSH: Medical Subject Headings, a collection of keywords for classifying articles.
- GenBank: a repository of gene structure data.
- GenPept: a repository of protein structure data.

the full text content was represented as a feature vector three times and each time the features are selected from one of the three external biological databases,

respectively. Then these three kinds of features are integrated together into a single one for each article, which named the WeBTC representation, and this newly combined text representation was used for the consequent construction of text classifier. However, when testing on the same KDD task, this new approach performed worse than the two previous manual intervention approaches ([RFLF+02] and [KOS+02]).

As we mentioned before, the bag-of-word approach has been popular for a long time mainly because it is easily applied to text from a large variety of sources, but it ignores the context information and may not result in the most discriminating features with various types from the fully marked up text. Therefore, researchers found it is necessary to do more work to determine what types of features are useful in addressing particular text mining tasks. The feature space available is large and includes a huge number of feature types including (but not limited to) words, concepts, headings, formatting, authors, references, links, and etc. For instance, using concepts as a basic unit instead of words has been shown to be practical in [Aro01] and useful in [RHA99] and [MW03]. Figure captions have also been shown to help in [RFLF+02]. In addition, a full text with XML markup may provide many more possible types of features than plain text and thus the potential feature space of XML full text need to be explored more further. Thus, to efficiently and effectively represent the content of text with more semantic information, more advanced Natural Language Processing (NLP) techniques are required.

Apart from the types of features, term weighting methods also have an impact on text representation and further on text classification. In the previous two experiments, we proposed and examined a new effective term weighting method

on two benchmark newswire domain data collections. We are also quite interested to validate whether this term weighting method is effective for biomedical articles.

In general, the full text of biomedical literature contains a wealth of scientific information important to users that may not be completely represented by abstracts and/or MeSH terms [YA03]. However, access to full text is still often limited due to copyright restrictions ([Her05], [Kra06]). Therefore, classifying biomedical articles with abstract and MeSH terms is still important at this time.

For the above purposes, in this experiment, we focus on the performance of term weighting methods when applied to two biomedical data corpora with abstract and MeSH terms.

5.3.2 Examples of Terms' Discriminating Power

To illustrate the terms' different discriminating power, we pick out several representative terms in the 18 Journals corpus and discuss their discriminating power in terms of different weighting factors. For each of the top 10 largest categories in the 18 Journals Corpus, Table 5.6 lists the top three terms with the largest feature selection measure (here using χ^2 metric).

We take the first two categories, namely, `chemistry` and `genetics`, for example. From each of the two categories, we select the top two terms with the largest feature selection measure χ^2. Table 5.7 and Table 5.8 list these four terms and their different term weights using the different term weighting factors with respect to the category `chemistry` and `genetics`, respectively.

Table 5.6: Statistics of the top 10 largest categories in the 18 Journal Collection and the top 3 terms with the largest feature selection metric χ^2

Category Name	#_doc	Top 3 Features with the biggest χ^2 measure		
`chemistry`	419	*crystal*	*linker*	*resolution*
`genetics`	1201	*sequences*	*genome*	*orthologous*
`metabolism`	1613	*ubiquitin*	*gtpase*	*dominant-negative*
`physiology`	672	*visual*	*eyes*	*stimulus*
`saccharomyces cerevisiae proteins`	98	*dsbs*	*chromatid*	*double-strand*
`signal transduction`	217	*endocannabinoid*	*transduction*	*proteoglycans*
`time factors`	103	*walks*	*cyanobacteria*	*skill*
`transcription factors`	295	*histones*	*phyb*	*proliferator-activated*
`transcription,genetic`	261	*polymerase*	*sigma*	*promoter*
`variation(genetics)`	95	*variation*	*snps*	*polymorphisms*

From literal meaning and the feature selection measure listed in Table 5.6, it is easy to find that the first two terms *crystal* and *linker* are closely related with category `chemistry` rather than with category `genetics`. On the other hand, the last two terms *sequence* and *genome* are more related with the content of category `genetics` than with the category `chemistry`. However, as the *idf* factor does not use the prior information of the category membership of training samples, each of these four terms has the same *idf* value irrespective of any of the two categories. On the contrary, by using rf factor, the first two terms *crystal* and *linker* are weighted more with respect to category `chemistry`. This indicates that these two terms are good discriminators which express the semantics of category `chemistry`. The similar observation can be found in the last two terms.

As for the other supervised term weighting factors, such as χ^2, *Odds Ratio*,

Table 5.7: Comparison of the weighting values of four terms with respect to category `chemistry`

Feature	Category: `chemistry`					
	idf	rf	χ^2	or	ig	gr
crystal	4.862	1.420	94.028	8.607	0.020	0.048
linker	8.110	1.485	1.414	8.847	0.002	0.006
sequence	3.404	1.042	3.440	0.613	0.001	0.002
genome	3.473	1.035	5.278	0.510	0.001	0.004

Table 5.8: Comparison of the weighting values of four terms with respect to category `genetics`

Feature	Category: `genetics`					
	idf	rf	χ^2	or	ig	gr
crystal	4.862	1.093	2.537	0.406	0.003	0.003
linker	8.110	1.087	0.033	0.392	0.0003	0.0004
sequence	3.404	1.468	47.473	2.698	0.014	0.0170
genome	3.473	1.531	62.457	3.180	0.018	0.023

information gain and *gain ratio*, they have been confirmed to perform worse than rf and even *idf* on two benchmark data collections in combination with kNN and SVM in the previous section. Thus, we only list the terms' weights value here.

5.3.3 Results and Discussion

Based on the results in the previous two experiments, we conduct experiments on four selected term weighting methods, i.e. *binary*, tf, $tf.idf$ and $tf.rf$, on two biomedical data collections.

Results on the Ohsumed Corpus

Figure 5.15 shows the micro-averaged breakeven results of the four term weighting methods on the Ohsumed corpus.

Table 5.9 summarizes the best results of four different schemes on the Ohsumed corpus, where the best scores are shown in bold font. Note that for each term weighting method the micro-averaged *precision* and micro-averaged *recall* score in this table are almost equal to each other, thus the micro-averaged F_1 value actually almost coincides with the micro-averaged *breakeven point*.

From the results in Figure 5.15 and Table 5.9, it is clear to find that $tf.rf$ performs consistently and significantly better than other term weighting methods as the feature set size increases and it achieves the best performance in all experiments in terms of micro-averaged breakeven point, i.e. 0.6805. On the other hand, *binary* performs consistently the worst among these four term weighting methods. The tf and $tf.idf$ perform comparable to each other and are better than *binary* all

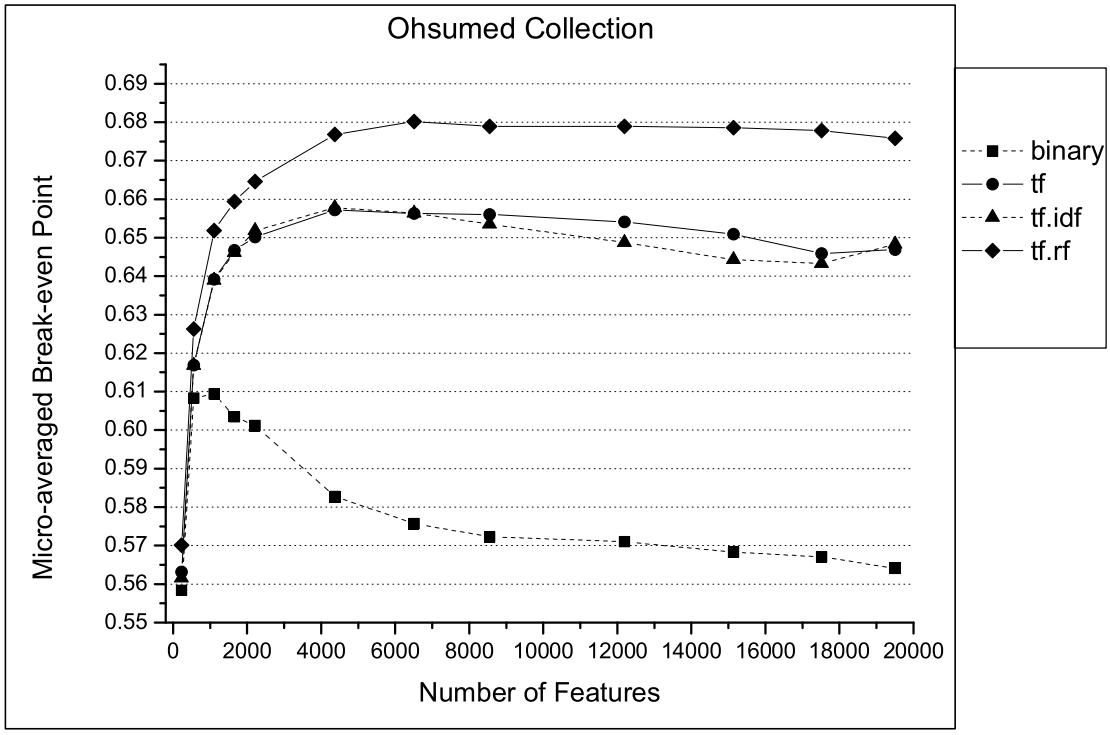

Figure 5.15: Micro-averaged break-even points results for the Ohsumed Data Collection by using four term weighting schemes at different numbers of features.

along. These results are not surprising and are consistent with the results on other benchmark data collections by using these term weighting methods and SVM in the two previous sections.

Since Joachims conducted experiments on the same corpus, it is easy to compare the two results. In [Joa98], the author used the classical $tf.idf$ to represent the text [1] and conducted experiments on the same data corpus using SVM in terms of micro-averaged breakeven point. Based on the comparison between the results

[1] After stemming and stop-word removal, the resulting training corpus has 15561 terms which occur in at least three documents. Moreover, the author used *information gain* measure to select the most discriminating features. Note that when using poly kernel functions of SVM, $d = 1$ means that the poly SVM actually is a linear SVM.

Table 5.9: The best performance of SVM with four term weighting schemes on the Ohsumed Corpus

Scheme	# of Features	micro-R	micro-P	micro-F1	macro-F1
binary	1111	0.6091	0.6097	0.6094	0.5757
tf	4363	0.6578	0.6566	0.6572	0.6335
$tf.idf$	4363	0.6567	0.6588	0.6578	0.6407
$tf.rf$	6511	**0.6810**	**0.6800**	**0.6805**	**0.6604**

reported in [Joa98] and our study, several observations are worth discussion.

- Our linear SVM gives a 68.05% micro-averaged breakeven point vs 60.7% for Joachims' linear SVM and 66.1% for his radial basis function with $\gamma = 0.8$. The observation that linear SVM outperforms other non-linear SVMs has already been supported by many researchers ([DPHS98], [YL99]) and our previous studies. There is no clear explanation for Joachims's different result.

- The performances of $tf.idf$ in two experiments are almost identical, i.e. 65.78% for our linear SVM and 66.1% for his radial basis function SVM. This difference is not significant.

- Last but not least, our proposed $tf.rf$ performs significantly the best among these four methods in our experiments. Moreover, it outperforms the $tf.idf$ in Joachims' experiments whether using linear SVM or non-linear SVMs.

Note that although our experiments use the same corpus and same evaluation measure as Joachims', there are minor differences in data preparation, such as the stemming or stop words lists for text preprocessing, and the different feature

selection measures. Joachims used *information gain* for feature selection, while we used χ^2 instead. This difference is not significant thus the comparison between the two experiments is still reasonable.

Results on 18 Journals Corpus

Since the breakeven point cannot be obtained on this data set, we use micro- and macro-averaged F_1 score instead, which are also widely-used in TC. We first conduct experiments on the top 10 largest categories of 18 Journals corpus. Figure 5.16 and Figure 5.17 show the micro- and macro-averaged F_1 scores of the four methods on the top 10 categories.

The results on the 18 Journals corpus is consistent with that on the Ohsumed corpus. Again our proposed $tf.rf$ is the best term weighting method among these four methods. Again *binary* is the worst method. However, tf performs better than $tf.idf$ and but still worse than $tf.rf$. These methods have shown consistent performance with respect to each other as the feature set size increases. There is an exception, i.e. the best performance of tf is comparable to $tf.rf$. This observation coincides with the conclusion we draw in previous two sections, i.e. tf has a rather good performance for TC.

Moreover, we explore the performance of these term weighting methods on the three subsets of 18 Journals corpus indicated in Table 4.2. Then the Figure 5.18 depicts the micro-averaged F_1 value on the three subsets of the 18 Journals corpus.

Although the absolute performance levels are not significant, their difference

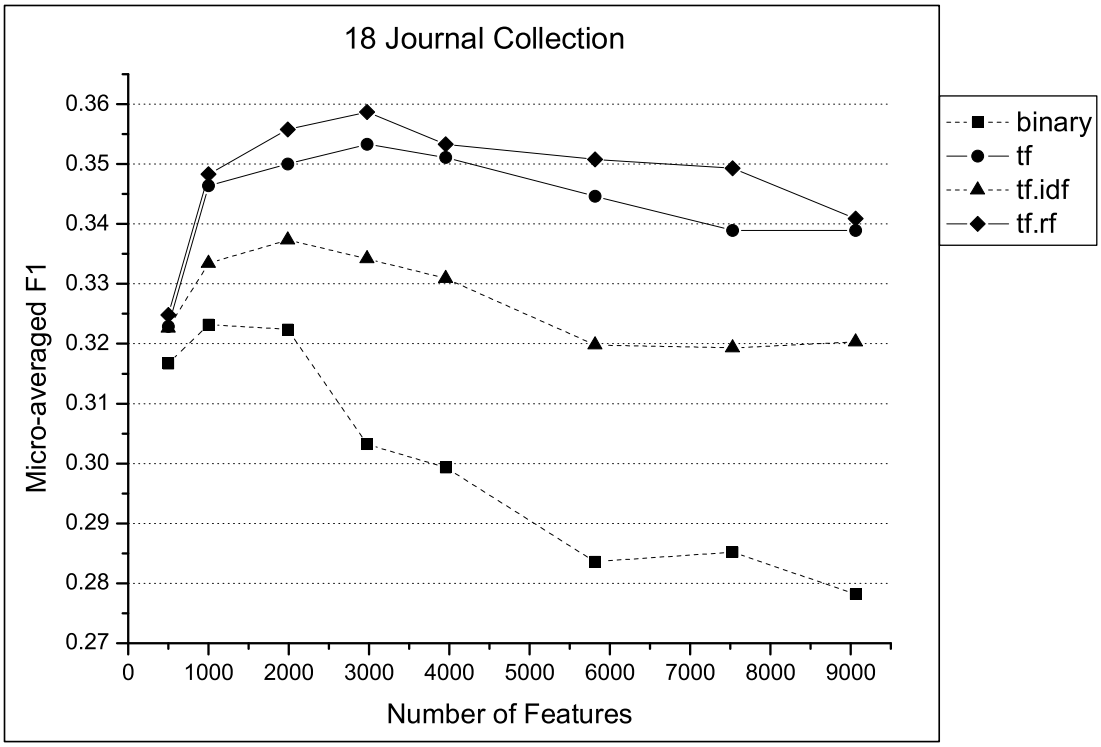

Figure 5.16: Micro-averaged F1 value of top 10 categories in 18 Journals Data Collection

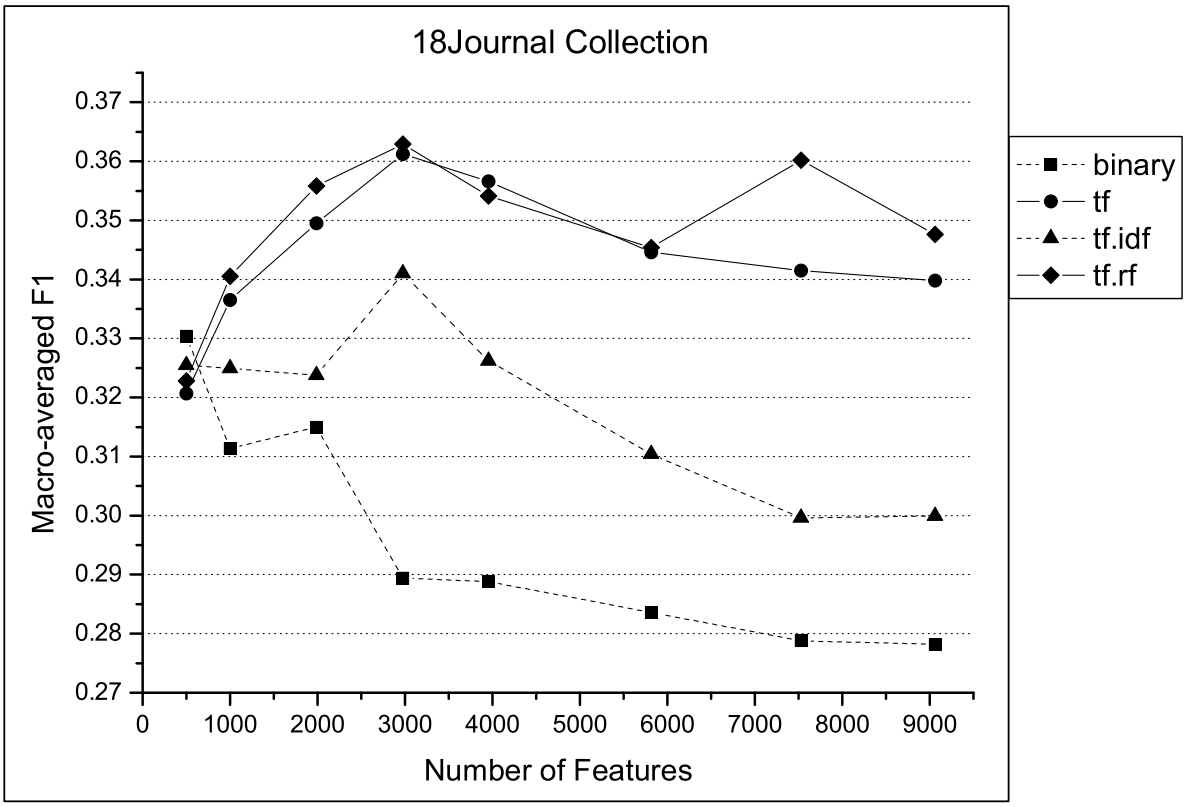

Figure 5.17: Macro-averaged F1 value of top 10 categories in 18 Journals Data Collection

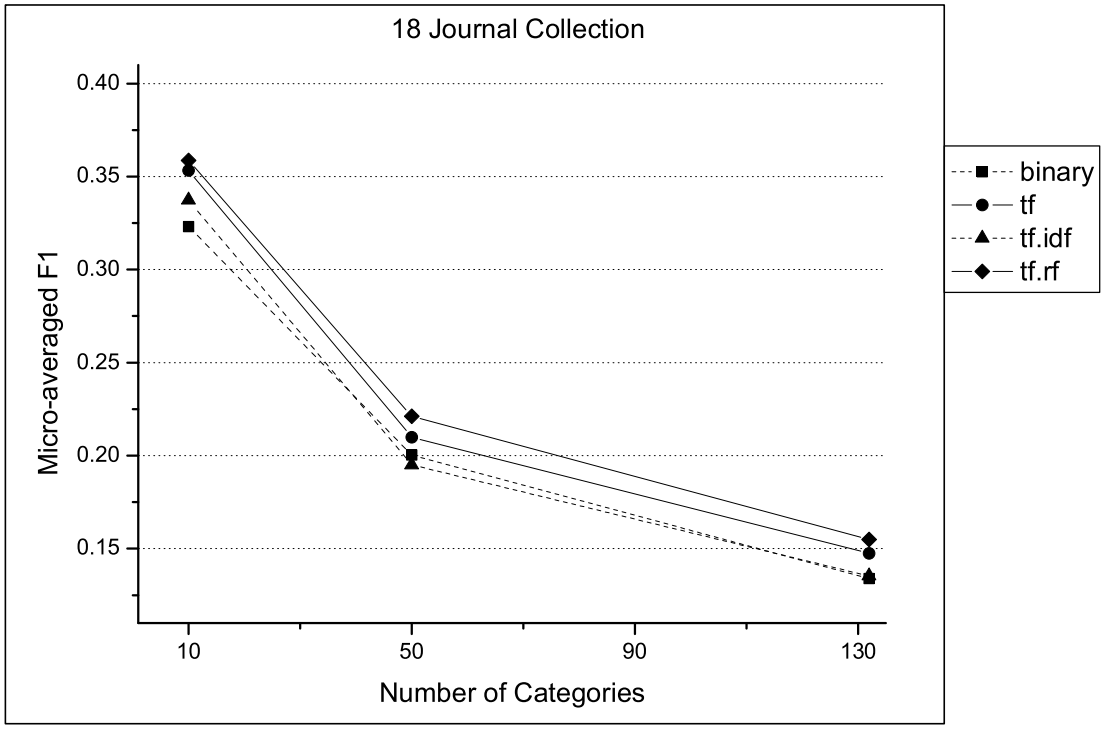

Figure 5.18: Micro-averaged F_1 value of different number of categories in 18 Journals Data Collection

is, since this is somehow indicative of the relative "hardness" of these subsets, and allows us to compare these term weighting methods on different subsets. The fact that the subset consisting of the top 10 categories turns out to be the easiest subset is quite obvious, given that its categories are the ones with the highest number of positive samples. With the increase of "hardness" of subsets, the micro-averaged F_1 performance is decreasing. However, the $tf.rf$ consistently has the best performance and tf also performs rather good. The performance of *binary* and $tf.idf$ is the worst all along and the difference between them is not significant as the number of categories increases.

5.3.4 Concluding Remarks

The experimental results on the published benchmark data corpus - Ohsumed corpus show that $tf.rf$ consistently achieve the best performance on categorization tasks, outperforming the other term weighting methods substantially and significantly. In addition, the comparison between our results and Joachims' results shows that $tf.rf$ can improve the classification performance on $tf.idf$.

Another experiment on the in-house manually-collected data corpus, i.e. 18 Journals corpus, shows that again $tf.rf$ outperforms other term weighting methods across different subsets of this corpus.

Therefore, these experimental results indicate that $tf.rf$ can improve the classification performance of biomedical text classification. This section gives evidence to support the effectiveness of our proposed $tf.rf$ once again.

We should point out that even though $tf.rf$ improves the classification perfor-

mance on the two data sets, the absolute levels of performance on the 18 Journals corpus is quite low. To significantly improve the performance of automatic biomedical literature classification, there are still a lot of challenges. One existing challenge is that the accuracy of classifying the documents based on MeSH keywords only is very low. The MeSH keywords are not definitely good indicators which can express the semantics of the texts. Some MeSH keywords themselves are quite general, such as *analysis*, *methods*, *growth and development* and so on. Others are too specific and quite close to each other, such as *rab5 gtp-binding proteins*, *rab gtp-binding proteins*, *rac gtp-binding proteins* and *ran gtp-binding proteins*, etc. On the contrary, since the texts of the Ohsumed are scientifically grouped under 23 diseases by domain experts rather than the MeSH keywords, the categorization task on it is definitely easier than that on the 18 Journals corpus.

Another attempt to significantly improve the classification performance is to try other complicated techniques for text representation, that is, more advanced techniques are necessary in order to capture and represent the semantics of text more precisely. One direction of this work is to use natural language processing (NLP) techniques. We believe more advanced NLP techniques and advanced ways of incorporating NLP output could further improve the performance of text classification, for example, high performance coreference resolution to normalize the protein names through different variations, nominal or pronominal expressions could generate more occurrences of the same protein names to facilitate the further text classification. This is beyond our work in this thesis but it is quite interesting for future work.

CHAPTER 6

CONTRIBUTIONS AND FUTURE DIRECTIONS

6.1 Contributions

For text categorization, although the inductive learning algorithm is an important part of enabling a text classifier system to achieve satisfactory performance, once given a learning algorithm, choosing the text representation becomes the central modelling tool of building text classifier. The bag-of-words approach remains the mainstay in many of today's text representation, but many other techniques from the term type or the term weighting aspect, have recently been studied with additional information. In this thesis, we focus on term weighting methods in an attempt to improve the text classifier performance. To address to three questions

raised at the beginning of this thesis, we gave a deep analysis of term weighting methods and conducted a series of experiments under various experimental conditions.

The main contribution of this thesis is to propose an effective supervised term weighting method $tf.rf$ to improve the performance of TC. It is the first and only one so far to discover the decisive effects of "a" and "c" only, rather than the more complicated variations of other schemes. Its effectiveness has been validated from a wide range of experiments, i.e. cross methods, cross classifiers, cross corpora and even cross application domains. Moreover, the significant advantage of this newly proposed method over the other methods is its robustness. That is, it consistently outperforms all other methods under different experimental conditions.

Another contribution of this thesis is to make an extensive comparative study of different term weighting methods under controlled conditions. No such work has been reported in the literature so far. This thesis is the first and only one to make such a comprehensive comparative study of these term weighting methods, including various unsupervised (traditional) term weighting methods and the state-of-the-art supervised weighting methods under different experimental conditions. As a result, this thesis leads to more convincing and comprehensive conclusions. For instance, it clarifies the original consideration that the supervised term weighting methods would be better than the unsupervised ones or otherwise.

Besides, this thesis makes other constructive contributions that have never been done by others. For instance, this thesis is the first to give a deep analysis of the terms' discriminating power for TC by using different methods from the qualitative and quantitative aspects. From this analysis, we gain insights into the

contributions of different distributions of documents in the corpus and a better understanding concerning the intuitive idea of the proposed $tf.rf$ method. In addition, this thesis is the first to investigate the relationships between term weighting methods and learning algorithms given different corpora and arrives at some interesting conclusions, for example the kNN algorithm favors *binary* representation while SVM does not; the popular $tf.idf$ performs well on uniform category distribution corpus rather than skewed corpus. This work will give a practical guidance on how to choose term weighting methods in terms of different learning algorithms.

6.2 Future Work

6.2.1 Extending Term Weighting Methods on Feature Types other than Words

This thesis has focused on the term weighting method. However, as described in Chapter 3, the term "term" in text categorization can be at different levels; hence, the term weighting method reported in this thesis can be extended to a variety of other feature types. Actually, the basic units of feature type for text representation could be syllabus, single word, phrases, or even complicated semantic and syntactic units such as word sense, concepts, and keywords. Although this thesis is based on the bag-of-word approach, other semantic feature types such as phrases, word senses, or concepts can be weighted by this weighting method as well. It would be quite interesting to see in the future work whether this $tf.rf$ method can improve the other feature types' discriminating power for text categorization.

On the other hand, besides these semantic units, other various types from the fully marked up text are also quite informative. For example, web pages may include many features like type font, font size, font color, headings, links, and so on. In addition, a full text with XML markup may provide many more possible types of features (such as authors, date, document type, publication source, references and etc which contain important information) than plain text and thus the potential feature space of XML full text need to be explored more further. Thus the feature space available may include a huge number of feature types including (but not limited to) semantic features (such as, words, concepts) and formatting features (such as headings, font, links, and etc). We expect these two kinds of feature types could be combined together to further improve the text classification performance.

This work is also useful for other types of text classification rather than dependent on the semantics only, such as text sentiment classification, text genre classification and etc.

6.2.2 Applying Term Weighting Methods to Other Text-related Applications

The $tf.rf$ term weighting method can be applied to tasks other than text categorization, e.g., information extraction, text summarization and so on. Since the term weighting is the most basic component of text preprocessing methods, we expect that our weighting method can be integrated into various text mining tasks, such as information retrieval, text summarization and so on. For example, in document summarization, Mori adopted *gain ratio* as a term weighting method to compare with a tf-based summarization system and the result showed that this

gr-based term weighting method is very effective in summarization [Mor02]. For the application of information extraction, the semantic information plays an essential role. Our $tf.rf$ method pays more attention to weight terms closer to the semantic of categories, which would be helpful for information extraction tasks. Similarly, to apply the $tf.rf$ method to other applications, we may need to extend the method to address corresponding issues.

APPENDIXES

I: 513 Stop Words List

a	about	above	across	after	afterwards
again	against	albeit	all	almost	alone
along	already	also	although	always	among
amongst	an	and	another	any	anybody
anyhow	anyone	anything	anywhere	arc	are
area	areas	around	as	ask	asked
asking	asks	at	away	b	back
backed	backing	backs	bc	be	became
because	become	becomes	becoming	been	before
beforehand	began	behind	being	beings	below
beside	besides	best	better	between	beyond
big	both	but	by	c	came
can	cannot	case	cases	certain	certainly
clear	clearly	co	come	could	d
did	differ	different	differently	do	does
done	down	downed	downing	downs	during
e	each	early	eg	either	else
elsewhere	end	ended	ending	ends	enough
etc	even	evenly	ever	every	everybody

everyone	everything	everywhere	except	f	face
faces	fact	facts	far	felt	few
find	finds	first	for	former	formerly
four	from	full	fully	further	further
furthered	furthering	furthers	g	gave	general
generally	get	gets	give	given	gives
go	going	good	goods	got	great
greater	greatest	group	grouped	grouping	groups
h	had	has	have	having	he
hence	her	here	hereafter	hereby	herein
hereupon	hers	herself	high	higher	highest
him	himself	his	how	however	i
ie	if	important	in	inc	indeed
interest	interested	interesting	interests	into	is
it	its	itself	j	just	k
keep	keeps	kind	knew	know	known
knows	l	large	largely	last	later
latest	latter	latterly	least	less	let
lets	like	likely	long	longer	longest
ltd	m	made	make	making	man
many	may	mean	while	member	members
men	might	more	moreover	most	mostly
mr	mrs	much	must	my	myself
n	namely	necessary	need	needed	needing
needs	neither	never	nevertheless	new	newer
newest	next	no	nobody	non	none

one	nor	not	nothing	now	nowhere
number	numbers	o	of	off	often
old	older	oldest	on	once	one
only	onto	open	opened	opening	opens
or	order	ordered	ordering	orders	other
others	otherwise	our	ours	ourselves	out
over	own	p	part	parted	parting
parts	per	perhaps	place	places	point
pointed	pointing	points	possible	present	presented
presenting	presents	problem	problems	put	puts
q	quite	r	rather	really	right
room	rooms	s	said	same	saw
say	says	second	seconds	see	seem
seemed	seeming	seems	sees	several	shall
she	should	show	showed	showing	shows
side	sides	since	small	smaller	smallest
so	some	somebody	somehow	someone	something
sometime	sometimes	somewhere	state	states	still
such	sure	t	take	taken	than
that	the	their	them	themselves	then
thence	there	thereafter	thereby	therefore	therein
thereupon	these	they	thing	things	think
thinks	this	those	though	thought	thoughts
three	through	throughout	thur	thus	to
today	together	too	took	toward	towards
turn	turned	turning	turns	two	u

under	until	up	upon	us	use
used	uses	v	very	via	w
want	wanted	wanting	wants	was	way
ways	we	well	wells	went	were
what	whatever	whatsoever	when	whence	whenever
whensoever	where	whereafter	whereas	whereat	whereby
wherefrom	wherein	whereinto	whereof	whereon	whereto
whereunto	whereupon	wherever	wherewith	whether	which
whichever	whichsoever	while	whilst	whither	who
whoever	whole	whom	whomever	whomsoever	whose
whosoever	why	will	with	within	without
work	worked	working	works	would	x
y	year	years	yet	you	young
younger	youngest	your	yours	yourself	yourselves
z	no	me			

II: Name List of Categories in Reuters and 20Newsgroups Corpora

Category ID	Reuters	20 Newsgroups
0	acq	comp.graphics
1	corn	comp.os.ms-windows.misc
2	crude	comp.sys.ibm.pc.hardware
3	earn	comp.sys.mac.hardware
4	gain	comp.windows.x
5	interest	misc.forsale
6	money-fx	rec.autos
7	ship	rec.motorcycles
8	trade	rec.sport.baseball
9	wheat	rec.sport.hockey
10		alt.atheism
11		sci.crypt
12		sci.electronics
13		sci.med
14		sci.space
15		soc.religion.christian
16		talk.politics.guns
17		talk.politics.mideast
18		talk.politics.misc
19		talk.religion.misc

BIBLIOGRAPHY

[AC99] Gianni Amati and Fabio Crestani. Probabilistic learning for selective dissemination of information. *Inf. Process. Manage.*, 35(5):633–654, 1999.

[AKCS00] Ion Androutsopoulos, John Koutsias, Konstantinos V. Chandrinos, and Constantine D. Spyropoulos. An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160–167, New York, NY, USA, 2000. ACM Press.

[Aro01] A. Aronson. Effective mapping of biomedical text to the UMLS metathesaurus: The MetaMap program. In *Proc AMIA Symp*, pages 17–21, 2001.

[ASYM03] Lynette Hirschman Alexander S. Yeh and Alexander A. Morgan. Evaluation of text data mining for database curation: lessons learned from the kdd challenge cup. *Bioinformatics*, 19 Suppl. 1:i331–i339, 2003.

[Att98] Marco S. Salvi D. Attardi, G. Categorization by context. *Int. J. Univers. Comput. Sci.*, 4(9):719–737, 1998.

[BC92] Nicholas J. Belkin and W. Bruce Croft. Information filtering and information retrieval: two sides of the same coin? *Commun. ACM*, 35(12):29–38, 1992.

[BM98] L. Douglas Baker and Andrew Kachites McCallum. Distributional clustering of words for text classification. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 96–103, New York, NY, USA, 1998. ACM Press.

[BSAS94] Chris Buckley, Gerard Salton, James Allan, and Amit Singhal. Automatic query expansion using SMART: TREC 3. In *In Proc. of the Third Text REtrieval Conference*, pages 69–80, 1994.

[CDAR98] Soumen Chakrabarti, Byron Dom, Rakesh Agrawal, and Prabhakar Raghavan. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *VLDB Journal: Very Large Data Bases*, 7(3):163–178, 1998.

[CDI98] Soumen Chakrabarti, Byron Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 307–318, New York, NY, USA, 1998. ACM Press.

[CH98] William W. Cohen and Haym Hirsh. Joins that generalize: text classification using WHIRL. In Rakesh Agrawal, Paul E. Stolorz, and Gregory Piatetsky-Shapiro, editors, *Proceedings of KDD-98, 4th International Conference on Knowledge Discovery and Data Mining*, pages 169–173, New York, US, 1998. AAAI Press, Menlo Park, US.

[CL01] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at:http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[CLL03] Ji-Rong Wen Cong Li and Hang Li. Text classification using stochastic keyword generation. In *Proceedings of the 20th International Conference on Machine Learning (ICML-2003)*, 2003.

[CMS01] Maria Fernanda Caropreso, Stan Matwin, and Fabrizio Sebastiani. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. pages 78–102, 2001.

[CMS04] Francisco M. Couto, Bruno Martins, and Mario J. Silva. Classifying biological articles using web resources. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 111–115, New York, NY, USA, 2004. ACM Press.

[CS96] William W. Cohen and Yoram Singer. Context-sensitive learning methods for text categorization. In *SIGIR '96: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 307–315, New York, NY, USA, 1996. ACM Press.

[DC00] Susan T. Dumais and Hao Chen. Hierarchical classification of Web content. In Nicholas J. Belkin, Peter Ingwersen, and Mun-Kew Leong, editors, *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 256–263, Athens, GR, 2000. ACM Press, New York, US.

[Die98] Thomas G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Comput.*, 10(7):1895–1923, 1998.

[DL04] T. Rose F. Li D.D. Lewis, Y. Yang. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, pages 361–397, 2004.

[DMdBW03] I. Donaldson, J. Martin, B. de Bruijn, and C. Wolting. Prebind and textomy - mining the biomedical literature for proteinprotein interactions using a support vector machine. *BMC Bioinformatics*, 4(11), 2003.

[DP03] G. Rigoll D. Peng, U. Iurgel. A novel feature combination approach for spoken document classification with support vector machines. *MMIR'03: Multimedia Information Retrieval Workshop 2003 in conjunction with the 26th annual ACM SIGIR conference on Information Retrieval*, 2003.

[DPHS98] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155. ACM Press, 1998.

[DS03] Franca Debole and Fabrizio Sebastiani. Supervised term weighting for automated text categorization. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 784–788, New York, NY, USA, 2003. ACM Press.

[DTY+04] Zhi-Hong Deng, Shi-Wei Tang, Dong-Qing Yang, Ming Zhang, Li-Yu Li, and Kun Qing Xie. A comparative study on feature weight in text categorization. In *APWeb*, volume 3007, pages 588 – 597. Springer-Verlag Heidelberg, March 2004.

[EMR00] Gerard Escudero, Lluís Màrquez, and German Rigau. Boosting applied to word sense disambiguation. In *ECML '00: Proceedings of the 11th European Conference on Machine Learning*, pages 129–141, London, UK, 2000. Springer-Verlag.

[F̈99] J. Fürnkranz. Exploiting structural information for text classification on the www. In *In Proceedings of IDA-99, 3rd Symposium on Intelligent Data Analysis, number 1642 in Lecture Notes in Computer Science*, pages 487–497, Amsterdam, NL, 1999. Springer Verlag, Heidelberg, DE.

[FHK+91] Norbert Fuhr, Stephan Hartmann, Gerhard Knorz, Gerhard Lustig, Michael Schwantner, and Konstadinos Tzeras. AIR/X – a rule-based multistage indexing system for large subject fields. In André Lichnerowicz, editor, *Proceedings of RIAO-91, 3rd International Conference "Recherche d'Information Assistee par Ordinateur"*, pages 606–623, Barcelona, ES, 1991. Elsevier Science Publishers, Amsterdam, NL.

[FK84] N. Fuhr and G. E. Knorz. Retrieval test evaluation of a rule-based automatic indexing (air/phys). In *SIGIR '84: Proceedings of the 7th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–408, Swinton, UK, UK, 1984. British Computer Society.

[GLF99] Norbert Gövert, Mounia Lalmas, and Norbert Fuhr. A probabilistic description-oriented approach for categorizing web documents. In *CIKM*, pages 475–482, 1999.

[GY93] Church K. Gale, W. and D. Yarowsky. A method for disambiguating word senses in a large corpus. volume 26, pages 415–439, 1993.

[HANS90] P. J. Hayes, P. M. Andersen, I. B. Nirenburg, and L. M. Schmandt. Tcs: a shell for content-based text categorization. In *Proceedings of the sixth conference on Artificial intelligence applications*, pages 320–326, Piscataway, NJ, USA, 1990. IEEE Press.

[HDW99] V. Vapnik H. Drucker and D. Wu. Automatic text categorization and its applications to text retrieval. volume 10, pages 1048–1054, 1999.

[Her05] William Hersh. Trec 2004 genomics track final protocol; http://ir.ohsu.edu/genomics/2004protocol.html. accessed February 23, 2005.

[HKK01] Eui-Hong Han, George Karypis, and Vipin Kumar. Text categorization using weight adjusted k-nearest neighbor classification. In *PAKDD '01: Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 53–65, London, UK, 2001. Springer-Verlag.

[Hul94] David Hull. Improving text retrieval for the routing problem using latent semantic indexing. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 282–291, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[ILS+00] Raj D. Iyer, David D. Lewis, Robert E. Schapire, Yoram Singer, and Amit Singhal. Boosting for document routing. In *CIKM '00: Proceedings of the ninth international conference on Information and knowledge management*, pages 70–77, New York, NY, USA, 2000. ACM Press.

[Joa97] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 143–151, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[Joa98] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[Joa02] Thorsten Joachims. *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[Jon72] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[KB04] Mikaela Keller and Samy Bengio. Theme topic mixture model for document representation. *Learning Methods for Text Understanding and Mining*, 26 - 29 January 2004.

[KHZ00] Yu-Hwan Kim, Shang-Yoon Hahn, and Byoung-Tak Zhang. Text filtering by boosting naive bayes classifiers. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 168–175, New York, NY, USA, 2000. ACM Press.

[KMW00] S. Singh K. Myers, M. Kearns and M.A. Walker. A boosting approach to topic spotting on subdialogues. In *Proceedings of ICML-00, 17th International Conference on Machine Learning*, pages 655–662, Stanford, CA, 2000.

[KOS+02] S. Sathiya Keerthi, Chong Jin Ong, Keng Boon Siah, David B. L. Lim, Wei Chu, Min Shi, David S. Edwin, Rakesh Menon, Lixiang Shen, Jonathan Y. K. Lim, and Han Tong Loh. A machine learning approach for the curation of biomedical literature: Kdd cup 2002 (task 1). *SIGKDD Explor. Newsl.*, 4(2):93–94, 2002.

[KPKF03] Athanasios Kehagias, Vassilios Petridis, Vassilis G. Kaburlasos, and Pavlina Fragkou. A comparison of word- and sense-based text categorization using several classification algorithms. *J. Intell. Inf. Syst.*, 21(3):227–247, 2003.

[Kra06] Martin Krallinger. Biocreative - critical assessment for information extraction in biology; http://www.pdg.cnb.uam.es/biolink/biocreative.eval.html. 2006.

[Lar99] Leah S. Larkey. A patent search and classification system. In *DL '99: Proceedings of the fourth ACM conference on Digital libraries*, pages 179–187, New York, NY, USA, 1999. ACM Press.

[LC96] Leah S. Larkey and W. Bruce Croft. Combining classifiers in text categorization. In Hans-Peter Frei, Donna Harman, Peter Schäuble, and Ross Wilkinson, editors, *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 289–297, Zürich, CH, 1996. ACM Press, New York, US.

[LCZ+04] Tao Liu, Zheng Chen, Benyu Zhang, Wei-Ying Ma, and Gongyi Wu. Improving text classification using local latent semantic indexing. In *ICDM*, pages 162–169, 2004.

[Lew92] David D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 37–50, New York, NY, USA, 1992. ACM Press.

[Lew98] David D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *ECML '98: Proceedings of the 10th European Conference on Machine Learning*, pages 4–15, London, UK, 1998. Springer-Verlag.

[LJ98] Y.H. Li and A.K. Jain. Classification of text documents. *The Computer Journal*, 41(8):537–546, 1998.

[LK02] Edda Leopold and Jorg Kindermann. Text categorization with support vector machines. how to represent texts in input space? *Machine Learning*, 46(1-3):423 – 444, January - February - March 2002.

[LM02] Yong-Bae Lee and Sung Hyon Myaeng. Text genre classification with genre-revealing and subject-revealing features. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 145–150, New York, NY, USA, 2002. ACM Press.

[MBGMF04] Dunja Mladenic, Janez Brank, Marko Grobelnik, and Natasa Milic-Frayling. Feature selection using linear classifier weights: interaction with classification models. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 234–241, New York, NY, USA, 2004. ACM Press.

[Mit97] Tom Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[MMG03] Huma Lodhi Yong Zhang Moustafa M. Ghanem, Yike Guo. Automatic scientific text classification using local patterns: Kdd cup 2002 task 1. *ACM SIGKDD Explorations Newsletter*, 4(2):95–96, 2003.

[MN98] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *In AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[Mor02] Tatsunori Mori. Information gain ratio as term weight: the case of summarization of ir results. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[MRMN98] Andrew McCallum, Ronald Rosenfeld, Tom M. Mitchell, and Andrew Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 359–367, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[MW03] Yandell MD Majoros WH, Subramanian GM. Identification of key concepts in biomedical literature using a modified markov heuristic. *Bioinformatics*, 19(3), 2003.

[NGL97] Hwee Tou Ng, Wei Boon Goh, and Kok Leong Low. Feature selection, perceptron learning, and a usability case study for text categorization. In *SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 67–73, New York, NY, USA, 1997. ACM Press.

[OML00] Hyo-Jung Oh, Sung Hyon Myaeng, and Mann-Ho Lee. A practical hypertext catergorization method using links and incrementally available class information. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 264–271, New York, NY, USA, 2000. ACM Press.

[PA98] Ron Papka and James Allan. Document classification using multiword features. In *CIKM '98: Proceedings of the seventh international*

conference on Information and knowledge management, pages 124–131, New York, NY, USA, 1998. ACM Press.

[PLV02] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 79–86, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[Por80] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[RFLF+02] Yizhar Regev, Michal Finkelstein-Landau, Ronen Feldman, Maya Gorodetsky, Xin Zheng, Samuel Levy, Rosane Charlab, Charles Lawrence, Ross A. Lippert, Qing Zhang, and Hagit Shatkay. Rule-based extraction of experimental evidence in the biomedical domain: the kdd cup 2002 (task 1). *SIGKDD Explor. Newsl.*, 4(2):90–92, 2002.

[RH84] S. E. Robertson and P. Harding. Probabilistic automatic indexing by learning from human indexers. *Journal of Documentation*, 40(4):264–270, 1984.

[RHA99] T. Rindflesch, L. Hunter, and A. Aronson. Mining molecular binding terminology from biomedical text. In *Proceedings of the AMIA '99 Annual Symposium.*, 1999.

[Rob04] S. Robertson. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of Documentation*, 60(5):503–520, 2004.

[Rot98] Dan Roth. Learning to resolve natural language ambiguities: a unified approach. In *Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence*, pages 806–813, Madison, US, 1998. AAAI Press, Menlo Park, US.

[RS99] Miguel E. Ruiz and Padmini Srinivasan. Hierarchical neural networks for text categorization (poster abstract). In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 281–282, New York, NY, USA, 1999. ACM Press.

[RS02] M. Ruiz and P. Srinivasan. Hierarchical text categorization using neural networks. *Information Retrieval*, 5(1):87–118, 2002.

[Sar75] T Saracevic. Relevance: A review of and a framework for the thinking on the notion in information science. *Journal of the American Society for Information Science*, 26:321–343, 1975.

[SB88] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, 1988.

[Sch98] Hinrich Schutze. Automatic word sense discrimination. *Comput. Linguist.*, 24(1):97–123, 1998.

[SDH90] G W Furnas T K Landauer S Deerwester, S T Dumais and R Harshman. Indexing by latent semantic indexing. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[Seb02] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[SH00] C.L. Sable and V. Hatzivassiloglou. Text-based approaches for non-topical image categorization. *Internat. J. Dig. Libr.*, 3(3):261–275, 2000.

[SHP95] Hinrich Schutze, David A. Hull, and Jan O. Pedersen. A comparison of classifiers and document representations for the routing problem. In *Research and Development in Information Retrieval*, pages 229–237, 1995.

[SM99] Sam Scott and Stan Matwin. Feature engineering for text classification. In *ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning*, pages 379–388, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[SM05] Pascal Soucy and Guy W. Mineau. Beyond tfidf weighting for text categorization in the vector space model. In *IJCAI*, pages 1130–1135, 2005.

[SS00] R.E. Schapire and Y. Singer. Boostexter: a boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

[SSS98] Robert E. Schapire, Yoram Singer, and Amit Singhal. Boosting and rocchio applied to text filtering. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 215–223, New York, NY, USA, 1998. ACM Press.

[TH93] Kostas Tzeras and Stephan Hartmann. Automatic indexing based on bayesian inference networks. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 22–35, New York, NY, USA, 1993. ACM Press.

[TKSK00] D. R. Tauritz, J. N. Kok, and I. G. Sprinkhuizen-Kuyper. Adaptive information filtering using evolutionary computation. *Inf. Sci.*, 122(2-4):121–140, 2000.

[Tur99] P. Turney. Learning to extract keyphrases from text. *Technical Report ERB-1057, National Research Council, Institute for Information Technology*, 1999.

[WPW95] Erik D. Wiener, Jan O. Pedersen, and Andreas S. Weigend. A neural network approach to topic spotting. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 317–332, Las Vegas, US, 1995.

[WS81] Harry Wu and Gerard Salton. A comparison of search term weighting: term relevance vs. inverse document frequency. In *SIGIR '81: Proceedings of the 4th annual international ACM SIGIR conference on Information storage and retrieval*, pages 30–39, New York, NY, USA, 1981. ACM Press.

[WWP99] Andreas S. Weigend, Erik D. Wiener, and Jan O. Pedersen. Exploiting hierarchy in text categorization. *Information Retrieval*, 1(3):193–216, 1999.

[YA03] H. Yu and E. Agichtein. Extracting synonymous gene and protein terms from biological literature. *Bioinformatics*, pages 340–349, 2003.

[YC94] Yiming Yang and Christopher G. Chute. An example-based mapping method for text categorization and retrieval. *ACM Trans. Inf. Syst.*, 12(3):252–277, 1994.

[YL98] Kwok Leung Yu and Wai Lam. A new on-line learning algorithm for adaptive text filtering. In *CIKM '98: Proceedings of the seventh international conference on Information and knowledge management*, pages 156–160, New York, NY, USA, 1998. ACM Press.

[YL99] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49, New York, NY, USA, 1999. ACM Press.

[YP97] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[YSG02] Yiming Yang, Sean Slattery, and Rayid Ghani. A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems*, 18(2-3):219–241, 2002.

AUTHOR BIOGRAPHY

Man Lan is a PhD candidate in the Department of Computer Science, School of Computing, National University of Singapore and Institute for Infocomm Research, Singapore. Her research interests include text mining, knowledge discovery, machine learning and neural networks. She obtained a Bachelor of Engineering in Fine Chemical and minored in Computer Application Technology in 1996 and a Master of Science in Computer Science in 2002 from Shanghai Jiaotong University, China.

During her PhD candidature, her publications include:

- Man Lan, Chew Lim Tan, Jian Su. "A Term Investigation and Majority Voting for Protein Interaction Article Sub-task 1 (IAS)". *The Proceedings of the Second BioCreative Challenge Evaluation Workshop.* ISBN 84-933255-6-2. April 2007, Madrid, Spain.
- Man Lan, Chew Lim Tan, Jian Su and Hwee Boon Low. "Text Representations for Text Categorization: A Case Study in Biomedical Domain". *The Proceedings of International Joint Conference on Neural Networks (IJCNN2007).* August 2007, Orlando, Florida.

- Man Lan, Chew Lim Tan and Hwee Boon Low. Proposing a New Term Weighting Scheme for Text Categorization. In *the Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI2006)*. Page 763-768. ISBN: 978-1-57735-279-2. July 2006, Boston, Massachusetts.

- Man Lan, Sam Yuan Sung, Hwee Boon Low and Chew Lim Tan. A comparative study on term weighting schemes for text categorization. In *the Proceedings of International Joint Conference on Neural Network (IJCNN2005)*. Page 546-551. July 2005. Montreal, Canada.

- Man Lan, Chew Lim Tan, Hwee Boon Low and Sam Yuan Sung. A comprehensive comparative study on term weighting schemes for text categorization with support vector machines. In *the Proceedings of 14th International World Wide Web Conference (WWW2005)*. page 1032–1033. ISBN: 1-59593-051-5. May 2005. Chiba, Japan.

- Ji He, Man Lan, Chew Lim Tan, Sam Yuan Sung, and Hwee Boon Low. Initialization of Cluster Refinement Algorithms: A Review and Comparative Study. In *the Proceedings of International Joint Conference on Neural Networks (IJCNN2004)*. July 2004. Budapest, Hungary.