Feature Ranking for Text Classifiers

by

Masoud Makrehchi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2007

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners. I understand that my thesis may be made electronically available to the public.

Masoud Makrehchi

Abstract

Feature selection based on feature ranking has received much attention by researchers in the field of text classification. The major reasons are their scalability, ease of use, and fast computation. However, compared to the search-based feature selection methods such as wrappers and filters, they suffer from poor performance. This is linked to their major deficiencies, including: *(i)* feature ranking is problem-dependent; *(ii)* they ignore term dependencies, including redundancies and correlation; and *(iii)* they usually fail in unbalanced data.

While using feature ranking methods for dimensionality reduction, we should be aware of these drawbacks, which arise from the function of feature ranking methods. In this thesis, a set of solutions is proposed to handle the drawbacks of feature ranking and boost their performance. First, an evaluation framework called feature meta-ranking is proposed to evaluate ranking measures. The framework is based on a newly proposed Differential Filter Level Performance (DFLP) measure. It was proved that, in ideal cases, the performance of text classifier is a monotonic, non-decreasing function of the number of features. Then we theoretically and empirically validate the effectiveness of DFLP as a meta-ranking measure to evaluate and compare feature ranking methods. The meta-ranking framework is also examined by a stopword extraction problem. We use the framework to select appropriate feature ranking measure for building domain-specific stoplists. The proposed framework is evaluated by SVM and Rocchio text classifiers on six benchmark data. The meta-ranking method suggests that in searching for a proper feature ranking measure, the backward feature ranking is as important as the forward one.

Second, we show that the destructive effect of term redundancy gets worse as we decrease the feature ranking threshold. It implies that for aggressive feature selection, an effective redundancy reduction should be performed as well as feature ranking. An algorithm based on extracting term dependency links using an information theoretic inclusion index is proposed to detect and handle term dependencies. The dependency links are visualized by a tree structure called a term dependency tree. By grouping the nodes of the tree into two categories, including hub and link nodes, a heuristic algorithm is proposed to handle the term dependencies by merging or removing the link nodes. The proposed method of redundancy reduction is evaluated by SVM and Rocchio classifiers for four benchmark data sets. According to the results, redundancy reduction is more effective on weak classifiers since they are more sensitive to term redundancies. It also suggests that in those feature ranking methods which compact the information in a small number of

features, aggressive feature selection is not recommended.

Finally, to deal with class imbalance in feature level using ranking methods, a local feature ranking scheme called reverse discrimination approach is proposed. The proposed method is applied to a highly unbalanced social network discovery problem. In this case study, the problem of learning a social network is translated into a text classification problem using newly proposed actor and relationship modeling. Since social networks are usually sparse structures, the corresponding text classifiers become highly unbalanced. Experimental assessment of the reverse discrimination approach validates the effectiveness of the local feature ranking method to improve the classifier performance when dealing with unbalanced data. The application itself suggests a new approach to learn social structures from textual data.

Acknowledgements

A long journey is easier when you have good companions. When you need to shed light on your questions, they provide a candle with their knowledge. When you are lost, they offer a map from their experience to help find the true path. When you need hope during the long moments of disappointment and isolation, when you need a hand or you feel weary, when you need to talk or even complain .... for all of these, they offer nothing but their true friendship. At the end of this journey, it is very pleasant to be able to express my gratitude to all who lent a hand.

Most notably, I would like to express my sincere gratitude to my supervisor, Prof. Mohamed S. Kamel, for the opportunity of working with him, and for his support and gentle guidance during my research. My appreciation is also extended to my committee members Prof. Ali Ghorbani, Dr. Hamid Reza Tizhoosh, Dr. Otman Basir, and Dr. Chrysanne DiMarco, for their valuable comments and suggestions.

Throughout my research I was fortunate to meet and benefit from the support of many individuals at the Pattern Analysis and Machine Intelligence (PAMI) lab. My warmest thanks go to the fellow PAMI members, in particular Abbas Ahmadi, Hanan Ayad, Heidi Campbell, Rozita Dara, Khaled Hammouda, Rasha Kashef, Adams Wai-Kin Kong, Shady Shehata, Maryam Shokri, Yanmin Sun, and Noha Yousri. I would like also to thank former PAMI members Dr. Jan Bakus and Dr. Ali Tehrani. My special thanks go to my friend Dr. Ali Ghodsi for many valuable discussions.

Many thanks also go out to the Natural Sciences and Engineering Research Council of Canada (NSERC), and the LORNET Network for providing the financial support that made this work possible.

During this journey, the most tragic thing happened to me when I lost my beloved Father, who had always dreamed of this day. I would like to thank him and also my Mother for everything that they offered to me without any demand.

All impossible journeys become possible if you have a strong faith and a true love. Words fail me to express my appreciation to my wife Elham Ashari whose dedication, love, and persistent confidence in me, has taken the load off my shoulder. Many thanks to Elham for her unconditional love and unlimited support.

To my father

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Recently, text classification has become one of the fastest growing applications of machine learning and data mining [100]. There are many applications that use text classification techniques, such as natural language processing and information retrieval [55]. All of these applications use text classification techniques in dealing with natural language documents. Since text classification is a supervised learning process, many machine learning methods such as K-Nearest Neighbor, regression models, Naive Bayes classifier, decision trees, inductive rule learning, neural networks, and Support Vector Machines can be employed [3].

Most text classification algorithms use vector space model and bag-of-words representation to model textual documents. Some extensions of the vector space model have also been proposed to utilize the semantic and syntactic relationship between terms [99]. In the vector space model, every word or group of words (depending on whether one is working with a single word or a phrase) is called a term, which represents one dimension of the feature space. A positive number, reflecting the relevancy and significance, is assigned to each term. This number can be the frequency of the term in the document [120].

The major problem of text classification is its high dimensionality. A high dimensional feature space addresses a very large vocabulary that consists of all terms occurring, at least once, in the collection of documents. High dimensional feature space has a destructive influence on the performance of most text classifiers. Additionally, it increases the complexity of the system. To deal with high dimensionality and avoid its consequences, dimensionality reduction is strongly desired [95, 123].

One well-known approach for excluding a large number of irrelevant features is feature ranking [95, 26]. In this method, each feature is scored by a feature quality measure such as information gain, χ^2, or odds ratio. All features are then sorted based on their scores. For feature selection, a small number of the best features are kept and the rest are removed. However, this method has a serious disadvantage in that it ignores the redundancies among terms. This is because the ranking measures consider the terms individually. Another drawback of feature ranking-based feature selection is evident when it is applied to unbalanced data. Feature ranking methods are unable to fairly select features from all classes. Consequently, small classes tend to be ignored when training the classifier.

Due to the high dimensionality of text classification problems, computational efficiency and complexity reduction are very important issues. One strategy in dimensionality reduction is aggressive feature selection, in which the classification task is performed by very few features with minimum loss of performance and maximum reduction of complexity. In aggressive feature selection, more than 90% of non-discriminant, irrelevant, and non-informative features are removed [95,26]. Both aforementioned drawbacks have a more destructive impact on the classifier performance in an aggressive feature selection scheme.

1.2 Objectives

This thesis attempts to propose new techniques towards enhancing the performance of feature ranking-based feature selectors for text classifiers. Any improvement in the performance of feature ranking methods can be made after identifying and analyzing the drawbacks of these methods.

Feature ranking methods are data dependent such that we cannot suggest a general method best-suited for all data sets. The performance of ranking methods also depends on the choice of classifiers. Furthermore, in order to filter out the lower ranked features by ranking methods, there is a free parameter which is called the feature ranking threshold. Estimating this parameter is quite subjective and highly dependent on the data set and the learning model. Because of all aforementioned issues, feature ranking methods are highly problem-dependent. To deal with this problem dependency, we need an objective framework to study, analyze, and predict the behavior of ranking methods with respect to the data set characteristics and learning model.

The performance of the majority of learning models decreases as the class imbalance

increases. Employing feature ranking methods can worsen this situation. Another objective of this thesis is to investigate the reason of feature ranking failure when applied to unbalanced data, and propose some modifications to improve the performance in the case of high class skew.

A feature selection scheme is efficient if it can reduce redundancy as well as noise. The other goal is to propose a measure to evaluate the asymmetric dependency between every two features and learn the dependency links between them, which can be employed to extract feature redundancies.

1.3 Thesis Organization

This thesis consists of eight chapters. After the introduction, in Chapter 2, feature selection based on feature ranking for text classifiers is reviewed. The most popular feature ranking methods are discussed. In addition to existing measures, two new ranking measures including normalized information gain and category-document frequency are introduced.

In Chapter 3, the drawbacks of feature ranking methods with emphasis on text classifiers application, are detailed. We theoretically and experimentally show that feature ranking methods ignore term redundancy and fail when applied to a problem with class imbalance.

The proposed method, differential filter level performance, to evaluate feature ranking methods, is introduced in Chapter 4. Based on the proposed method, a meta-ranking framework is proposed to select the best-suited feature ranking method with respect to the data set and the choice of classifier.

Chapter 5 provides an application of the meta-ranking framework to extract domain-specific stopwords, which is an inverse task of feature selection. In this chapter, we extend the notion of filter level performance and try to estimate it approximately by data set characteristics such as global sparsity index. One interesting result is that if a given term ranking can perform well for selecting good features, it will not necessarily perform well when selecting poor features (stopwords).

In Chapter 6, the proposed approach to extracting term dependency is explained. We introduce inclusion index, a new information theoretic dependency measure to assess the dependencies between features. Using the term dependency tree, an algorithm is proposed to extract the redundant terms. We show that by using the proposed method, we can improve the classifier performance in the aggressive feature selection scheme.

In Chapter 7, we deal with another serious drawback of feature ranking methods. By introducing the reverse discrimination method for feature ranking, a local feature ranking method is proposed to re-balance the unbalanced data at the feature level. The problem of learning a social network from text data is employed to examine the proposed local feature ranking method.

Chapter 8 concludes the thesis with a discussion of the contributions made in addition to some suggestions for future work.

Appendix A provides some information about the data sets employed in this thesis and Appendix B includes the list of notations.

Chapter 2

Feature Selection based on Feature Ranking for Text Classifiers

2.1 Introduction

Text classification is one of the fast paced applications of machine learning and data mining [100]. There are many applications in natural language processing and information retrieval employing text classification techniques [55]. All these applications use the capability of text classification techniques in dealing with natural language documents. Since text classification is a supervised learning process, a lot of learning methods, for example, nearest neighbor, regression models, Bayesian approach, decision trees, inductive rule learning, neural networks, and Support Vector Machines (SVM), can be employed [3,124].

Most text classification algorithms use the vector space model to represent text documents. In this model, every word or group of words (terms), depending on working with a single word or a phrase, represents one dimension of the feature space. A positive number is assigned to each term. This number can be the frequency of the term in the text [120].

One major problem with text classifiers, especially when modeled by the vector space model, is high dimensionality, not only in the data but also in the feature space. A high dimensional feature space means a very large vocabulary that consists of all terms occurring at least once in the collection of documents. Although high dimensional feature space has destructive influences on text classification, its impact on increasing complexity is worse and expensive. To deal with performance degradation and complexity, feature selection is strongly desired [95,123].

Based on Heap's law, the size of vocabulary is non-linearly related to the size of data set, $O(n^{\beta})$, where n is the size of data set and β is a small number ($0.4 \leq \beta \leq 0.6$). On the other hand, according to Zipf's law, in a vocabulary, only few words are very frequent. For instance, two most frequent words in English can account for more than 10% of occurrences, while the majority of words occurs only once. Although both groups may carry some linguistic content and help to understand the meaning of a text (for example in natural language processing), for text classification purpose both groups should be removed because they do not contribute in the learning process and are not predictive enough. The words in the first group are called stopwords which can be dealt with as noise.

In this chapter, dimensionality reduction methods for text classifiers are briefly outlined. One class of dimensionality reduction methods is feature ranking-based feature selection techniques, which are reviewed in this chapter. Their major advantages, design parameters, and difficulties are also discussed. Finally, the most popular methods are introduced.

2.2 Document Representation

Document representation is an important issue in text mining. It can affect the text categorization process and its performance. Most researches in text categorization assume that a document consists of a Bag Of Words (BOW). In other words, in this representation, the smallest segment of information in textual data is word, not letter or sentence. In the literature, the atomic piece of information in textual data is called term, which may include a single word, multiple words or even a phrase. However, in this thesis, term is used as a single word.

BOW representation is not sufficient by itself to be employed in text categorization task as a vector of features. One problem is the size of documents, which are different using the BOW representation. One solution is to employ the Vector Space Model (VSM) to represent BOW of the documents. VSM, which is originally a representation model in information retrieval systems, has been first proposed by Salton [126].

In this model, every document is represented by a sequence of terms. Each term is represented either by a binary or a weighted format. There are different weighting schemas such as Term Frequency (TF), Term Frequency-Inverse Document Frequency (TF-IDF), and Term Frequency Constraint (TFC). In TFC the length of the document is considered in weighting scheme. The length of this vector is as big as the size of the vocabulary,

which is the set of all distinct word occurred in the data set. The j^{th} entry of the VSM represents the weight or score of the j^{th} term of the vocabulary in the document. This process is called term indexing.

In text classification, term indexing can be performed using local vocabulary or global vocabulary. The local vocabulary T_i includes all distinct terms occurring in the documents belonging to the category c_i. Obviously, in C-class problem, there are C local vocabulary. On the other hand, $\mathbf{T}$ is the global vocabulary and includes all distinct terms occurring in the collection.

VSM representation has some disadvantages, which includes ignoring four important facts of natural language text [77]:

- term dependencies and correlation
- text structure
- grammar and language model (which can be also considered as advantage once we are looking for a language independent framework)
- ordering of terms in the document

Let $\mathbb{D} \in \mathbb{R}^{n \times m}$ be the training text samples represented by a structure called document-term matrix. Each row such as $\mathbf{D_i}$ represents a document vector and every column such as $\mathbf{t_j}$ depicts the distribution of the term in the training samples. According to the vector space model, $\mathbf{D_i}$ is modeled as follows,

$$\mathbf{D_i} = (w_{i,1}t_1, w_{i,2}t_2, ..., w_{i,m}t_m) \tag{2.1}$$

where $m =$ the number of terms or the size of the vocabulary;
$w_{i,j} =$ the weight of the j^{th} term in the i^{th} document.

In the case of using binary weighting scheme, $w_{i,j} = \{0, 1\}$ due to the occurrence of t_j in $\mathbf{D_i}$. Otherwise, the weight consists of two weighting factors, local and global weighings. In local weighting the distribution of the term in the document is observed. For example, TF and TFC are two local weighting schemes. The global weighting scales the term into its distribution across the collection, for example inverse document frequency, term contribution, and all other term ranking measures such as information gain, odds ratio, and χ^2.

Since the length of documents in the collection could be different, in order to prevent the learning model to get biased to the very large documents, all terms in $\mathbf{D_j}$ are also

locally normalized across the document vector. Calculating and storing the normalized document vectors can considerably save further computation time.

$$\mathbf{D_i} = \frac{w_{i,1}t_1, w_{i,2}t_2, ..., w_{i,m}t_m}{\|\mathbf{D_i}\|} = \frac{w_{i,1}t_1, w_{i,2}t_2, ..., w_{i,m}t_m}{\sqrt{\sum_{j=1}^{m} w_{i,j}^2}} \tag{2.2}$$

where $\|\mathbf{D_i}\| =$ the norm or simply Euclidian length of the vector $\mathbf{D_i}$.

There have also been some efforts to use more sophisticated models that utilize semantic and syntactic relationship between terms [99, 105]. These two relationships are ignored by vector space model, which trusts completely in statistical behavior of terms within a document.

2.3 Dimensionality Reduction Methods

High dimensionality is known to be an intrinsic property of text classification problems, which not only increases the complexity of the problem but also degrades the performance of the system. In text categorization, dimensionality is directly linked to the notion of overfitting, which can harm the generalization capacity of a classifier [19, 128].

Three strong motivations for feature selection and dimensionality reduction for text classifiers are [76]:

- improving the scalability of the text classifier
- reducing the size of data base and saving computer resources
- improving the classifier performance through reducing noise, stopwords, and redundancy

Dimensionality reduction is either supervised or unsupervised depending on whether it is being used in a classification or a clustering task, respectively. Compared to unsupervised dimensionality reduction [75], supervised techniques are more successful and well-appreciated by researchers in text categorization [25, 33, 96, 28, 110].

Methods of dimensionality reduction are grouped into two categories: features extraction and feature selection [82]. Feature extraction uses all dimensions and measurement space to achieve a newly transformed space for compacting the feature space without eliminating any of them, while feature selection is mainly searching for a subset of

features among the total number of features based on one or more feature quality measures [60]. The quality measure can be either directly or indirectly linked to the classifier performance, which correspond to two families of feature selection approaches: wrapper and filter.

2.3.1 Feature Extraction

Feature extraction uses all dimensions and measurements to achieve a newly transformed space for compacting the feature space without eliminating any of them. The process of feature extraction is searching a mapping f, by which $\mathbf{Y} = f(\mathbf{X})$ where $\mathbf{X}$ and $\mathbf{Y}$ are former and new feature spaces, respectively. The function compacts the original feature space $\mathbf{X}$ such that $|\mathbf{Y}| \ll |\mathbf{X}|$. A good function is the one, which offers the best reconstruction of the data and provides the most predicting features. One advantage of feature transformation techniques are their solid mathematical background. The other is their promising results in many dimensionality reduction problems.

The transformation can be either linear or non-linear. Some examples of linear models are Principle Component Analysis (PCA), Singular Value Decomposition (SVD) also known as Latent Semantic Indexing (LSI) in information retrieval, Linear Discriminant Analysis (LDA), and Independent Component Analysis (ICA). For non-linear transformation, we can use techniques such as non-linear PCA and neural networks [32]. However, most feature extraction algorithms such as PCA fail with large data base of text mining. Another drawback is the computational time of feature extraction methods. For example calculating eigenvectors and eigenvalues are two time demanding and expensive tasks for high dimensional text problems [82].

2.3.2 Feature Selection

Feature selection reduces dimensionality by searching for a subset of features among the total number of features based on one or more performance indices. The performance indices can be either directly or indirectly linked to the classifier performance, which addresses two families of feature selection methods: wrappers and filters [47].

It is worthwhile to mention that for some applications such as genomic microarray analysis and text information retrieval, improving the classification performance and reducing the dimensionality are not the only reasons of feature selection. For example, in DNA microarray data analysis, biologists measure the gene expression including tens of

thousand genes in a tissue sample and try to explain how the genes relate to some deceases. For example, although some genes are strongly correlated to a particular type of cancer; biologists prefer to target a small subset of most relevant genes rather than conducting detail analysis and expensive experiments with a larger set of genes [31,58].

Wrapper Approach

In the wrapper approach, the corresponding performance index of a candidate feature subset is tied to the performance of the classifier. Therefore, the process of feature selection is not independent of the data mining task. In wrapper approaches, a classifier is employed to assign scores for features and feature selection depends on the choice of the classifier [44,58].

Since most feature selection methods have been proposed for non-textual data classification, the majority of them follow the wrapper strategy; not an efficient means of text classification. Two major drawbacks of wrappers are: *(i)* their huge computational time, which makes them inappropriate and unscalable for high dimensional text mining problems; and *(ii)* they are biased to the classification model. On the other hand, the great advantage of wrappers is their taking into account the correlations and dependencies between features. Because of this trait, the wrapper approach is called a multivariate (non-univariate) method, which is consistent with the multivariate function of text classifiers.

Filter Approach

To free the feature selection from the classification model, the classification performance can be indirectly estimated by some other quality measures. In the filter approach, prediction capacity of a classifier is translated into a merit function of selected features. This function reflects the quality of the selected features and can include information theory and information retrieval measures. The filter approach includes a wide range of search and ranking techniques. They employ almost the same search algorithms as wrappers. Filter approaches are implemented in a prepossessing step of classification, independent from the choice of the classification method [44,48,58].

Hybrid Methods

Recently some hybrid methods have also been proposed [14]. For instance, in [6], a method based on ranking features by weighing each feature on the separating hyper-

plane of the SVM classifier has been proposed. The other example is feature weighting based on generalization error bounds of SVM classifier with respect to the feature [89,51]. One drawback of these methods is their use of the expensive SVM classifier for feature selection. Another hybrid feature selection technique is proposed in [76], in which a linear classifier is employed to rank the features. This method is not a pure wrapper approach because the classifier model individually ranks every feature without considering the subset of features. It means that the method is still univariate without removing any possible term redundancy or correlation.

Other combinations of feature selection methods have also been considered to implement hybrid approach. For example, in [121], a feature ranking method (see section 2.4) such as information gain is followed by a recursive filter-based feature selection using Markov Blanket filter. The method is applied to genomic microarray data, which is similar to text problem in terms of high dimensional feature space. Also, [22] proposes the use of feature ranking techniques to provide more efficient search to wrapper-based feature selection. This method can also be extended to combination of feature ranking and a wrapper approach. In both cases, feature ranking is to substantially reduce the dimensionality, in which the major target is noise reduction. At the second part, filters or wrappers, which are recursive search-based algorithms, the main target is to remove redundancy and improving the classifier performance [58].

2.4 Feature Ranking

A class of filter approach feature selection algorithms is feature ranking methods, which is also know as Best Individual Features (BIF) method [40]. Feature ranking retains a certain number of features according to a ranking threshold; that is, those having the highest scores as measured by term relevance, predictive capacity, information content or quality index. Simply defined, feature ranking is sorting the features according to a feature quality index, feature weighting, or feature ranking measure, which reflects the relevance, information, and discriminating power of the features.

Compared to the wrapper approach and other filter techniques, feature ranking is less expensive, simpler, and more scalable. All these properties make feature ranking suitable for text classification problems. Similar to other filter methods, its design process is independent of the classifier model. It has been shown, however, that some ranking methods work better with particular classifiers [23,6]. As a rule, feature ranking is performed only on training data. Some researchers prefer to use test data to obtain term count, without

using the class labels [22].

2.4.1 Feature Ranking Design Factors

A feature ranking-based feature selector requires the use of a ranking measure and filtering out the low scored features. Designing a feature ranking-based feature selector needs to make decision about the following issues:

Feature Ranking Scope

The process of feature ranking is either local or global. In the local case, feature ranking is performed for each class individually, which implies employing a local vocabulary. Using a local vocabulary is potentially one of the solutions to overcome the performance degradation caused by class skew problem [23]. In global feature ranking, we are dealing with only one unified vocabulary associated with the training data set.

Feature Ranking Measure

Selecting a ranking measure must be performed by considering the classification model and the characteristics of the training data set. There is a link between feature ranking measures and classifiers. Some classifiers work better with a particular set of feature ranking measures; for example, the Naive Bayes classifier (NBC) works better with odds ratio, such that features with a higher rank in odds ratio measure are more influential in NBC [6, 76]. It can also be shown that the performance of feature ranking methods vary from one data set to another. For instance, in [76], it has been shown that the odds ratio feature ranking performs more successfully with moderately sparse data sets in the range of 10 to 20 terms per document vector, while the classifiers are NBC or SVM. Due to this correlation, one challenging problem is selecting the appropriate ranking measure for a particular data set. According to [76], sparsity can be a strong parameter by which one can decide about the feature ranking method. In this thesis, the main goal is to propose an objective framework for choosing the best feature ranking, one which may differ from one data set to the next.

Feature Ranking Threshold

The other crucial problem in feature ranking is to determine the appropriate threshold to filter out noise and irrelevant terms. This threshold represents the number of desired

features and reflects the complexity of the classifier. The ranking threshold can be applied to either the value of the ranking measures or the number of features.

The threshold is either a predetermined number of features k, which means that the k first features are to be selected, or a predefined value τ to select every feature whose score is higher than the threshold τ. The value of a threshold to be applied to sorted features depends on the training data set and the classification algorithm. For example, unlike a K-Nearest Neighbor (KNN) text classifier, which works with a short length feature vector, a NBC needs more features, and SVM is almost insensitive to the feature length [43].

In [26] a measure is proposed to find out whether a feature ranking improves the performance of a text classifier or not. This measure can be potentially employed for estimating the feature ranking threshold. The method is based on analyzing the distribution of features according to their information gain values. The introduced measure is called Outlier Count (OC), representing the number of features that their information gain are greater than $\mu + 3\sigma$:

$$OC = |\{t \in \mathbf{T} : IG(t) > \mu_{IG} + 3\sigma_{IG}\}| \tag{2.3}$$

where μ_{IG} = mean of information gain scores;
σ_{IG} = standard deviation of information gain scores.

According to [26], the speed of decline of sorted information gain across the features is more important than absolute value. OC is strongly correlated with the magnitude of improvement that can be achieved by feature selection.

The OC is also potentially applicable to investigate the effectiveness of *aggressive feature selection*, in which about 90% to 95% of features are removed [95]. According to [26], with lower OC, aggressive feature selection is highly recommended; but in higher OCs, however, aggressive feature selection can harm the classifier performance. In such cases, a very smooth feature selection is preferred. Figure (2.1) demonstrates the sorted, normalized information gain of the best 500 features for the six data sets employed in this thesis. Among the data sets, the 20 Newsgroups, WebKB, and Industry Sectors have quite sharp decline and consequently low OC compared to the rest of the data sets. In other words, the ranking threshold for these data sets should be low such that an aggressive scheme can be applied. On the other hand, for data sets whose sorted information gain have smooth decline, for example the Reuters, aggressive feature selection is not recommended and the threshold should be large number.

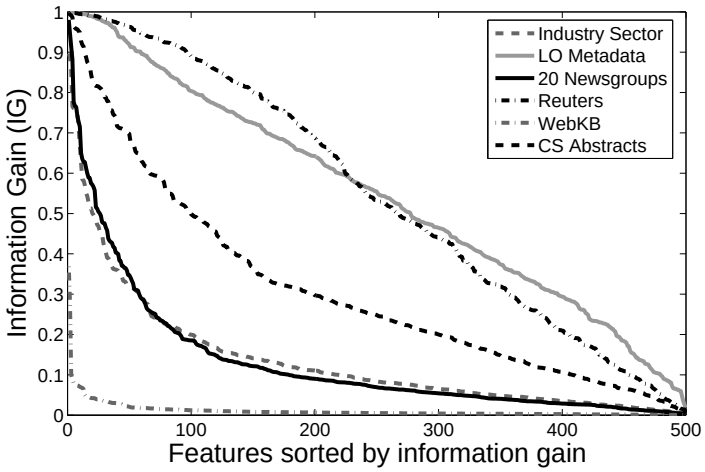


Figure 2.1: Sorted information gain of the best 500 features for the six data sets.

2.4.2 Evaluating Feature Ranking Methods

A number of ranking measures have been perviously proposed. In [123,95,77,6], the measures have been comprehensively compared with each other. According to their findings, information gain and χ^2 offer better results. Although these are comprehensive studies, there is no objectively formulated merit function to find the best feature ranking measure for a given data set. The problem is more complex when considering the fact that the functionality of feature ranking highly depends on the type of classifier, and the characteristics of the training data, such as sparsity and imbalance [6,76].

Various ranking measures adopted from information theory and/or information retrieval have been already introduced. Examples include information gain, mutual information, term entropy, χ^2, (inverse) document frequency, and odds ratio. Among the list of measures, a few usually offer a good ranking result compared to the others. Since, the functionality of the feature ranking for feature selection depends on other factors, including the type of classifier and training data characteristics, proposing a general feature ranking measure as the best method for all conditions is not possible. In other words, a framework is needed to indicate the best ranking measure according to the characteristics of the training data. In this thesis, a model is formulated to select the best feature ranking measure among a set of candidate measures.

2.5 Feature Ranking Measures

All feature ranking methods are based on the following three steps:

- estimating a feature ranking measure as a merit index for each feature;
- sorting (weighting) all features in decreasing order based on the ranking measures;
- applying a threshold to the sorted feature list.

Feature ranking measures can provide each feature either one value per category (C scores for each term, where C is the number of categories), or one single value per collection (one value for each term) [23,77], which are called local and global feature rankings, respectively. Local feature ranking, which has C local feature vectors, one for each category, produces a table called category-term ranking matrix $\mathbb{F} \in \mathbb{R}^{C \times m}$. Every entry $\phi(t_j, c_k)$ of the matrix $\mathbb{F}$ is called a category-term weight representing the relevance of the term j to the class k. Since class information is involved in the ranking process, this group of ranking methods can be used only in supervised learning problems. Examples are mutual information, χ^2, and odds ratio. The k^{th} row of $\mathbb{F}$ is the local feature ranking for the k^{th} category, and the j^{th} column is the distribution of the term j over the categories.

The popular approach is to use a global feature vector, which requires a unified ranking method, producing one score for each term per collection. The result of global ranking is the ranking vector $\mathbf{\Phi} \in \mathbb{R}^{1 \times m}$. Global ranking is performed either by an unsupervised measure such as inverse document frequency and term contribution, or a supervised measure such as information gain.

The category-term ranking matrix can be transformed into a ranking vector and employed using a global vocabulary. In order to create a ranking vector from a matrix, all weights assigned to a term (the j^{th} column of the category-term ranking matrix) across different classes are aggregated using an aggregation function:

$$\Gamma : \mathbb{F} \in \mathbb{R}^{C \times m} \rightarrow \mathbf{\Phi} \in \mathbb{R}^{1 \times m} \tag{2.4}$$

where Γ = the aggregation operator and might be any function such as maximum or mean.

To implement most feature ranking methods, it is essential to estimate various probability distributions such as joint and conditional probabilities. Let T_p, F_n, F_p be document frequency measures with different conditions as follows:

Table 2.1: Class-Term contingency table.

	t_j	$\bar{t}_j$
c_k	T_p	F_n
$\bar{c_k}$	F_p	T_n

- True-Positive (T_p): number of documents in c_k containing t_j where c_k is the k^{th} class and t_j is the j^{th} feature (term),
- False-Negative (F_n): number of documents in c_k without t_j,
- False-Positive (F_p): number of documents not in c_k containing t_j,
- True-Negative (T_n): number of documents not in c_k without t_j.

Most feature ranking measures can be grouped into two families:

- information theory measures, for example, information gain, mutual information, entropy, χ^2, Bi-Normal Separation (BNS) [22], and correlation coefficient ranking [31];
- information retrieval measures, such as document frequency-based measures, including document frequency and its variants, odds ratio, and F-measure based ranking.

According to [77], there is also a group of machine learning-based measures, which have good recall but poor precision such as Laplace measure, Difference measure, and Impurity level. In addition, they do not perform well in aggressive feature reduction tasks. Due to these drawbacks, they are not considered in this thesis.

2.5.1 Information-Theoretic Feature Ranking Measures

Mutual Information

Mutual information is a measure of statistical information that is shared between two probability distributions. Based on the definition in [11], mutual information $I(x;y)$ is computed by the relative entropy of a joint probability distribution such as $P(x,y)$, and the product of the marginal probability distributions $P(x)$ and $P(y)$ as follows:

$$I(x;y) = D(P(x,y)||P(x)P(y)) = \sum_x \sum_y P(x,y) log \frac{P(x,y)}{P(x)P(y)} \tag{2.5}$$

Mutual information has been applied in text mining and information retrieval applications including word association [9] and feature selection [117, 123]. Mutual information can be computed either as a single term or as a multiple term measure.

The single-term mutual information measures information shared between term t_j and category c_k as follows:

$$I(t_i; c_k) = P(t_i, c_k) log \frac{P(t_i, c_k)}{P(t_i)P(c_k)} \tag{2.6}$$

and using Table (2.1)

$$I(t_i; c_k) = \frac{T_p}{n} \log \frac{n.T_p}{(T_p + F_p).(T_p + F_n)}, \tag{2.7}$$

where $n =$ the total number of documents in the training data.

In multiple-term mutual information, information shared between two or more terms occurring in a particular category is considered. In this case, the mutual information is viewed as the entropy of co-occurrence of any two terms, when a category is given. Although this is one of the most promising solutions for redundancy reduction [65], estimating mutual information of higher orders (more than two) is very expensive [1]. Despite the multiple-term measure, the single-term mutual information provides poor results [95, 123] due to its tendency to select rare terms [39].

Information Gain (IG)

Information Gain (IG) is one of the improved variants of mutual information. IG not only measures information shared between a term and a class, but also evaluates the information shared between a term and the rest of categories. IG measures the discriminating capacity of a term. According to [123], of all the different ranking measures, IG and χ^2 offer better results. Based on [6], IG can have a high precision and low recall result, which is more desirable for classification problems. The local information gain of the term t_j against the class c_k contains three components as follows:

$$\begin{aligned} IG(t_j, c_k) &= -ig_1 + P(t_j).ig_2 + P(\bar{t_j}).ig_3 \\ ig_1 &= P(c_k). \log P(c_k) \\ ig_2 &= P(c_k|t_j). \log P(c_k|t_j) \\ ig_3 &= P(c_k|\bar{t_j}). \log P(c_k|\bar{t_j}) \end{aligned} \tag{2.8}$$

where $P(c_k) =$ the probability of a document belongs to the class c_k;
$P(t_j) =$ the probability of a document contains the term t_j;

$P(c_k|t_j) =$ the conditional probability of c_k given term t_j.

In practice, all these probabilities can be estimated using Table (2.1) as follows:

$$P(c_k) = \frac{T_p + F_n}{T_p + F_n + F_p + T_n} = \frac{T_p + F_n}{n} \tag{2.9}$$
$$P(c_k|t_j) = \frac{P(t_j, c_k)}{P(t_j)} = \frac{T_p}{T_p + F_p}, \;\; P(t_j) = \frac{T_p + F_p}{n}$$
$$P(c_k|\bar{t_j}) = \frac{P(\bar{t_j}, c_k)}{P(\bar{t_j})} = \frac{F_n}{T_n + F_n}, \;\; P(\bar{t_j}) = \frac{T_n + F_n}{n}$$

The global information gain of a term across all categories in the training data is estimated by weighted average of (2.8) for all classes.

$$IG(t_j) = \sum_{k=1}^{C} ig_1 + \frac{T_p + F_p}{n} \sum_{k=1}^{C} ig_2 + \frac{F_n + T_n}{n} \sum_{k=1}^{C} ig_3 \tag{2.10}$$

Normalized Information Gain (NIG)

Using the entropy of term t_j, Normalized Information Gain (NIG) is introduced as follow:

$$NIG(t_j) = \frac{IG(t_j)}{H(t_j)} = \frac{IG(t_j)}{-P(t_j) \log P(t_j)} \tag{2.11}$$

Several formulations have been previously reported for normalizing information gain measure [78, 102]. In the proposed NIG, the aim is to normalize the average information shared between a feature and classes by the information provided by the feature itself. Figure (2.2) illustrates a diagram to explain the IG measure. Using this figure, we can estimate the following components of IG: $H(c_k) = \{2 + 3\}$, $H(c_k|t_j) = \{3\}$, and $H(c_k|\bar{t_j}) = \{2\}$.

$$IG(t_j, c_k) = \{2, 3\} - P(t_j).\{3\} - P(\bar{t_j}).\{2\} \tag{2.12}$$

The lower bound of IG is zero and it is when the feature is correlated with all classes (the term is a stopword). The upper bound of IG is estimated as follows: t_j and c_k are completely correlated, then $H(c_k|t_j) = \{3\} = 0$, $H(c_k|\bar{t_j}) = H(c_k)$, and $H(c_k) = H(t_j) = \{1\} = \{2\}$. As a result, we have: $IG(t_j, c_k) = \{1\}.(1 - P(\bar{t_j}))$. From (2.11):

$$NIG(t_j, c_k) = \frac{\{1\}.(1 - P(\bar{t_j}))}{\{1\}} = 1 - P(\bar{t_j}) = P(t_j) \tag{2.13}$$

It means that by the proposed NIG, we bounded IG measure to the weight of the term (*i.g.* its document frequency). $NIG(t_j)$ is the average of $NIG(t_j, c_k)$ over all classes. The

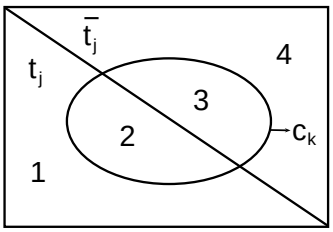


Figure 2.2: Information shared between a term and a class to estimate information gain measure.

above explanation is true for the average NIG, because $NIG(t_j)$ for a term is maximum when it is correlated with one class and uncorrelated with others.

The χ^2 Statistic

χ^2 has presented promising results in many text classification problems, such that it can usually outperform IG [95]. χ^2 is referred to the χ^2 distribution with one degree of freedom, which measures the lack of independence between a term and a category. An estimation of χ^2 is as follows:

$$\chi^2(t_i, c_k) = \frac{(T_p T_n - F_n F_p)^2}{(T_p + F_p)(F_n + T_n)(T_p + F_n)(F_p + T_n)} \tag{2.14}$$

According to [77], simplified χ^2 is calculated as

$$s\chi^2(t_j, c_k) = P(t_j, c_k).P(\bar{t}_j, \bar{c}_k) - P(\bar{t}_j, c_k).P(t_j, \bar{c}_k) = \frac{T_p T_n - F_n F_p}{n^2} \tag{2.15}$$

It should be noted that χ^2 values are normalized across a category, and can be compared as long as they are in the same class.

This measure yields unreliable results for very low frequent terms. However, by applying a conservative Low Document Frequency (Low-DF) term elimination, usually χ^2 offers the best results of the different ranking methods.

2.5.2 Information Retrieval Term Ranking Measures

Document Frequency (DF)

Document Frequency (DF) is a measure that reflects the contribution of a term in a whole data set. It is assumed that all the terms in the feature vector have the same importance. This assumption does not always work, because from a pragmatic point of view,

the importance of the terms across the collection and its categories varies. The second assumption in estimating DF is that all the terms are uniformly distributed over the categories [61]. In other words, the DF is biased to uniformly distributed terms across the categories, which means DF can be potentially employed in stopword reduction. Since DF ignores the labels and class information of the documents, it is an unsupervised ranking measure that is widely used in text clustering. Each term is assigned a measure, representing the number of documents, containing the term. Other variants of DF are term contribution, inverse document frequency, variance quality index [3], and local DF [54]. We also introduce category-document frequency measure, which is an extension of local DF.

Inverse Document Frequency (IDF) is an information retrieval ranking measure and widely used in removing high frequent terms, which are potentially considered as stopwords. For example, in [22] stopword removal is performed by removing all the terms with $DF \geq n/2$. This rule fails in the case of domain-specific stopword reduction and data sets with high class distribution imabalance [22].

IDF is calculated by different formulations such as

$$IDF(t_j) = \log \frac{n - n(t_j) + 0.5}{n(t_j) + 0.5} \tag{2.16}$$

where $n(t_j) =$ the number of documents containing the term t_j.

Category-Document Frequency (CDF)

IDF is an unsupervised feature ranking measure, where only the frequency of the term is observed across the collection. By involving the class information in the DF measure, other DF variants are defined. The class information of the documents can be involved in the DF measure by the joint probability of a term and a category $P(t_j, c_k)$. From the definition of $P(t_j, c_k)$, Local Document Frequency (LDF) and Document Frequency Variance (DFV) [54, 3] are calculated by

$$LDF(t_j) = \arg \max_{k=1}^{C} P(t_j, c_k) \tag{2.17}$$

and

$$DFV(t_j) = \frac{1}{C} \sum_{k=1}^{C} [P(t_j, c_k) - \sum_{k=1}^{C} P(t_j, c_k)]^2 \tag{2.18}$$

Although both LDF and DFV use the class information, from our experiments, they do not offer good results. For more efficient class information in DF measure, Category-Document Frequency (CDF) is proposed as follows:

$$CDF(t_j) = LDF(t_j) \log \frac{1+C}{\sum_{k=1}^{C} \mathcal{T}(t_j, c_k)}, \;\; \mathcal{T}_c(t_j, c_k) = \begin{cases} 0, & P(t_j, c_k) < \tau_d \\ 1, & P(t_j, c_k) \geq \tau_d \end{cases} \tag{2.19}$$

where $\tau_d =$ the category-term frequency threshold and $0 < \tau_d \leq 1$. Here $\tau_d = 0.1$.

Odds Ratio

Odds Ratio (OR), as a feature ranking measure, has been adopted from relevance ranking in information retrieval [92]. OR is estimated as follows:

$$odds(t_j|c_k) = \frac{P(t_j|c_k)}{1 - P(t_j|c_k)}, \;\; OR(t_j, c_k) = \frac{odds(t_j|c_k)}{odds(t_j|\bar{c}_k)} = \frac{T_p T_n}{F_p F_n} \tag{2.20}$$

In OR, the terms, which rarely occur in a category but never appear in other categories, attain higher ranks [76]. This property renders OR appropriate for keyword extraction. The risk is to select singletons (less frequent noise words) as relevant features [6]. Similar to the χ^2 measure, which is unreliable in low frequent terms, the solution is to apply a low-DF filter to reject rare and obscure words. According to [6], OR offers a high recall and low precision, which makes it insufficient for classification, but proper for retrieval tasks.

Ranking Measure based on F-measure

Let h be a single classification rule as follows:

$$h: \;\; if \;\; t_j \in \mathbf{D_i} \;\; \rightarrow \;\; \mathbf{D_i} \in c_k, \tag{2.21}$$

which implies an association between term j and class k, $t_j \rightarrow c_k$. The rule h is applied to each term in the vocabulary. By referring to Table (2.1), the Precision (P_c) and Recall (R_c) measures of h can be estimated as follows:

$$P_c(t_j \rightarrow c_k) = \frac{|retrieved \cap relevant|}{|retrieved|} = \frac{T_p}{T_p + F_p} \tag{2.22}$$

and

$$R_c(t_j \rightarrow c_k) = \frac{|retrieved \cap relevant|}{|relevant|} = \frac{T_p}{T_p + F_n} \tag{2.23}$$

where $P_c(t_j \rightarrow c_k) =$ the precision of h;
$R_c(t_j \rightarrow c_k) =$ the recall of h.

Eq. (2.22) and (2.23) are read as $P_c(t_j, c_k)$ and $R_c(t_j, c_k)$. With these two measures, the F-measure ranking method [23] is defined as follows:

$$F(t_j, c_k) = \frac{2P_c(t_j, c_k)R_c(t_j, c_k)}{P_c(t_j, c_k) + R_c(t_j, c_k)} = \frac{2T_p}{2T_p + F_n + F_p} \tag{2.24}$$

2.6 Estimating Classifier Performance

Text classifiers are usually evaluated by measures such as accuracy, Break-Even Point (BEP), precision and recall, or some combination such as F-measure [26]. The behavior of each of these evaluation measures highly depends on the data set characteristics, such as multiple or binary class, multiple or single label, imbalance (class skew), and homogeneity [23]. It should be noted that in a homogenous problem, the degree of difficulty is uniformly distributed over all categories, which means the difficulty of all classes is quite similar.

Classification accuracy (or error rate) is usually used as performance index in non-textual data classification. It may fail in the case of imbalance, which are more common in textual data. In order to deal with imbalance, one can adopt performance measures from information retrieval. Unlike text classification, in which imbalance is rarely considered, information retrieval systems always face with imbalance of class distribution. One simple example is when processing a user query. While the user looks for a single relevant (positive) record, the retrieval system may return many irrelevant (negative) records along with the positive result. This is the reason to adopt F-measure of information retrieval in text classification problems.

F-measure is defined based on precision and recall, which are two popular information retrieval measures. Precision is the fraction of retrieved documents that are relevant or $P_c = P(relevant|retrieved)$. Recall is the fraction of relevant documents that are retrieved or $R_c = P(retrieved|relevant)$. Recall is a non-decreasing function of the number of retrieved documents, while precision usually decreases. In performance evaluation of a classifier, given a query (a document) the output of the classifier (assigned label) is the retrieved document, which can be either relevant (true label) or irrelevant (false label). We can achieve very high recall while scoring very poor precision. For example, a system whose precision is very poor can retrieve all existing documents as results. This

simple example shows that precision is as important as recall. Due to this fact, a weighted harmonic mean, which is a conservative mean of precision and recall, is used:

$$F_{measure} = \frac{(\beta^2 + 1)P_c R_c}{\beta^2 P_c + R_c} \tag{2.25}$$

We usually use balanced F-measure, in which $\beta = 1$.

F-measure is usually employed for a binary classification problem. In the case of a multiple class problem, micro-averaged and macro-averaged F-measure is employed. Micro-averaged F-measure is a weighted average of F-measures of all categories by class distribution, while in macro-averaged F-measure, classes have no weights and all are similarly treated. While dealing with uniformly distributed classes, both averages are the same. Otherwise macro-average is less than micro-averaged F-measure [22, 23]. In other words, any difference between these two averages, could be a sign of class skew.

Most classification techniques do not obtain strong recall in the case of highly skewed classes because they focus on the accuracy measure. As a result, while evaluating unbalanced classes, the classification accuracy can be misleading. A better alternative is to use F-measure and, in the case of unbalanced multiple-class problems, macro-averaged F-measure is strongly recommended [22]. For this research, then, all classifier performances are measured by macro-averaged F-measure.

In order to estimate the classifier performance, different validation strategies have been introduced; for example, hold-out, leave-one-out, and K-fold cross validation [27,5]. The K-fold cross validation has been widely used in text classification [26, 23]. In most experiments in this thesis, a five-fold cross validation has been used for estimating the performance of the classifier, which is the macro-averaged F-measure. In this process, the collection (the data set) is divided into five subsets. The experiment is repeated five times. Each time we train the classifier with four subsets and leave the fifth one for the test phase. The average of five evaluations is the estimated performance.

2.7 Conclusion

In this chapter, feature ranking-based feature selection methods were detailed. Three elements of the feature ranking methods, including the scope of feature ranking, feature ranking measures, and ranking threshold, were also discussed. The ranking methods were categorized into two groups including information theoretic and information retrieval methods. We also introduced NIG which is an extension of IG and CDF as the supervised version of IDF.

Chapter 3

Feature Ranking Drawbacks

3.1 Introduction

Feature ranking methods are problem-dependent such that their behavior varies from one problem to the other. The problem means a combination of data set characteristics, for example data sparsity, and classification techniques. In previous section, it has been explained that the performance of a feature ranking measure varies based on the type of classifier. Furthermore, feature ranking methods, despite their scalability and lower cost algorithms, suffer from lower performance compared to the search-based feature selection approaches such as wrappers. The low performance of feature ranking techniques arises from two issues: *(i)* ignoring the correlation and dependency between terms, and realizing a univariate selection while the nature of text classifiers is multivariate [19, 80]; and *(ii)* failing in multi-class problems, especially with high class distribution imbalance [23].

3.2 Multivariate Characteristic of Text Classifiers

Feature selection based on ranking is a univariate approach, in which only one feature is considered to be retained or removed. In other words, feature ranking measures such as IG simply ignore the dependency and correlation between terms. The consequences can be low discriminating capacity and increasing redundancy. Let ϕ be a feature ranking measure such as IG, and t_1, t_2, t_3, and t_4 be four features (terms) from the vocabulary. Let us suppose that the features t_1 and t_2 have higher ranks than t_3 and t_4 as follows: $\{\phi(t_1) > \phi(t_2) > ... > \phi(t_3) > \phi(t_4) > ...\}$. In other words, it implies that t_1 and t_2 are

more relevant features than the others. Univariate feature selection approaches, such as feature ranking, fail in the following scenarios:

1. The feature ranking measure ϕ ranks t_1 and t_2 higher than t_3 and t_4, while in natural language texts, sometimes two individually irrelevant terms, such as t_3 and t_4 are jointly relevant. A well-known example is the phrase "To be or not to be", in which all terms are individually noise, but meaningful together.

2. By most feature ranking methods, t_1 and t_2 will be kept, while in textual data these two terms can be redundant as well as relevant, such as synonym terms [125].

As a result, neglecting the multivariate characteristic of text classifiers causes two serious problems: *(i)* ignoring correlation and dependency between features, and *(ii)* ignoring feature redundancies.

3.2.1 Term Correlation and Dependency

In spite of feature ranking, a text classifier behaves based on a combination of features returning to the correlation between them. By a simple test, we demonstrate the impact of the multivariate characteristic of text classifiers on their performance. Five ranking measures, including: IG, Max(χ^2), Mean(χ^2), IDF, and random feature ranking (RND) are applied to the six data sets, detailed in Appendix A. In addition to these five measures, the sixth ranking measure called Single Term Prediction (STP), which is defined based on the discriminant capability of every feature, is introduced. Let $J(t_j)$, $1 \leq j \leq m$ be the classifier performance using only feature t_j. Here J is the performance (*i.g.*, macro-averaged F-measure) of a Rocchio classifier [57]. After estimating J for all features, the terms are sorted based on their corresponding performance.

The classifier performance of the six ranking measures is estimated across all levels of filtering (ranking threshold) for the six data sets. Figure (3.1) depicts the classifier performance for 50% of the best features for the six ranking measures. It shows that STP ranking always performs very poorly as compared to the other methods, including random ranking. It suggests ignoring the correlation and dependency between terms is as destructive as noise in feature ranking.

3.2.2 Term Redundancy

All terms of the vocabulary with respect to their contribution to the categorization and retrieval processes can be grouped into four classes: *(i)* non-redundant and relevant; *(ii)*

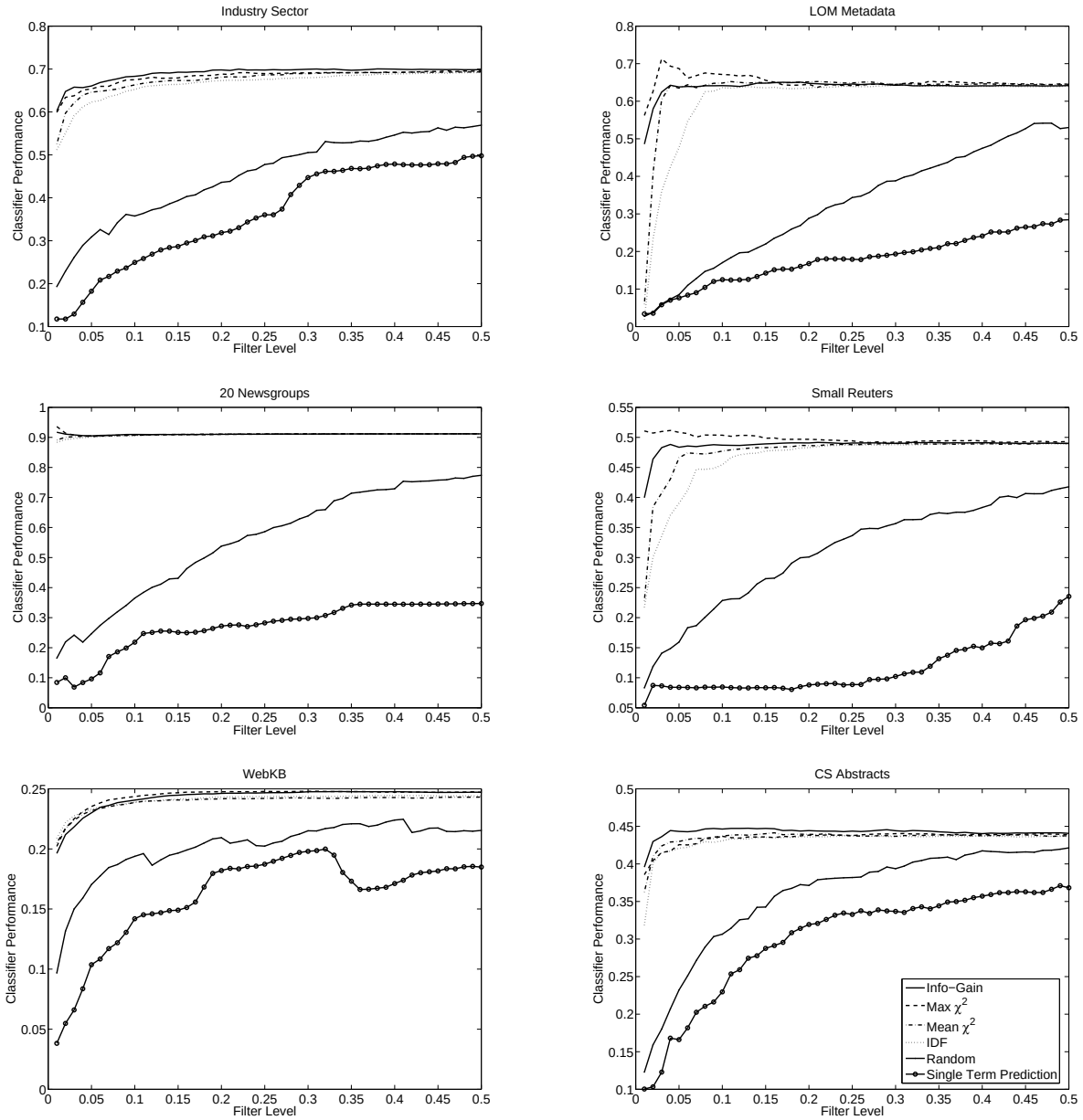


Figure 3.1: The impact of ignoring term correlation on the Rocchio classifier performance for the six data sets.

non-redundant and irrelevant; *(iii)* redundant and relevant; and *(iv)* redundant and irrelevant. In feature selection for text classifiers, we are only interested in the first group, which is non-redundant and relevant terms. Measuring the relevancy of the terms, by employing strong feature ranking methods, such as IG, is quite feasible. The difficulty is to extract term redundancies.

Redundancy is a kind of data dependency and correlation, which can be estimated by different measures, such as the Jaccard, Cosine, co-occurrence, and correlation coefficients [15, 122, 107]. Redundancy between two terms can be also measured by mutual information. If two terms have similar probability distributions on class labels, one of the terms might be considered as a redundant term such that removing it does not hurt the classifier performance. The problem is to find the possible redundancies and identify the redundant terms to be removed.

In this section, the result of an experiment illustrating the influence of redundancy on the classifier performance is presented. Two different text classifiers are employed: a Rocchio classifier which is sensitive to noise, and a SVM classifier with a linear kernel, as an optimum classifier which usually does not need feature selection and is commonly used as a text classifier. All the six data sets are employed in the experiment.

We show that adding redundancy, especially in the case of a very low number of features (aggressive feature selection), can degrade the performance. The testing process is as follows: Let $\mathbf{T}$ be the m sorted terms of the vocabulary according to a feature ranking measure ϕ, $\mathbf{T} = \{t_1, t_2, \ldots, t_m\}$, where t_1 is the best term and t_m the worst. The vectors $\mathbf{T_1}$, $\mathbf{T_2}$, and $\mathbf{T_3}$, three versions of $\mathbf{T}$ are generated by the following setups:

1. q best terms: $q < m$ best terms of the set $\mathbf{T}$ are selected such that $\mathbf{T_1} = \{t_i \in T | 1 \leq q\}$.

2. $q/2$ best terms + $q/2$ redundant terms: The vector $\mathbf{T_2}$ includes two segments. For the first segment, $q/2$ best terms of $\mathbf{T}$ are selected. The $q/2$ terms of the second segment are artificially generated by adding a very small amount of noise to each term of the first segment. The result is a set of redundant terms. Using this setup, the rate of redundancy ($\frac{\#redundant}{\#terms}$) is at least 50%.

$$\mathbf{T_2} = \mathbf{P_1} \cup \mathbf{P_2}, \;\; \mathbf{P_1} = \{t_j \in \mathbf{T} | 1 \leq j \leq \frac{q}{2}\}, \qquad (3.1)$$
$$\mathbf{P_2} = \{t'_j \in \mathbf{P_1} | t'_j = t_j + \sigma_n, 1 \leq j \leq \frac{q}{2}\}$$

where σ_n = added noise to each term;
P_1 = the set of the best and the most informative terms;

P_2 = the set of redundant terms.

The process of adding noise is as follows: Each term such as t_j is associated with a vector $\mathbf{t_j} = \{w_{1,j}, w_{2,j}, ..., w_{n,j}\}$, where $w_{i,j} \in \{0, 1\}, 1 \leq j \leq n$, is the weight of t_j in document $\mathbf{D_i}$. Here, a binary weighting scheme is used. We randomly add binary noise to each vector, which means inverting $\sigma_n = \frac{n}{10}$ weights in each vector.

3. $q/2$ best terms + $q/2$ noise: It is the same as the previous setup, except that the second part consists of noisy terms. Due to the use of feature ranking measures, $q/2$ last (worst) terms in $\mathbf{T}$ should be noisy and less informative. Therefore, we do not have to generate artificial noise.

$$\mathbf{T_3} = \mathbf{P_1} \cup \mathbf{P_2}, \;\; \mathbf{P_1} = \{t_j \in \mathbf{T} | 1 \leq j \leq \frac{q}{2}\}, \qquad (3.2)$$
$$\mathbf{P_2} = \{t_j \in \mathbf{T} | m + 1 - \frac{q}{2} \leq j \leq m\}$$

where P_1 = the set of the best and the most informative terms;
P_2 = noisy terms (worst and less informative).

We use five-fold cross validation for estimating the performance of classifiers. In this process, the collection (whole data set) is divided into five subsets. The experiment is repeated five times. Each time we train the classifier with four subsets and leave the fifth one for the test phase. The average of the five measures is the estimated classifier performance, which is the macro-average F-measure. All three feature vectors with $q = \{0.1, 0.2, ..., 0.5\} \times m$, are submitted to the SVM and Rocchio classifiers and the average of the performance of each classifier is calculated. We consider performance of the first classifier as the baseline, in which the original feature vector $\mathbf{T_1}$ without any artificially added noise or redundancy is employed. To compare the classification using noisy and redundant feature vectors, we compare their performance degradation from the baseline classifier. Let $\delta_2 = Mean_q[J(\mathbf{T_2}) - J(\mathbf{T_1})]$ where $J(\mathbf{T_2})$ and $J(\mathbf{T_1})$ are the performance of the classifier using term vector $\mathbf{T_2}$ and $\mathbf{T_1}$, respectively. We estimate $\delta_3 = Mean_q[J(\mathbf{T_3}) - J(\mathbf{T_1})]$ for the feature vector $\mathbf{T_3}$. Both δ_2 and δ_3 are always negative.

To compare the performance of $\mathbf{T_2}$ and $\mathbf{T_3}$, $\delta_J = (\delta_2 - \delta_3)$ is calculated for all feature ranking measures, including F-measure, IG, NIG, Max(χ^2), mean(χ^2), Max(OR), CDF, IDF, and random feature ranking, for the six data sets discussed in this thesis. A negative value for δ_J means that redundancy is more destructive than noise. Tables (3.1) and (3.2) illustrate the result of δ_J for both Rocchio and SVM classifiers. For all combinations of data sets and feature rankings, in both classifiers, the measure δ_J is a negative value,

Table 3.1: The estimation of δ_J for all feature rankings and the six data sets using Rocchio classifier.

Data Set	F-measure	IG	NIG	Max(χ^2)	Mean(χ^2)	Max(OR)	CDF	IDF	Random
Industry Sectors	-0.2282	-0.1550	-0.2233	-0.2122	-0.3806	-0.3724	-0.1882	-0.3676	-0.5070
LO Metadata	-0.4783	-0.5255	-0.4640	-0.4470	-0.5343	-0.5713	-0.5027	-0.5775	-0.5296
20 Newsgroups	-0.7099	-0.7717	-0.7691	-0.7114	-0.6403	-0.6460	-0.6788	-0.8289	-0.3228
Reuters	-0.2415	-0.3173	-0.3649	-0.2927	-0.3420	-0.3270	-0.2601	-0.3141	-0.3916
WebKB	-0.0880	-0.1183	-0.1252	-0.0536	-0.0341	-0.1077	-0.1125	-0.0276	-0.1336
CS Abstracts	-0.2362	-0.1109	-0.2306	-0.2679	-0.3937	-0.2532	-0.2531	-0.4094	-0.3537

Table 3.2: The estimation of δ_J for all feature rankings and the six data sets using SVM classifier.

Data Set	F-measure	IG	NIG	Max(χ^2)	Mean(χ^2)	Max(OR)	CDF	IDF	Random
Industry Sectors	-0.3117	-0.3393	-0.1843	-0.3433	-0.4806	-0.4317	-0.3345	-0.4666	-0.2943
LO Metadata	-0.4903	-0.6095	-0.6365	-0.5209	-0.6407	-0.5925	-0.4724	-0.6144	-0.5467
20 Newsgroups	–0.6973	-0.8114	-0.7799	-0.7075	-0.7613	-0.7082	-0.6577	-0.8526	-0.2335
Reuters	-0.2935	-0.3770	-0.3614	-0.2907	-0.4521	-0.3899	-0.2817	-0.4368	-0.2660
WebKB	-0.1323	-0.2251	-0.2371	-0.1978	-0.1948	-0.1659	-0.1586	-0.1312	-0.1774
CS Abstracts	-0.2702	-0.1749	-0.2597	-0.2303	-0.3570	-0.2890	-0.2497	-0.3606	-0.2875

which means the impact of the redundancy is always worse than that of noise. This fact is more clearly shown in Figures (3.2) and (3.3), in which the performance of classifiers using noisy feature vector $\mathbf{T_3}$ is better than that of using feature vector with redundancy ($\mathbf{T_2}$).

In conclusion, according to this experiment, redundant terms not only have no discriminating benefits for the classifier, but also reduce the chance that other less informative but non-redundant terms can contribute to the classification process. This experiment conformed findings in [58], which showed using feature correlation matrix that the fewer redundant features in a small subset of features includes, the more information it can offer for accurate prediction.

3.3 Class Distribution Imbalance

The majority of feature ranking methods fail when applied to multiple class problems with non-uniform class distributions (class skew). According to [23], feature ranking methods can select relevant features for easy classes, while being unable to learn difficult or small classes. They only pay more attention to the easy and large classes to compensate for their weakness against the difficult classes. By means of this trick, they improve their overall accuracy even though there is a large negative number of results for small and difficult classes. One potential approach to decrease the destructive impact of imbalance

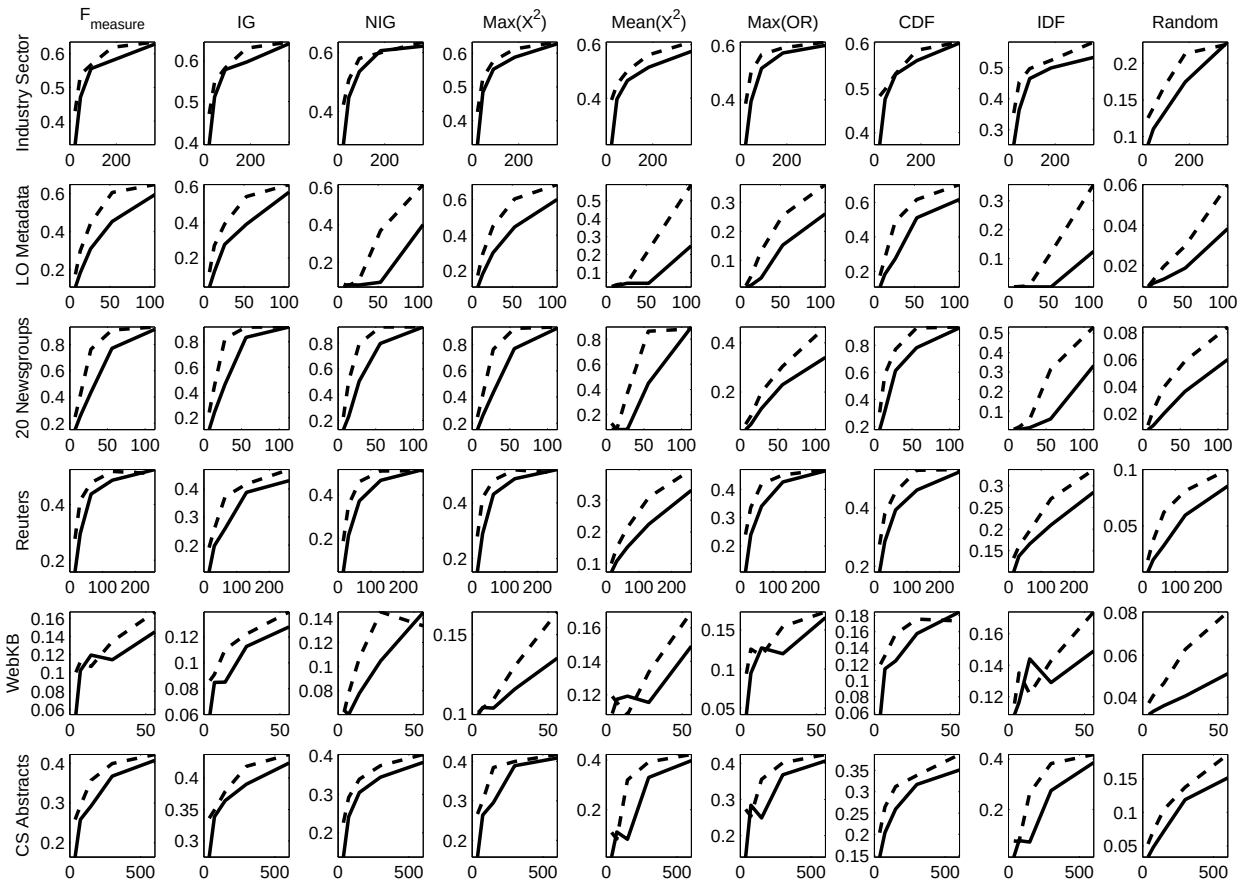


Figure 3.2: The effect of noise and redundancy on the Rocchio classifier performance with various feature ranking methods. The solid lines show the feature vectors with added redundant terms, and the dashed lines depict those with added noise terms.

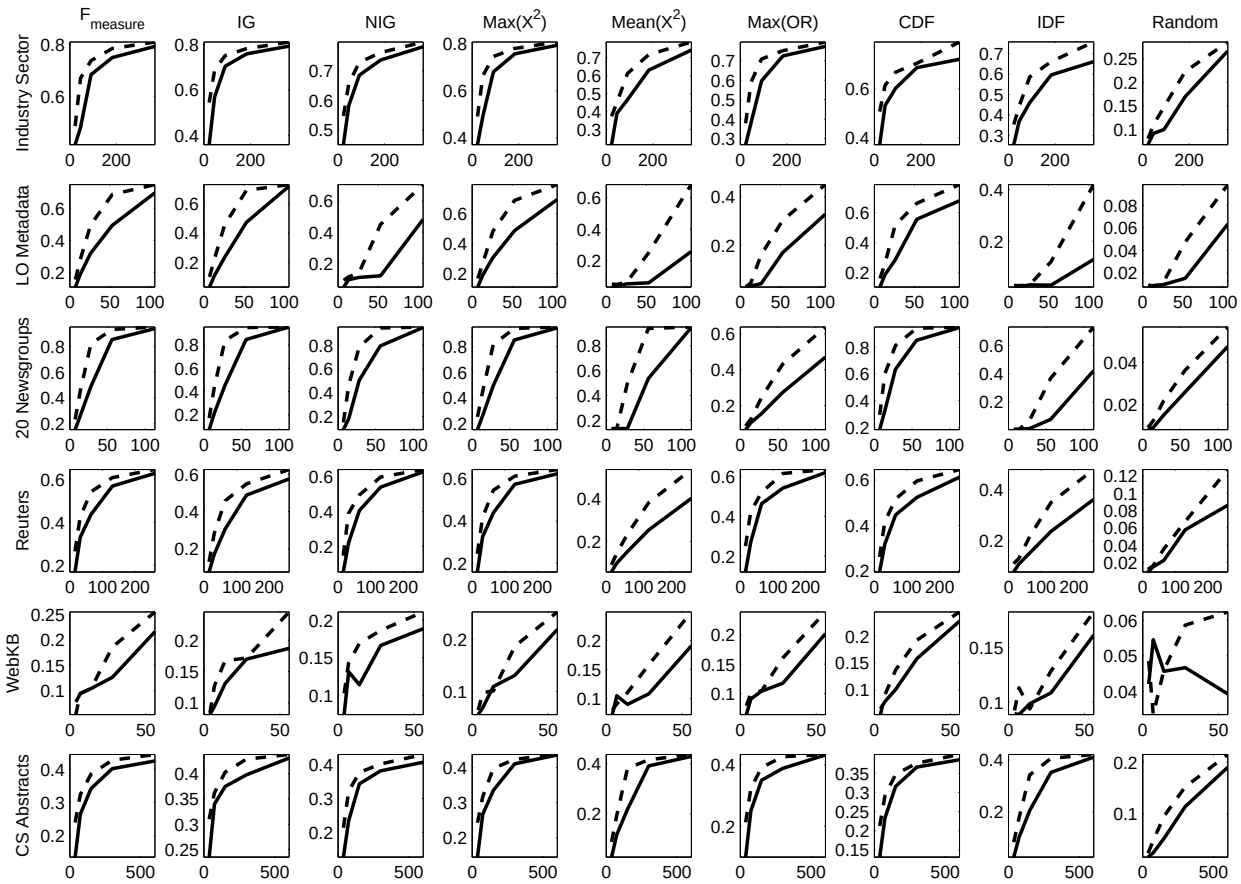


Figure 3.3: The effect of noise and redundancy on the SVM classifier performance with various feature ranking methods. The solid lines show the feature vectors with added redundant terms, and the dashed lines depict those with added noise terms.

is to make use of local feature ranking instead of a global scheme [23], which is discussed in Chapter 7.

Figure (3.4) illustrates the results of some experiments, which have been performed to investigate the impact of class distribution of training data on the classification performance. The six data sets, detailed in Appendix A, have been examined. The experiment is performed by several random samplings of each data set with a predefined imbalance level. It should be noted that the imbalance is measured by the variance in the class size of the training data. Since most data sets are not homogenous, to ensure that the performance loss is independent of the class difficulty, the average classifier performance of 20 random samplings for a desired level of class distribution imbalance is obtained. According to Figure (3.4), by increasing the imbalance of document distributions across the categories, the performance of both Rocchio and SVM classifiers, dramatically drops. The results of 20 Newsgroups and Reuters clearly illustrate the correlation between classifier performance and class distribution imbalance. These two data sets are more homogenous as compared to the other data sets. In a homogenous classification problem, the classification difficulty is uniformly distributed across all categories in the data set. In other words, in a homogenous classification problem all classes have similar complexities.

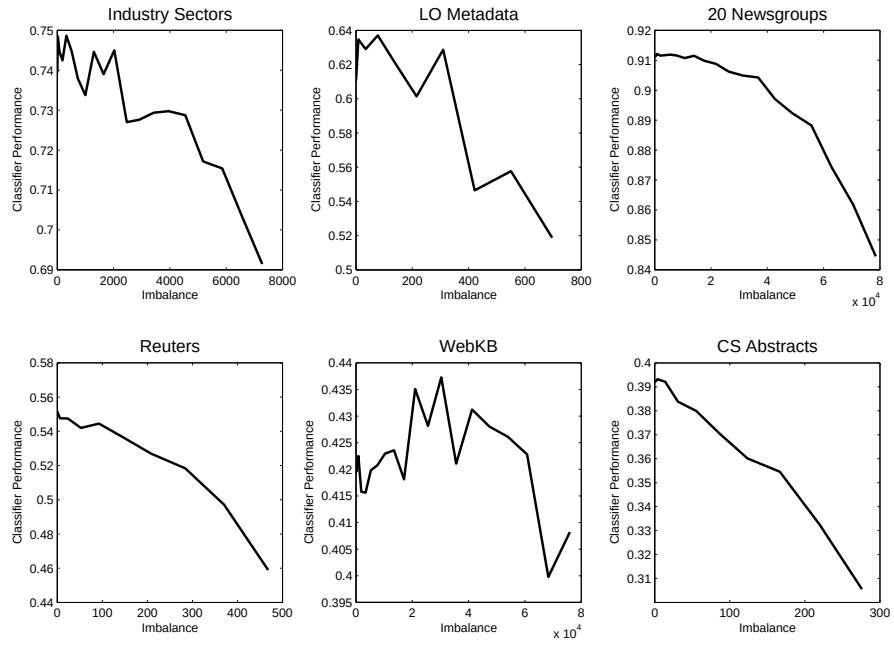
3.3.1 Data Sparsity

In Figure (3.4), classifier performance is considered as a function of class distribution imbalance. In some cases such as WebKB data with Rocchio classifier or Industry Sectors with SVM, this function is not monotonically decreasing. In order to explain these cases, we should note that, in fact, the most influential characteristic of training data, which can directly affect the performance of classifier, is data sparsity [6,76]. While the sparsity of classes are not balanced, class imbalance may re-balance the sparsity of classes and improve the performance. As a result, any class distribution imbalance affects classifier performance through increasing or decreasing the degree of sparsity in classes. The *local sparsity* of class k is estimated as follows:

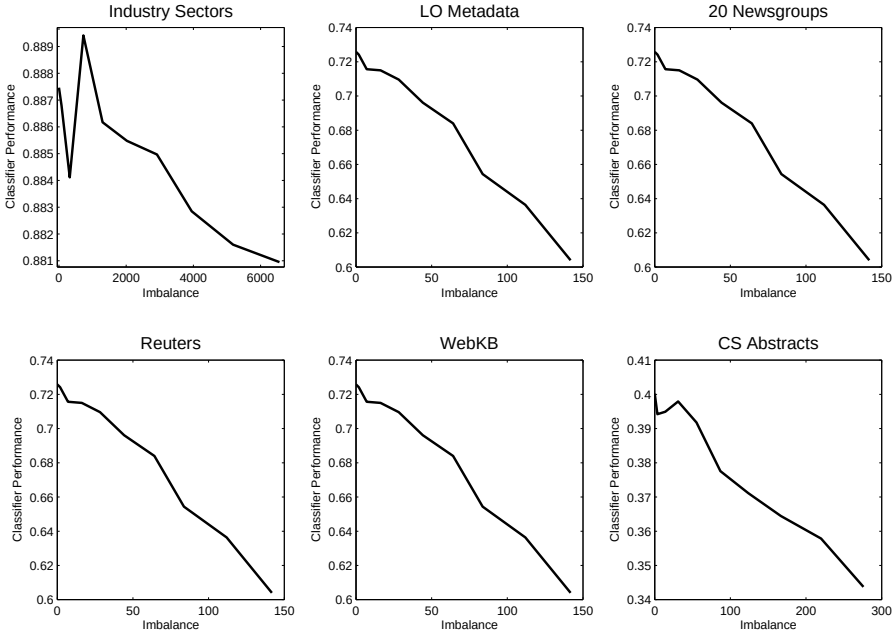
$$S(c_k) = 1 - \frac{\sum_{i=1}^{n_k} x_i}{m.n_k} \tag{3.3}$$

where x_i = the number of distinct words in the i^{th} document or the number of non-zero entries in the i^{th} row of the document-term matrix;

n_k = the number of samples in class k.



(a)



(b)

Figure 3.4: The impact of class distribution imbalance on the classifier performance: (a) Rocchio, and (b) SVM classifier.

In other words, because of high dimensional feature space in text classification problems, more than having a balanced training samples, we need to have all categories in the training data balancedly compact.

The measure of sparsity reflects the compactness of the training data space. The more compact data space, the more accurate model we can fit to predict unknown test data. As a result, the classifier performance is reduced by increasing (decreasing) the data sparsity (density which is equal to $1 - sparsity$). Let m be the length of the feature vector. In fact, the document data set, which is represented by a document-term matrix, is stored as a sparse matrix. Then, the size of the database can be measured by the sparsity index,

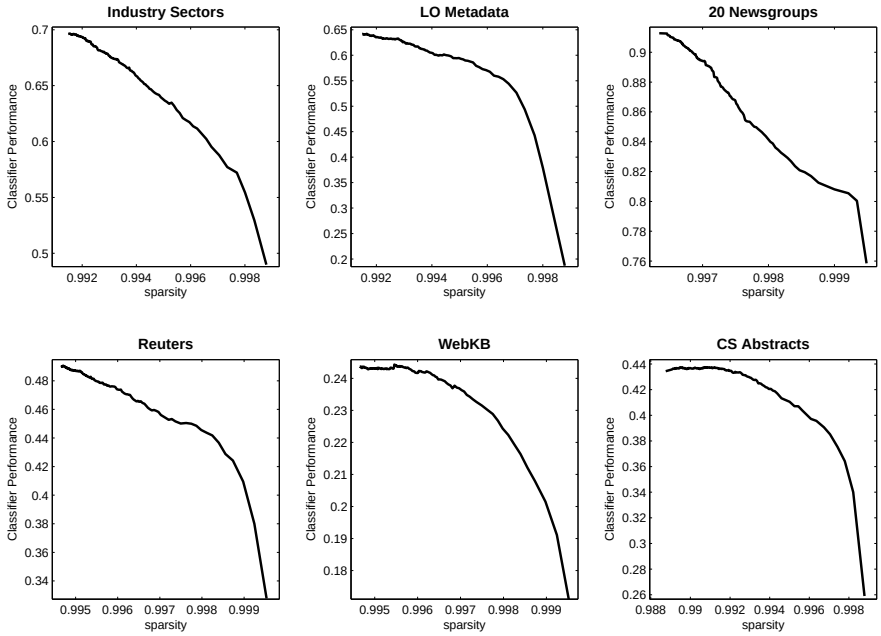
$$S = 1 - \frac{\sum_{i=1}^{n} x_i}{m.n}, \tag{3.4}$$

According to [6], by decreasing the sparsity index (3.4), the compactness of the learning model is improved, implying a more accurate prediction. We call this measure *global sparsity* because the categories are not involved. Figure (3.5) illustrates classifier performance as a function of global sparsity. The experiment is similar to that of Figure (3.4). In both classifiers and all data sets, classifier performance, which reflects the prediction accuracy of the classifier, is negatively correlated with global sparsity of training data.

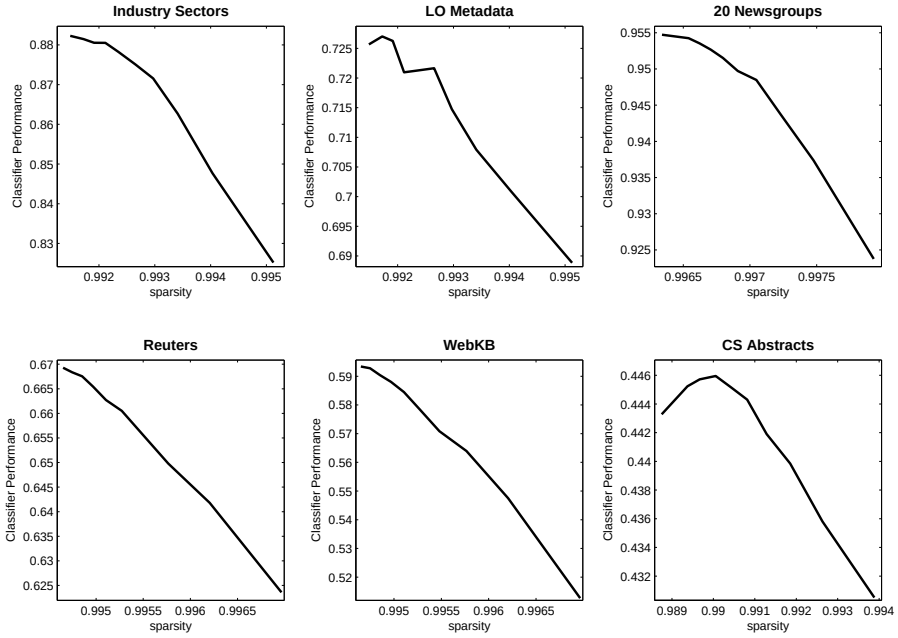
3.3.2 Feature Ranking for Imbalance Data

In Figure (3.4), the impact of unbalanced class distribution on the performance of classification was demonstrated. All experiments in this figure have been performed on full feature length and no feature ranking involved. We expect that feature ranking can intensify the impact of class imbalance. To illustrate the function of feature ranking, while dealing with unbalanced class distribution, an experiment, which is a simplified binary classification problem, is set up. Two data sets, including Industry Sectors and 20 Newsgroups, have been examined. From industry sectors, the classes "financial" and "healthcare" have been selected. Two classes of 20 Newsgroups are "sci.crypt" and "sci.med". From each class, 100 samples are randomly selected. Since we expect that feature ranking fails on smaller classes, in each experiment, F-measure of the smaller class ("financial" in Industry Sectors and "sci.crypt" in 20 Newsgroups data set) are estimated by using a Leave One Out Cross Validation (LOOCV) scheme. Two feature ranking measures, including IG and OR, have been employed using five ranking thresholds including $\{20, 40, 60, 80, 100\}$ percent of the total number of features (m).

According to Figure (3.6), by increasing the class imbalance, the classifier performance is decreased, which is exactly similar to the results in Figure (3.4). Figure (3.6) also shows



(a)



(b)

Figure 3.5: The correlation between global sparsity and the classifier performance: (a) Rocchio, and (b) SVM classifier.

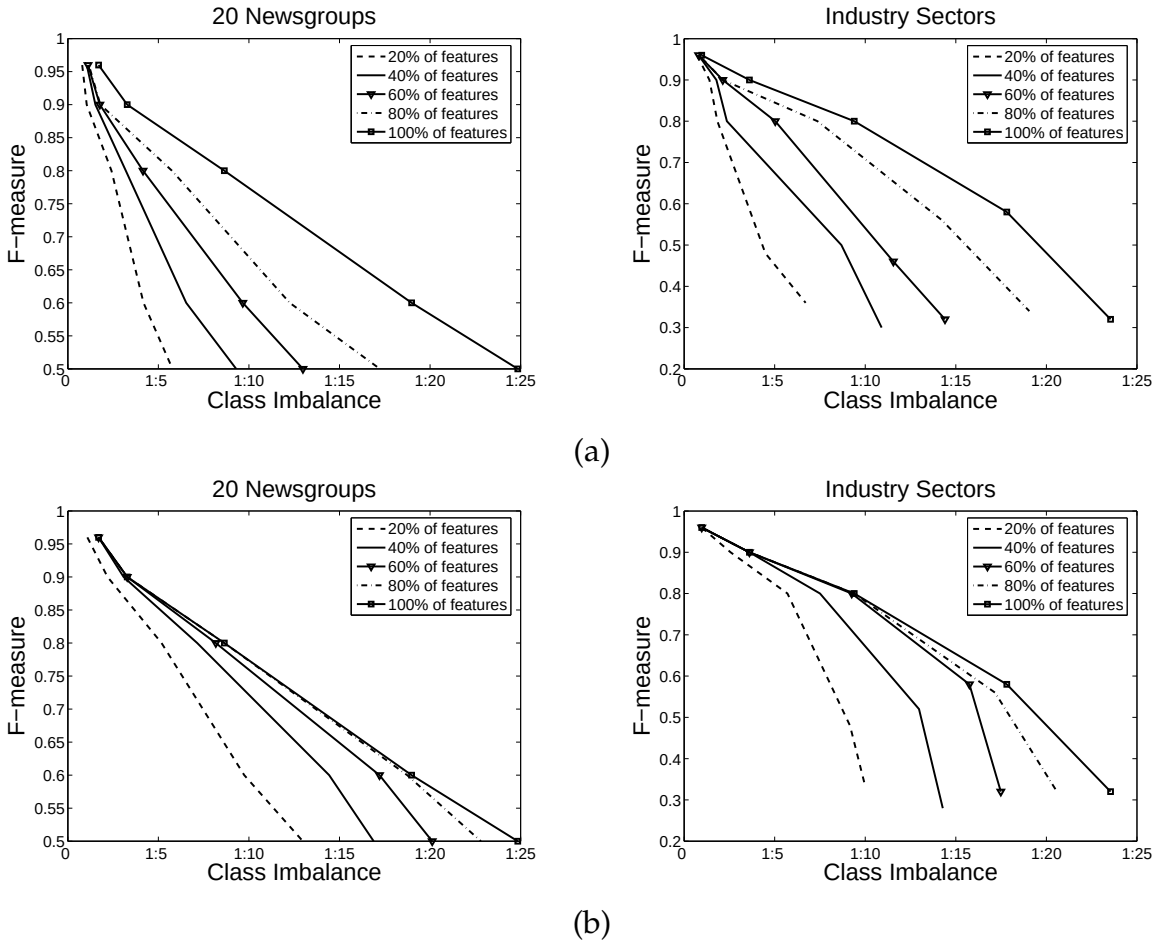


Figure 3.6: The impact of class distribution imbalance on the F-measure of the minority class with various feature length (feature ranking threshold) for 20 Newsgroups and Industry Sectors data sets using: (a) information gain, and (b) odds ratio ranking.

the impact of various ranking threshold on the classifier performance while dealing with imbalance data. In small feature vectors (aggressive feature selection), for instance 20% of full length of feature vector, the performance more sharply declines, while in longer feature vectors, the decline is smoother. The reason is that by removing more features of the training samples in small classes, they have less chance to have representative features contributed in building the classifier model. It should be noted that because of using binary-class problem, the class imbalance has been measured by notation ($n_s : n_l$), where n_s and n_l are the size of smaller and larger classes, respectively.

3.4 Conclusion

One well-known filter-based approach for excluding a large number of non-discriminant and irrelevant terms is feature ranking. Although feature ranking methods have a few disadvantages, such as ignoring the correlation between terms and the risk of term redundancy, they are highly scalable and less expensive as compared to other feature selection techniques. All these properties make feature ranking suitable for text classification problems. Similar to other filter methods, its design process is independent of the classifier model.

In this chapter, we explained that feature ranking methods have some serious drawbacks. They are problem-dependent such that their behavior varies from one problem to another. It has also been explained that the performance of a feature ranking measure varies based on the type of classifier. In Chapter 4, a framework is proposed to choose proper feature ranking measure with respect to the training data set and the choice of classifier. The proposed framework, which is called differential filter level performance, is practically considered a hybrid method, and requires to estimate the performance of a classifier.

Furthermore, we explained that feature ranking methods, despite their scalability and lower cost algorithms, suffers from lower performance compared to the search-based feature selection approaches such as wrappers. The low performance of feature ranking techniques arises from two issues:

1. Ignoring the dependency between terms and realizing a univariate selection while the nature of text classification problems is multivariate. The univariate function of feature ranking causes two serious problems: *(i)* ignoring correlation between features, and *(ii)* ignoring feature redundancies. According to the experiments in this

chapter, redundant terms not only have no discriminating benefits for the classifier, but also reduce the chance that other less informative but non-redundant terms can contribute to the classification process.

2. Failing in multi-class problems, especially with high class distribution imbalance. The majority of feature ranking methods fail when applied to multiple class problems with non-uniform class distributions (class skew). We showed in this chapter that any class distribution imbalance affects classifier performance through increasing or decreasing the degree of local (class) sparsities. Because of high dimensional feature space in text classification problems, more than having a balanced training samples, we need to have all classes in the training data balancedly compact. It was also explained that feature ranking can intensify the class imbalance effects on the classifier performance. By removing more features of the training samples in small classes, they have less chance to have representative features contributed in building the classifier model.

While using feature ranking methods to reduce the dimensionality, we should be aware of these two drawbacks, which arise from the function of feature ranking methods. For the first drawback, ignoring term dependency and correlation, we need to take extra effort to search for term dependencies including redundancies and correlations. Although they increase the cost of feature reduction using feature ranking, this extra cost is worth it since the search will be performed on a small number of features according to the ranking threshold, which is practically very small. For example, to search for only pair-wise term dependencies among q terms, the whole space, including $q(q-1)$ pairs, should be investigated.

The next step is to make a decision about the extracted dependencies. We can either merge the dependent terms or remove some of them and retain the rest. Heuristically, in the case of redundant terms, if asymmetric dependency between terms is given, it will be possible to consider removing the term with more dependency and retain the term with less dependency. This task needs to search for asymmetric term dependency. In this case, we cannot employ the mutual information measure to extract term dependencies.

The alternative approach is to focus on correlated terms regardless of the dependency direction. This approach is less expensive because we can employ mutual information measure, which requires $q(q-1)/2$ pair-wise (two variables) mutual information estimations. In this case, the conservative approach is to merge the terms with each other and form a new multiple-word term.

Empirical results on the impact of term dependency on feature ranking performance prompt us to work on extracting pair-wise term correlation using mutual information and merging correlated terms rather than removing the terms. We are also interested in extracting asymmetric dependencies, which can implement the redundancy reduction. This approach is detailed in Chapter 6.

Another direction in our research is to deal with imbalance data in feature level rather than data and algorithm level. Dealing with highly unbalanced data has been received lots of attention in machine learning community [87]. In this chapter, it has also been shown that the destructive effect of class imbalance on the classifier performance is worsened as we apply feature ranking-based feature selection. It is because the majority of feature ranking measures pay more attention to the features in large classes while those in small classes are usually ignored [23]. In other words, in an aggressive feature selection (very small feature vector), small classes are rarely represented by features in the feature vector. Although the data re-balance strategies such as minority class oversampling and majority class down-sampling [87] may increase the chance of small classes to have more representative features, we are interested in working on balancing techniques in feature level. In this approach, selected feature vector by feature ranking is compelled to cover all classes and contain a minimum number of relevant features from each class. In this case, we can employ a local feature selection (one classifier for each class). This technique is explained in Chapter 7.

Chapter 4

Differential Filter Level Performance

4.1 Introduction

In this chapter, a new method called differential filter level performance is introduced to identify the best feature ranking measure for text classification among a set of candidate measures. The proposed method requires two elements: first, a set of feature rankings, and second, their corresponding classifier performances. We introduce some heuristic criteria to compare two feature ranking measures. Next, a simplified, objective version of the criteria is proposed.

In the proposed approach, a classifier is used to estimate the trend of performance vs. the various levels of feature filtering. Such classifiers that are well-suited for this study are those that are sensitive to feature selection and the level of filtering. These classifiers are able to reflect the behavior of the feature rankings. For example, the SVM classifier is not a good choice due to its insensitivity to noise reduction. Therefore, a weaker classifier, sensitive to noise reduction and feature selection, in addition to being simple, fast and inexpensive, is more desirable. These characteristics point to the Rocchio classifier.

4.2 Classifier Performance as a Function of Number of Features

The objective in a text categorization system is the classification of documents into a fixed number of predefined categories. Let $\mathcal{D}$ be the set of documents and $\mathbf{C}$ be the finite set of categories. Let us suppose there is an ideal classifier $\mathbb{H}$, which is defined as $\mathbb{H} : \mathcal{D} \rightarrow \mathbf{C}$, where $\mathbb{H}$ is a many-to-one mapping of the documents to the class space. Practically, the

document space is randomly sampled such that a finite number of documents, which is called the set of training samples $\mathbb{D} \subset \mathcal{D}$, is employed to build the model. The resulting learning model $\mathbf{h}$ is, optimistically, a good approximation of $\mathbb{H}$ such that by applying the model to a set of unknown random samples or test data, which are independent of the training set, the test error $\epsilon_{\mathbf{h}}$ is minimized. To rephrase the text classification problem, it is an optimization of $\mathbf{h}$ to gain the highest generalization ability by minimizing the test error.

The classifier $\mathbf{h}$, in fact, is a family of learning solutions which can be defined based on the distribution of training data and the set of features. The text classification design process is to find an optimum $h \in \mathbf{h}$ minimizing the test error. In order to represent the classifier by the set of its features, the set of training data is assumed to be fixed during feature selection.

Let J be the performance of the classifier h. It can be expressed as a function of three independent variables

$$J(h) = f(\mathcal{G}, \mathcal{D}, H) \tag{4.1}$$

where $\mathcal{G}$ = the generalization ability or learn-ability of the classifier;
$\mathcal{D}$ = the data distribution;
H = the information provided by the selected features.

It should be mentioned that in (4.1) all three variables $\mathcal{G}$, $\mathcal{D}$, and H are assumed to be independent of each other.

During the feature selection, regardless of the approach, we always employ same classification algorithm in all steps. As a result, the learn-ability of a classifier has no change during feature selection. The performance function is also consistent with data distribution during feature selection, because in all iterations of feature selection, we use the same data. The total classifier performance is also unbiased to data distribution by applying different cross validation techniques (*i.g.* leave one out and K-fold cross validation).

As a result, in order to compare two feature ranking-based feature selection methods, while using exactly similar classifiers and data sets, the estimated performance is a function of information provided by features. It has been previously shown that there is a link between classifier Bayes error and mutual information [115, 21, 30]. In these works, the lower bound of Bayes error is estimated by maximizing the mutual information shared between features and classes. In this thesis, to simplify the case, we assume that the classifier performance is a function of information expressed by the features:

$$J(h) = f(H) \tag{4.2}$$

Theorem 1. Sigma-additive Information of Feature Space
In a feature ranking-based feature selection, the behavior of classifier performance $J(h)$ is a monotonic non-decreasing function of the number of features.

Proof. Let $\mathbf{T} = \{t_i | 1 \leq i \leq m\}$ be a finite set of terms, which is also known as vocabulary in text categorization. According to the vector space model [97], each term is considered as a feature, a degree of freedom, or a variable of the text categorization model. From [12], the information content of features can be estimated as follows:

$$H(t_1, t_2, ..., t_m) = H(t_1) + H(t_2) + ... + H(t_m) - MI(t_1, t_2, ..., t_m) \tag{4.3}$$

where $H(t_i) =$ the Shannon's entropy of feature t_i;
$MI(t_1, t_2, ..., t_m) =$ the summation of all mutual information values, addressing the correlation and information shared between features.

In feature ranking-based feature selection, the features are considered independent of each other [19]. In other words, in feature ranking, the features are assumed to be orthogonal, which means $MI(t_1, t_2, ..., t_m) = 0$. As a result, we have:

$$H(t_1, t_2, ..., t_m) = H(t_1) + H(t_2) + ... + H(t_m) \tag{4.4}$$

or we can rewrite (4.4) as follows

$$H(\bigcup_{j=1}^{m} t_j) = \sum_{j=1}^{m} H(t_j) \tag{4.5}$$

It means that the information provided by the features are a sigma-additive function. In other words, because $H(t_i) \geq 0$, $H(t_1, t_2, ..., t_m)$ is a monotonic non-decreasing function. As a result, from (4.2), the classifier performance is also a monotonic non-decreasing function of the number of features: $J(h(t_1, t_2, ..., t_m)) \equiv f(t_1, t_2, ..., t_m)$ or simply:
$J(t_1, t_2, ..., t_m) \equiv f(t_1, t_2, ..., t_m)$ where f is a monotonic non-decreasing function. □

The Theorem (1) can be explained by the following statement: the classifier performance monotonically increases as we increase the number of features. This is almost true if we assume that the feature vector is clean (no noise) and the features are independent of each other. This is an ideal case and a strong assumption made by feature ranking-based feature selection. In practice, we know that no ranking is perfect and always there are some noise in higher ranks, as well as feature correlation and redundancy.

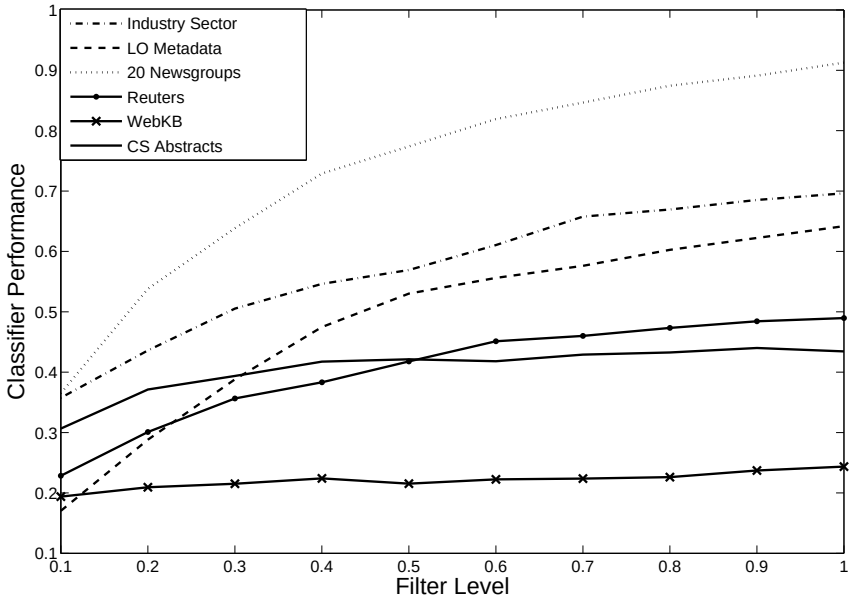
We can also empirically state the sigma-additivity of feature space (Theorem (1)) by an experiment (see Figure (4.1)). In this experiment, all features are randomly ranked. Starting with an empty selected feature set, each time we add 10% of highly ranked features to the selected pool of features, train the classifier, and estimate the classifier performance on test data using a five-fold cross validation scheme. The experiment is repeated 10 times on the six data sets studied in this thesis using Rocchio and SVM classifiers. In all cases, depicted in Figure (4.1), the classifier performance has a monotonic non-decreasing behavior, which confirms the Theorem (1).

4.3 Feature Ranking Sequence

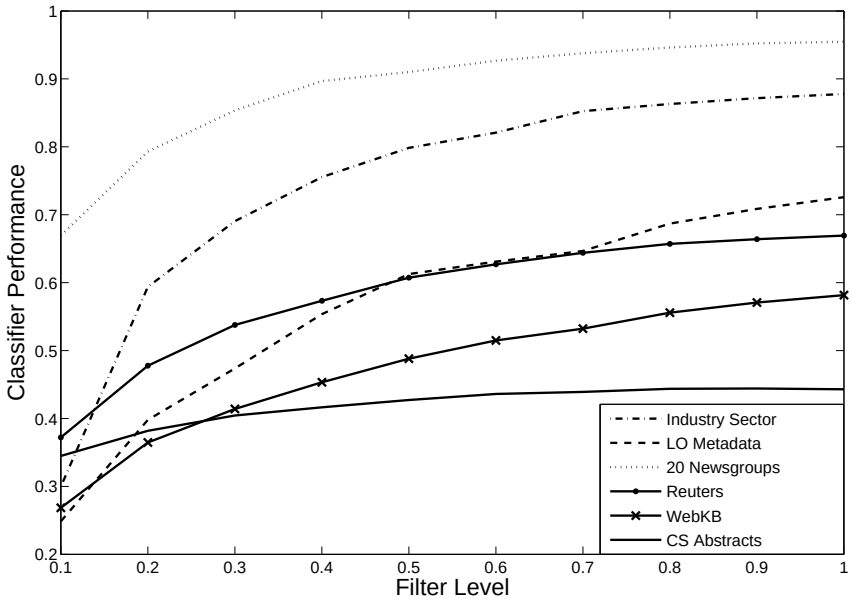
Definition 1. Feature Ranking Sequence
In a feature ranking method, all terms are sorted from high to low according to a ranking measure ϕ. The sort function is ζ such that $\zeta : \mathbf{T} \rightarrow \mathbf{V}$ where $\mathbf{V} = \{v_1, v_2, ..., v_m\}$ is the sorted set of the terms, such that $\phi(v_1) \geq \phi(v_2) \geq ... \geq \phi(v_m)$. The elements v_1 and v_m are called Head $H_s(\mathbf{V})$ and Tail $T_s(\mathbf{V})$ elements, respectively. $\mathbf{V}$ is a monotonically decreasing, finite sequence and called Feature Ranking Sequence, such that every subsequence $\mathbf{S}$ of the sequence $\mathbf{V}$ is still a sequence. $\mathbf{S} \sqsubseteq \mathbf{V}$ means $\mathbf{S}$ is a subsequence of $\mathbf{V}$.

In the feature ranking sequence, there is a restriction in order to define subsequences. Since we always want to select the terms with highest ranking measures, then the head of any subsequence of $\mathbf{V}$ has to be the same as the head of the original sequence $\mathbf{V}$. As a result, we always change the tail element of a feature ranking sequence to make subsequences. Due to this restriction, the *α-cut* of a sequence is defined.

Definition 2. α-cut Sequence
Let $0 < \alpha \leq 1$, the sequence $\mathbf{V}^\alpha$ is called the α-cut of the finite sequence $\mathbf{V}$ ($\mathbf{V}^\alpha \sqsubseteq \mathbf{V}$), such that $H_s(\mathbf{V}^\alpha) = H_s(\mathbf{V}) = v_1$ and $T_s(\mathbf{V}^\alpha) = v_{\lfloor \alpha . |\mathbf{V}| \rfloor}$ where $|\mathbf{V}|$ is the cardinal number of the sequence $\mathbf{V}$ and $\lfloor . \rfloor$ is the floor function (greatest integer less than or equal of the argument). The parameter α is called filter level or feature ranking threshold.



(a)



(b)

Figure 4.1: The classifier performance as a monotonic non-decreasing function of number of features: (a) Rocchio, and (b) SVM classifier.

For all $0 < \alpha_1, \alpha_2 \leq 1$ we have the following properties of α-cut operation on sequences:

$$\forall\, \alpha_1 > \alpha_2 : |\mathbf{V}^{\alpha_1}| \geq |\mathbf{V}^{\alpha_2}| \tag{4.6}$$

$$\forall\, \alpha_1 > \alpha_2 : |\mathbf{V}^{\alpha_1}| = |\mathbf{V}^{\alpha_1} \cup \mathbf{V}^{\alpha_2}| \tag{4.7}$$

$$for\ \ \alpha = 1 :\ \mathbf{V}^{\alpha} = \mathbf{V} \tag{4.8}$$

Definition 3. Power Sequence
Given a feature ranking sequence $\mathbf{V}$ *with size* m*, the power sequence* $\mathcal{P}(\mathbf{V})$ *is the set of all* α*-cut subsequences of* $\mathbf{V}$*, such that* $|\mathcal{P}(\mathbf{V})| = |\mathbf{V}| = m$*. For example, the power sequence of* $\mathbf{V} = (v_1, v_2, v_3)$ *is* $\mathcal{P}(\mathbf{V}) = \{(v_1), (v_1, v_2), (v_1, v_2, v_3)\}$.

According to Definition (3), feature selection based on feature ranking is a search among the power sequences of the feature space, while other feature selections such as wrappers are looking for a solution among the power set of the feature space. In other words, feature ranking is an $O(m)$ problem while others, *i.g.* wrappers, are in the order of $O(2^m)$.

Now let us build a feature ranking sequence by inverting the sorted set of feature scores. Let $\mathbf{\Phi}^- = \{\phi(v_m), \phi(v_{m-1}), ..., \phi(v_1)\}$ be the sorted set of feature scores such that $\phi(v_m) \leq \phi(v_{m-1}) \leq ... \leq \phi(v_1)$. The *backward feature ranking sequence* is defined as $\mathbf{\Lambda} = \{v_m, v_{m-1}, ..., v_1\}$. Since the backward feature ranking sequence $\mathbf{\Lambda}$ is a sequence like $\mathbf{V}$, it inherits all properties of the forward (regular) feature ranking sequence $\mathbf{V}$.

Definition 4. Inverse of Feature Ranking Sequence and Power Sequence
Given a feature ranking sequence $\mathbf{V}$*, its backward sequence* $\mathbf{\Lambda}$ *is also its inverse sequence, such that* $\mathbf{\Lambda} = \mathbf{V}^{-1}$*. We also define* $\mathcal{P}(\mathbf{\Lambda}) = \{(v_m), (v_m, v_{m-1}), ..., (v_m, v_{m-1}, ..., v_1)\}$ *as the inverse of power sequence* $\mathcal{P}(\mathbf{V})$.

4.4 Filter Level Performance

Definition 5. Filter Level Performance (FLP)
In a feature ranking sequence $\mathbf{V}$ *with corresponding vector of scoring measures* $\mathbf{\Phi}$ *and*

the set of corresponding classifiers $\mathbf{h}$*, Filter Level Performance (FLP) is defined as the set of classifier performance values of the power sequence* $\mathcal{P}(\mathbf{V})$*. Let us define the set of all corresponding classifiers* $\mathbf{h}$ *as follows,*

$$\mathbf{h} = h(\mathcal{P}(\mathbf{V})) = \{h(v_1), h(v_1, v_2), ..., h(v_1, v_2, ..., v_m)\} = \bigcup_{0<\alpha\leq 1} h(\mathbf{V}^{\alpha}) \quad (4.9)$$

Let $J(\mathbf{V}^{\alpha})$ *be the performance of the classifier* $h(\mathbf{V}^{\alpha})$*. We define* $\mathbf{J} = J(\mathcal{P}(\mathbf{V}))$*, which is the set of corresponding performance values, as FLP of the feature ranking sequence* $\mathbf{V}$*.*

According to Theorem (1), ideally, the FLP of any feature ranking sequence is a monotonic non-decreasing sequence. In other words, we have:

$$J(v_1) \leq J(v_1, v_2) \leq ... \leq J(v_1, v_2, ..., v_m) \quad (4.10)$$

or formally

$$\forall\ 0 \leq \alpha_1, \alpha_2 \leq 1,\ \ \alpha_1 \leq \alpha_2 :\ \ J(\mathbf{V}^{\alpha_1}) \leq J(\mathbf{V}^{\alpha_2}) \quad (4.11)$$

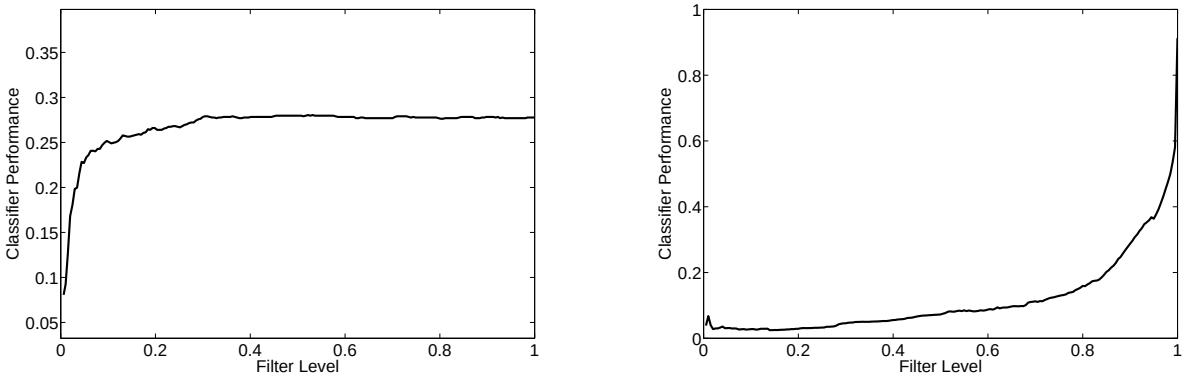
The implication of (4.10) is that if we repeatedly accumulate a new lower rank term (less ϕ) from the sequence to the set of terms, we expect to achieve a more accurate or at least similar classifier performance in every step. It means that in an effective feature ranking measure, classifiers become more accurate by increasing α.

Filter level performance can also be applied to the backward feature ranking sequence $\mathbf{\Lambda}$. Using sigma-additive property of feature space (Theorem (1)), regardless of ascending order of corresponding feature scores of $\mathbf{\Lambda}$, its FLP is also monotonic non-decreasing function of increasing number of features.

$$\forall\ 0 \leq \alpha_1, \alpha_2 \leq 1,\ \ \alpha_1 \leq \alpha_2 :\ \ J(\mathbf{\Lambda}^{\alpha_1}) \leq J(\mathbf{\Lambda}^{\alpha_2}) \quad (4.12)$$

The FLP characteristic is depicted by performance of the classifier as a function of the power sequence of $\mathbf{V}$ or simply as a function of α. The classifier performance is estimated by measures such as accuracy or F-measure. Figure (4.2) illustrates an example of a FLP characteristic. Each point of the FLP represents a classifier performance, or simply $J(\mathbf{V}^{\alpha})$.

Since the size of the power sequence used to calculate the FLP characteristic can be large, factor α is stepwise broken into $K \ll m$ steps called *resolution*. In other words, the $\mathcal{P}(\mathbf{V})$ is downsized to K sequences. For example, in most experiments in this Thesis, $K = 200$ such that every step of α is equal to $1/200$, and the corresponding increase in the number of features is approximately equal to $m/200$.



(a) (b)

Figure 4.2: Filter Level Performance (FLP) of feature ranking sequence: Classifier performance as a function of α or filter level: (a) forward, and (b) backward FLP.

Now, we are ready to characterize the feature ranking behavior using FLP model. Let us revisit the objectives of the feature ranking-based feature selection methods: *(i)* In a feature ranking, we are looking for a subsequence $\mathbf{V}^{\alpha} \subset \mathcal{P}(\mathbf{V})$ offering the best classifier performance $J(\mathbf{V}^{\alpha})$. In other words, we are looking for the best filter level α maximizing the $J(\mathbf{V}^{\alpha})$. *(ii)* Furthermore, we are also interested in smaller filter levels, which supports the smaller feature space and consequently more compact database.

The two objectives can be casted in a unified statement: a good feature ranking method is the one that compacts more informative features in a small feature ranking subsequence. It means that it can accumulate most information provided by features in very first features.

4.5 Comparing Feature Ranking Measures

Theorem 2. Differential Filter Level Performance (DFLP)
Let $\mathcal{P}(\mathbf{V}_A)$ and $\mathcal{P}(\mathbf{V}_B)$ be two power sequences of two feature ranking sequences $\mathbf{V}_A$ and $\mathbf{V}_B$ generated by two feature ranking measures Φ_A and Φ_B, respectively. Both sequences are generated from the same vocabulary $\mathbf{T}$. $\mathbf{V}_A$ is better ranking sequence than $\mathbf{V}_B$ if $\mathbf{V}_A$ does satisfy the following criterion:
$J(\mathcal{P}(\mathbf{V}_A)) - J(\mathcal{P}(\mathbf{\Lambda}_A)) > J(\mathcal{P}(\mathbf{V}_B)) - J(\mathcal{P}(\mathbf{\Lambda}_B))$.

Proof. Let us suppose

$$J(\mathcal{P}(\mathbf{V}_A)) > J(\mathcal{P}(\mathbf{V}_B)) \tag{4.13}$$

According to Definition (4), $\mathcal{P}(\mathbf{\Lambda}_A)$ and $\mathcal{P}(\mathbf{\Lambda}_B)$ are respectively the inverse of the given power sequences $\mathcal{P}(\mathbf{V}_A)$ and $\mathcal{P}(\mathbf{V}_B)$. Then we can write:

$$J(\mathcal{P}(\mathbf{\Lambda}_A)) < J(\mathcal{P}(\mathbf{\Lambda}_B)) \tag{4.14}$$

or

$$-J(\mathcal{P}(\mathbf{\Lambda}_A)) > -J(\mathcal{P}(\mathbf{\Lambda}_B)) \tag{4.15}$$

Inequalities 4.13 and 4.15 can be added together:

$$J(\mathcal{P}(\mathbf{V}_A)) - J(\mathcal{P}(\mathbf{\Lambda}_A)) > J(\mathcal{P}(\mathbf{V}_B)) - J(\mathcal{P}(\mathbf{\Lambda}_B)) \tag{4.16}$$

□

We call $J(\mathcal{P}(\mathbf{V}_A)) - J(\mathcal{P}(\mathbf{\Lambda}_A))$ in 4.16 *Differential Filter Level Performance (DFLP)*. The summation of this set can be used as a measure of merit to evaluate feature ranking measures. It is calculated as follows

$$\delta_{FLP} = \frac{1}{K}\sum_{i=1}^{K}(J(\mathbf{V}^{\lfloor\frac{m.i}{K}\rfloor}) - J(\mathbf{\Lambda}^{\lfloor\frac{m.i}{K}\rfloor})) \tag{4.17}$$

Figure (4.3) shows forward and backward characteristics for a ranking measure such as IG. Both are obtained under the same conditions, including the same classifier (Rocchio) and data set (WebKB). In order to select the best feature ranking measure, we are looking for a feature ranking measure that maximizes DFLP. It implies a backward-forward FLP with maximum positive and minimum negative areas in Figure (4.3). The positive area is associated with $forward > backward$ and the negative is associated with $forward < backward$.

Using DFLP, we are able to evaluate and compare a set of feature ranking measures. Let $\mathbf{V}_1$, $\mathbf{V}_2$, ..., and $\mathbf{V}_r$ be r feature ranking sequences generated by feature ranking measures ϕ_1, ϕ_2, ..., and ϕ_r, respectively. According to 4.16, the best feature ranking sequence among the sequences is the one that not only its power sequence maximizes the classifier performance J, but also the power sequence of its backward feature ranking sequence minimizes the classifier performance J. It means that focusing on the forward FLP is not enough to evaluate a feature ranking measure. In many cases, as we show in Section 4.6, forward FLP are flat for most feature ranking measures and we can differentiate

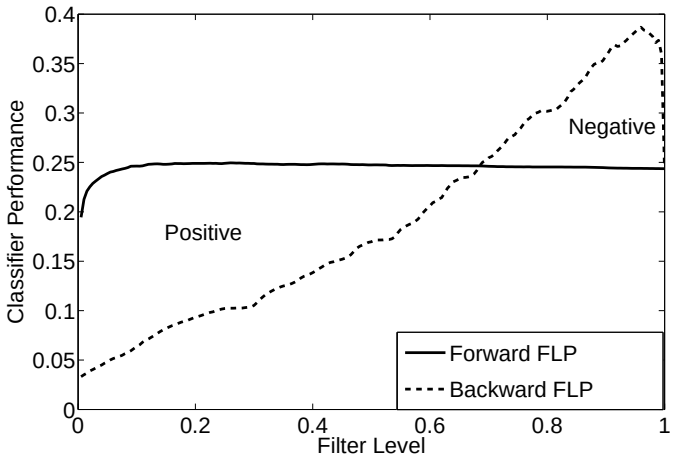


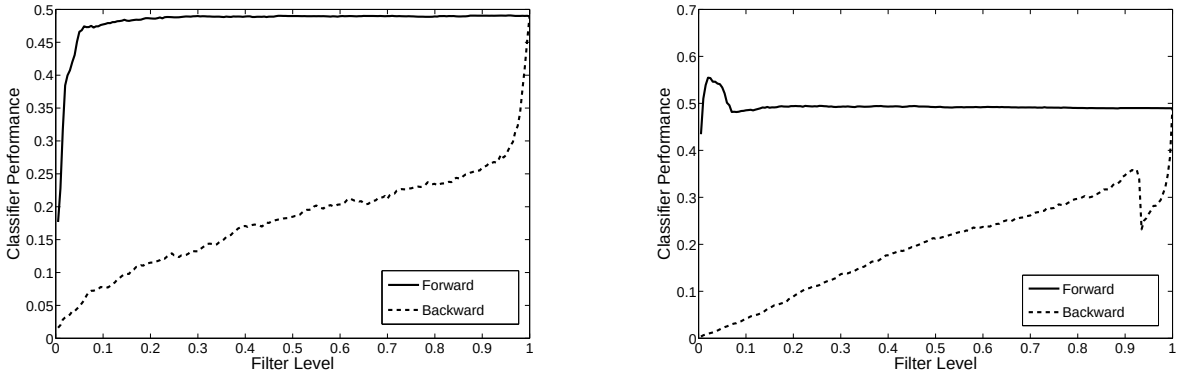
Figure 4.3: Forward and Backward FLPs of $Max(OR)$ ranking measure for the WebKB data set using Rocchio classifier. In ranking-based feature selection, the goal is to achieve larger positive and smaller negative areas.

them only by involving backward FLP. The feature ranking sequence and filter level performance provide a framework to compare the measures and identify the best feature ranking among a set of candidate measures.

In practice, the FLP characteristic is not necessarily monotonic. The reason can be explained as follows: In Theorem (1), features are assumed to be independent of each other, which is an essential assumption in feature ranking-based feature selection methods but nevertheless does not always work in reality. Especially in text classification, there is correlation and dependency between terms.

Figure (4.4) depicts two FLP examples, including the FLP of two feature ranking measures: Mean(χ^2) and CDF for Reuters data set. In Figure (4.3), any decline in forward or backward FLP implies that some informative and discriminant terms have been misplaced in the lower ranks, and some noise terms have been ranked as relevant terms. In Figure (4.4), CDF is not monotonic ($\delta_{FLP} = 0.28$), while Mean(χ^2) behaves monotonically and is more appropriate for feature ranking ($\delta_{FLP} = 0.34$).

It should be noted that the generalization capacity of text classifiers has significant influence on the rate of test error. For example, a SVM classifier can deal with noisy features with negligible loss of performance, while Rocchio and KNN are strongly sensitive to noise. With respect to the length of the feature ranking sequence, KNN suffers from a



(a) (b)

Figure 4.4: Two examples of forward and backward FLP characteristic for Reuters data set: (a) Mean(χ^2), and (b) CDF ranking.

high number of features while NBC performs better with a large number of terms [107]. Therefore, to consolidate the result of the study on feature ranking strategies, we should use those classifiers, which do not interfere with the performance of the feature ranking. For example, a SVM classifier can compensate for any misplacing of the terms in feature ranking and even deal properly with noise. Due to this fact, a classifier such as Rocchio, which is sensitive to feature selection, is used in most experiments.

4.6 Evaluation and Discussion

In this chapter, the method of differential and backward-forward filter level performance has been introduced to find the best feature ranking measure among a set of candidate ranking measures. To evaluate the proposed framework, nine feature ranking measures are applied to the six data sets, which are detailed in Appendix A. The set of feature ranking measures represented by RankSet includes: *F-measure*, *IG*, *NIG*, *Max(χ^2)*, *Mean(χ^2)*, *Max(OR)*, *CDF*, *IDF*, and *Random*. The set of document collections represented by DataSet includes: *Industry Sectors*, *LO Metadata*, *20 Newsgroups*, *Reuters*, *WebKB*, and *CS Abstracts*. To estimate the FLPs, a macro-averaged F-measure of the Rocchio classifier is obtained by applying the ranked features with variable resolutions. Every classification experiment is validated by the five-fold cross validation approach.

Figure (4.5) depicts the forward-backward FLPs for all *RankSet* $\times$ *DataSet* cases with resolution $K = 200$. As we are using the average of a five-fold cross validation, the

Table 4.1: The DFLP measures of DataSet × RankSet using Rocchio classifier.

	F-measure	IG	NIG	Max(X^2)	Mean(X^2)	Max(OR)	CDF	IDF	Random
Industry Sectors	0.32	**0.41**	0.40	0.34	0.28	0.25	0.26	0.15	-0.01
LO Metadata	0.51	0.49	0.51	0.51	0.49	0.40	**0.52**	0.42	0.00
20 Newsgroups	0.45	0.41	**0.80**	0.47	0.38	0.74	0.36	0.27	-0.00
Reuters	0.35	0.30	**0.36**	0.36	0.30	0.29	0.30	0.22	0.00
WebKB	0.06	0.07	**0.08**	0.06	0.05	0.06	0.06	0.03	-0.00
CS Abstracts	0.14	0.14	**0.23**	0.14	0.12	0.12	0.11	0.07	-0.01

results are reliable. Tables (4.1) and (4.2) represent the corresponding DFLP measures and ranking of feature ranking measures (meta-ranking) according to their DFLP values, respectively.

The following conclusions can be immediately drawn from the results:

1. The behavior of feature ranking measures depends on not only the classification algorithm [6,76], but also the data set characteristics. Figure (4.5) shows the extreme differences in behavior of the feature rankings while being applied to various data sets. In Table (4.2), IG scores far more varied results as compared to the other ranking methods.

2. The NIG ranking measure has the best performance in most cases. Surprisingly, IG and χ^2 offer poor results compared to NIG. Another newly proposed measure, CDF, is also beating other methods in the case of the LO Metadata collection.

3. IDF, which is widely used in information retrieval for term ranking, scores nearly the worst ranking and in fact is only better than the random ranking. IDF is also employed for removing high-DF terms, also known as stopwords. According to Table (4.2), the result of IDF-based filtering cannot be reliable, because it filters out not only noisy terms but also some relevant terms. Its sharp rising parts of backward FLP for Industry Sectors, 20 Newsgroups and CS abstracts, illustrate this point. IDF has also produced negative DFLP measures for Industry Sectors and WebKB, which means it is not able to filter out all noise terms such that there are some non-relevant terms in the higher ranks.

In similar experiments, we used a SVM text classifier instead of the Rocchio. Since the SVM classifier is more time-demanding, a lower resolution level ($K = 40$) was examined. Figure (4.6) illustrates the forward-backward FLPs for all *RankSet* × *DataSet* cases. A five-fold cross validation is also used to estimate the SVM classifier performance. In Table (4.3), the DFLP measures are depicted for the case of the SVM classifier, and Table (4.4)

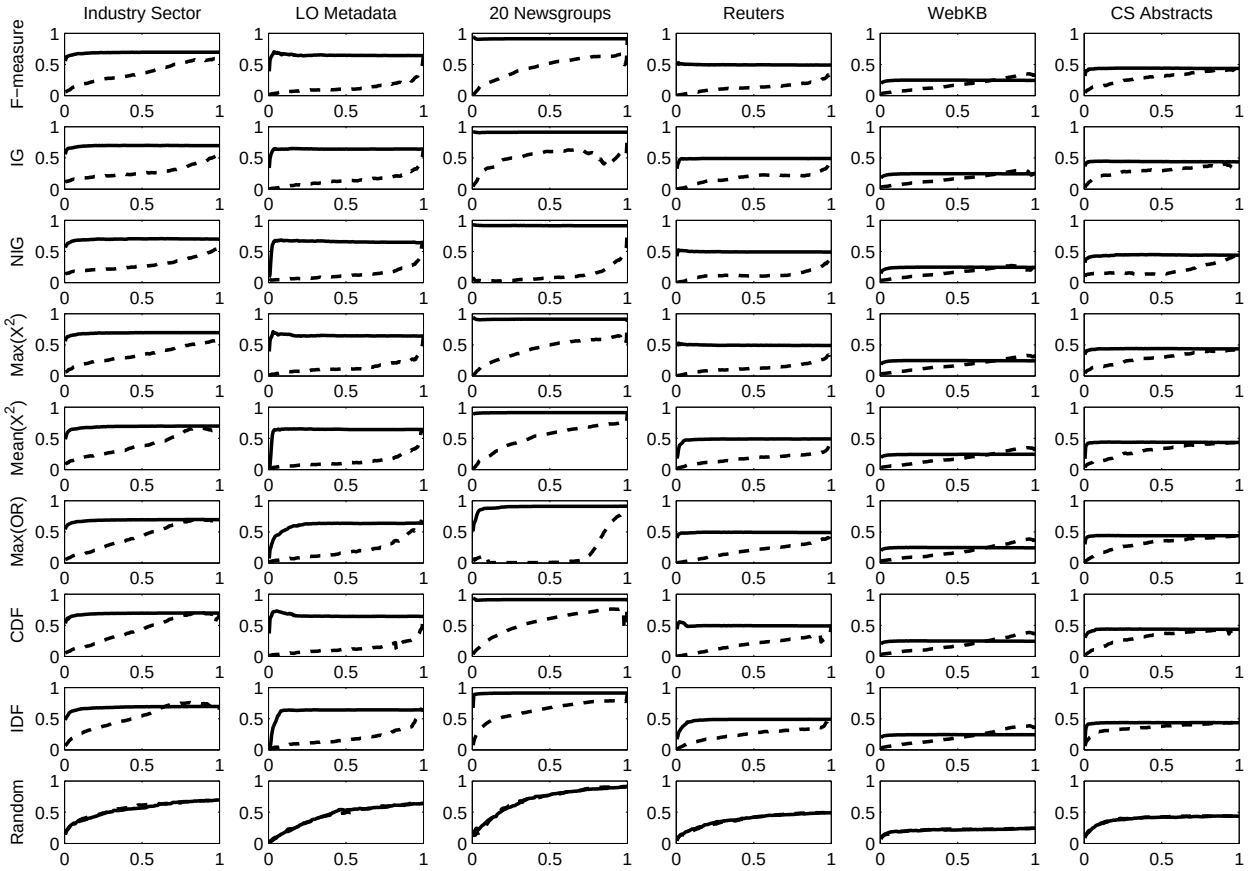


Figure 4.5: Forward and Backward FLPs of DataSet $\times$ RankSet using Rocchio classifier. Forward and backward FLPs are shown by solid lines and dashed lines, respectively.

Table 4.2: Meta-ranking results using Rocchio classifier: the data sets and their sorted feature ranking from best to worst.

	Meta-Ranking								
Data Set	1(*the best*)	2	3	4	5	6	7	8	9(*the worst*)
Industry Sectors	IG	NIG	Max(X^2)	F-measure	Mean(X^2)	CDF	Max(OR)	IDF	Random
LO Metadata	CDF	NIG	F-measure	Max(X^2)	Mean(X^2)	IG	IDF	Max(OR)	Random
20 Newsgroups	NIG	Max(OR)	Max(X^2)	F-measure	IG	Mean(X^2)	CDF	IDF	Random
Reuters	NIG	Max(X^2)	F-measure	Mean(X^2)	IG	CDF	Max(OR)	IDF	Random
WebKB	NIG	IG	Max(X^2)	F-measure	Max(OR)	CDF	Mean(X^2)	IDF	Random
CS Abstracts	NIG	Max(X^2)	IG	F-measure	Mean(X^2)	Max(OR)	CDF	IDF	Random

shows the meta-ranking results for the candidate ranking measures studied in this thesis. The results of the SVM classifier confirms our statement, which is that the feature ranking measure should be selected with respect to the data set characteristics and the choice of classifier. Based on DFLP measures, we are unable to recommend a unique ranking solution either for data sets or for classifiers. It means we must identify the best ranking solution based on the data set and the classification model to be employed. One approach, which has been introduced in this chapter, is the *meta-ranking* method.

Table 4.3: The DFLP measures of DataSet $\times$ RankSet using SVM classifier.

	F-measure	**IG**	**NIG**	**Max(X^2)**	**Mean(X^2)**	**Max(OR)**	**CDF**	**IDF**	**Random**
Industry Sectors	0.43	**0.46**	0.45	0.44	0.40	0.40	0.40	0.29	0.01
LO Metadata	0.46	0.46	0.44	0.46	0.46	0.37	**0.47**	0.41	0.00
20 Newsgroups	**0.35**	0.31	0.33	0.35	0.35	0.31	0.33	0.27	-0.01
Reuters	**0.41**	0.38	0.39	0.40	0.37	0.40	0.40	0.32	0.02
WebKB	0.35	0.35	0.35	0.36	0.33	**0.36**	0.36	0.31	-0.00
CS Abstracts	0.15	0.13	0.14	**0.16**	0.15	0.14	0.14	0.09	-0.00

To validate the results, and investigate the impact of the resolution, the experiment has been tried using different values for the resolution K. The DFLP measures, similar

Table 4.4: Meta-ranking results using SVM classifier: the data sets and their sorted feature ranking from best to worst.

	Meta-Ranking								
Data Set	1(*the best*)	2	3	4	5	6	7	8	9(*the worst*)
Industry Sectors	IG	NIG	Max(X^2)	F-measure	Max(OR)	CDF	Mean(X^2)	IDF	Random
LO Metadata	CDF	F-measure	Max(X^2)	Mean(X^2)	IG	NIG	IDF	Max(OR)	Random
20 Newsgroups	F-measure	Mean(X^2)	Max(X^2)	CDF	NIG	IG	Max(OR)	IDF	Random
Reuters	F-measure	Max(X^2)	CDF	Max(OR)	NIG	IG	Mean(X^2)	IDF	Random
WebKB	Max(OR)	CDF	Max(X^2)	IG	F-measure	NIG	Mean(X^2)	IDF	Random
CS Abstracts	Max(X^2)	F-measure	Mean(X^2)	Max(OR)	CDF	NIG	IG	IDF	Random

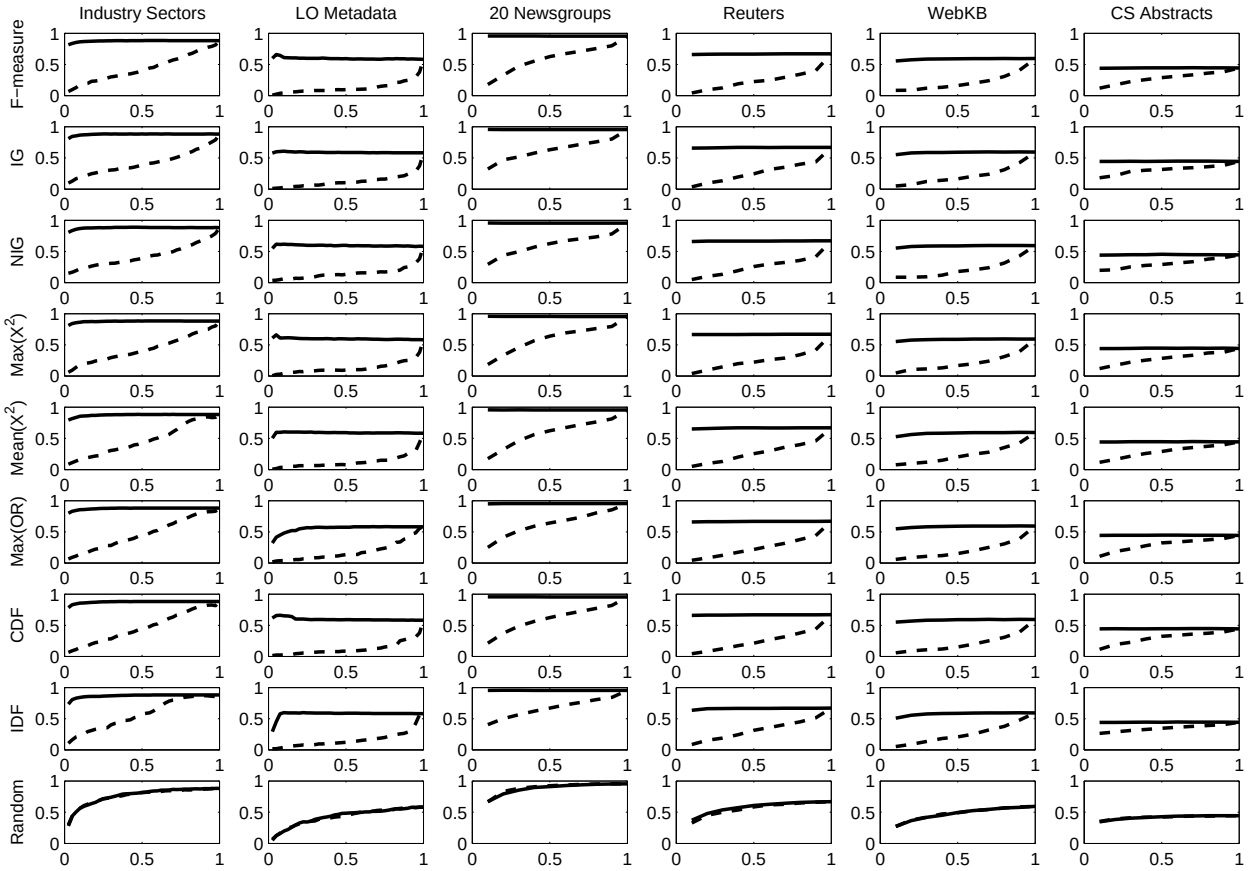


Figure 4.6: Forward and Backward FLPs of DataSet $\times$ RankSet using SVM classifier. Forward and backward FLPs are shown by solid lines and dashed lines, respectively.

to those in Figure (4.5), have been obtained for $K = \{5, 10, 20, 50, 100, 200\}$. In Table (4.5), each entry presents the average (and corresponding standard deviation) of DFLP measures for every couple of data set and ranking measures with 6 different resolutions. Since the standard deviation for almost all measures is quite low, we may conclude that the resolution has no strong effect on the FLP analysis of feature ranking measures. Table (4.6) represents the order of feature ranking measures with different resolution values. According to Table (4.6), DFLP enjoys stable behavior against the resolution. This means that for estimating the FLP characteristic of a feature ranking measure, or comparing two or more feature ranking measures, it is not necessary to perform experiments at higher resolutions. A less complex, and inexpensive algorithm may be employed to find the best feature ranking among a set of candidate measures.

The proposed method is mainly a meta-ranking algorithm (ranking the ranking measures), in which we are evaluating a set of candidate feature ranking measures. Although each feature ranking measure is a filter-based feature selection (with no classifier in the loop), the meta-ranking technique may be considered as a hybrid method, since we use the classifier in the loop to find the best ranking measure. Using the classifier in feature selection process can increase the cost of the algorithm, but we should note that the cost is almost negligible compared to that of the wrapper approach. In the DFLP method, the order of search or running the classifier is $O(K)$, while in the wrappers, the order is $O(2^m)$. We also showed that it is not necessary to employ a large resolution. According to the results in Table (4.6), even by the very small resolution K, the trend of forward and backward FLPs can be estimated.

The DFLP-based meta-ranking is recommended for the case of pattern classification with a high dimensional feature space such as text classification, gene recognition, and bio-informatics. In low dimensional cases, use of meta-ranking technique has no benefits compared to the wrappers, which offer good results with reasonable cost in the search for the best feature subset. In other words, when we are unable to employ the wrapper approaches, because of high dimensional feature space, we prefer to use DFLP-based meta-ranking to find the best feature ranking measure.

4.7 Conclusion

It has been previously demonstrated that there is no single feature ranking measure best-suited for all data sets and all classifiers [85]. In other words, the performance of a particular ranking measure depends on the data set characteristics, such as class distribution

Table 4.5: The average DFLP measures of DataSet × RankSet with six different values of resolution.

	F-measure	IG	NIG	Max(X^2)	Mean(X^2)	Max(OR)	CDF	IDF	Random
Industry Sectors	0.30±0.02	0.39±0.03	0.38±0.03	0.32±0.03	0.27±0.02	0.24±0.02	0.24±0.02	0.14±0.02	-0.01±0.00
LO Metadata	0.49±0.04	0.47±0.03	0.49±0.03	0.48±0.04	0.48±0.03	0.39±0.02	0.49±0.04	0.41±0.02	0.00±0.00
20 Newsgroups	0.41±0.04	0.37±0.05	0.76±0.05	0.43±0.04	0.36±0.04	0.72±0.03	0.33±0.04	0.25±0.03	-0.00±0.00
Reuters	0.33±0.03	0.28±0.02	0.34±0.03	0.34±0.03	0.29±0.02	0.27±0.02	0.28±0.03	0.21±0.02	0.00±0.00
WebKB	0.06±0.00	0.07±0.01	0.07±0.01	0.06±0.00	0.05±0.00	0.05±0.00	0.05±0.00	0.03±0.00	-0.00±0.00
CS Abstracts	0.13±0.01	0.13±0.01	0.22±0.01	0.13±0.01	0.11±0.01	0.11±0.01	0.10±0.01	0.06±0.01	-0.01±0.00

Table 4.6: Meta-ranking results: the achieved ranks of feature ranking methods for the six levels of resolution.

	F-measure	IG	NIG	Max(X^2)	Mean(X^2)	Max(OR)	CDF	IDF	Random
Industry Sectors	444444	111111	222222	333333	555555	666777	777666	888888	999999
LO Metadata	333333	666666	111122	554444	245555	888888	422211	777777	999999
20 Newsgroups	444444	655555	211111	333333	566666	122222	777777	888888	999999
Reuters	333333	555555	111111	222222	444444	667777	776666	888888	999999
WebKB	344444	222222	111111	433333	777777	555555	666666	888888	999999
CS Abstracts	434444	343333	111111	222222	555555	666666	777777	888888	999999

imbalance and sparsity. Its performance may also differ from one classifier to another. To deal with the nonrobust behavior of feature ranking measures, one strategy is to tailor the feature ranking to the particular data set and classifier.

In this Chapter, the problem of finding the best ranking measure among a set of candidate feature ranking measures was detailed and a framework to find the best solution based on meta-ranking. The proposed feature meta-ranking strategy used a newly introduced structure called feature ranking sequence with some specific properties. By extending the concept of the forward feature ranking sequence to backward sequences, Differential Filter Level Performance (DFLP), the difference between forward and backward filter responses, was defined.

According to the findings, while using the differential filter level performance measure, the best feature ranking among a set of candidate measures should have: *(i)* monotonically increasing classifier performances by increasing filter levels for both forward and backward sequences; *(ii)* sharp and smooth risings for forward and backward responses, respectively; and *(iii)* higher forward FLP response compared to the backward response in all levels of filtering. We learn from the numerical experiments that the proposed method offers promising and stable results with different resolution of filter levels. Compared to the recursive wrapper approaches, the proposed method is more scalable.

Chapter 5

Extracting Domain-Specific Stopwords

5.1 Introduction

In Chapter 4, a framework was proposed to find the best feature ranking method among a set of candidate methods for selecting relevant features. In this Chapter, FLP technique is employed in the reverse process, which is extraction of stopwords.

Stopwords or so-called common words, noise words or negative dictionary are considered as irrelevant, non-predictive, and non-discriminating words in text classification. They carry low information content and cause low retrieval rate and prediction results. In addition, stopwords make up a large portion of the textual data in text mining tasks, where dimensionality is a critical issue. Stopwords are identified by the following attributes:

- low discriminating values;
- negligible information content;
- high document frequency;
- correlation with most categories (in labeled corpus);
- correlation with most words in the vocabulary.

Typically, stopword removal, as well as stemming, is conducted prior to any textual data processing task. In most information retrieval and text categorization systems, all terms (words) are stemmed by a stemming algorithm such as Porter stemmer. Unlike stopword removal that reduces the vocabulary size by only a few hundred terms, stemming can cut the vocabulary size by as much as 40% [84, 36].

Stopwords are grouped into two categories: *general* and *domain-specific*. The former includes those standard stopwords, which are available in the public domain or non-standard stopwords which are generated inside information retrieval or text categorization systems [62, 103, 7, 83]. Domain-specific stopwords are recognized as a set of words which have no discriminant value within a specific domain or context. Domain-specific stopwords differ from one domain to another. For example, the term "learning", can be a stopword in the domain of "education", but a keyword in "computer science". The result of removing these terms, are similar to those of removing general stopwords that is an improved performance of the retrieval and categorization tasks. Not only can the performance be improved, but also the size of the database. Domain-specific stopwords have already been employed in areas such as physics, human resource management [14], bioinformatics, and gene ontology [101]. Unlike general stopword lists (stoplists) which usually have a fixed size, the size of domain-specific stoplists depends on the contextual and statistical characteristics of the corpus, including the sparsity (or density) of the corpus, the size of the vocabulary, and the number of sub-domains.

Both stopwords and stemming continue to be subjects of debate among researchers working on different aspects of natural language textual data. Their interests can be categorized into three groups.

- Natural Language Processing (NLP): For applications in this area, stopword removal and word stemming are sometimes questionable. They can reduce the performance of results. For example, each of the following words, "to be or not to be", is pivotal to the meaning. From this viewpoint, the effectiveness of stemming and stopword reduction, due to the linguistic correlation among words, is not guaranteed. For instance, by stemming or stopword reduction, the meaning of many singular and plural nouns, prepositions, verbs, and negations are lost. Therefore, NLP researchers conservatively conduct stopword removal [93, 22].

- Information Retrieval: In information retrieval applications, stopwords carry a very low retrieval value. They are removed not only from the database but also from the user's query [88]. However, even in information retrieval systems, stopword removal can reduce the performance. For example, abbreviations and symbols can be removed by stopword removal process, degrading information retrieval performance [106].

- Text Categorization: In text categorization applications, stopwords are treated as noise that increases the dimensionality without offering any significant prediction

capacity [100].

Stopword reduction is achieved by a standard stoplist, a high document frequency (High-DF) filtering, a feature ranking scheme, or a combination of all three methods [23]. Inspired by Zipf's law, where the number of documents in the data base is sufficiently high, the terms with high document frequencies are treated as stopwords. However, this rule fails in some cases, including the one where the documents are not uniformly distributed across the categories.

In this chapter, a newly developed method is described for generating non-standard and domain-specific stopwords. The drawback of previous methods, based on document frequency for stopword extraction, are outlined. Conventional feature ranking-based methods are evaluated by backward FLP. Using this evaluation, we illustrate that the behavior of feature ranking measures for scoring relevant terms differs from that for scoring the irrelevant terms and stopwords. According to the experimental results which are obtained by applying the proposed method to the six data sets, choosing the proper feature ranking measure for building stoplists depends on the data set characteristics, including the sparsity index.

5.2 Related Work

In [62, 103], outdated stoplists and web influences are mentioned as the two major motivations for the automatic extraction of the stoplists. There are also some other reasons to develop stopword extraction algorithms.

5.2.1 Outdated Stoplists

Standard stoplists become quickly outdated. For example, the first English stoplist was published in the 1970's [116]. Obviously, over time the usage of some popular words have changed, depending on social factors such as technological changes, cultural shifts, new media, and education levels. It is not surprising that revising, updating and optimizing current stoplists are crucial [103].

Although general stoplists, mostly for the English language, are available, the need for automatically constructing stoplists has not been obvious to researchers in text mining. The first initiative in stopword extraction has been accredited to Van Rijsbergen in 1979 [116]. His stoplist is one of the most used ones in NLP and information retrieval ap-

plications. Brown has developed the next most popular English stoplist adopted from [24] and called the Brown general stoplist.

In [62], information gain ranking has been applied for stopword extraction. The results have been compared with those of several document and term frequency measures. Additionally, the extracted stopwords are evaluated in an information retrieval query processing task by using TREC collections. As we show in this chapter, the information gain ranking is not comprehensive enough for stopword extraction. Choosing the proper ranking measure depends significantly on the data set characteristics.

To generate stoplists for web-specific documents, word entropy was employed in [104]. Since the method is unsupervised, the generated stoplist is evaluated by a web clustering scheme. In [103], the stoplist, generated by word entropy, is optimized via a k-means clustering and stochastic search algorithm. In [45], an association algorithm for producing stoplists, based on Receiver Operating Characteristics (ROC) analysis, has been suggested.

5.2.2 Web Impact

Influenced by new media such as the Web and new communication tools such as chat, email, and Short Message Service (SMS), some new words have become more common in daily English, for example, “email”, “contacts”, “URL”, and “link” [62].

5.2.3 Stoplist for Non-English Text Mining

Most of the research activities on NLP, information retrieval and text categorization have gone into the English language text. Recently, stopword lists have been published for other European languages. Table (5.1) lists the statistics of some stoplists [38, 37]. For more details on available stoplists one can refer to [38] and [37]. Regardless of European languages, there still exist some languages without standard stoplist.

In [98], a general stoplist, in addition to stemming, has been developed for French text. Also, stoplists for other languages such as Russian, Arabic, and Farsi (Persian) have been generated [81,7,112,83]. In most of the experiments, stopwords have been extracted according to their high frequent words, and evaluated by information retrieval systems. There are still some languages without standard stoplists.

Table 5.1: The number of stopwords in some European languages.

Catalan	124	Czech	136	Danish	99	Dutch	46-101	English	37-360	Finish	1134
French	124-155	German	127-231	Hungarian	33	Italian	132-279	Polish	108	Portuguese	145
Russian	256	Spanish	176-313	Swedish	114	Norwegian	117	Turkish	112		

5.2.4 Domain-Specific Stopwords

For the automatic extraction of domain-specific text mining and retrieval applications, we always need stoplists from the local vocabulary of a corpus. Domain-specific stopwords have been manually extracted in areas such as physics, human resource management [14], bioinformatics, and gene ontology [101]. Unlike general stoplists which have fixed size, the size of domain-specific stoplists depend on the contextual and statistical characteristics of the corpus such as sparsity (or density) of the corpus, the size of the vocabulary, and the number of sub-domains.

5.2.5 Formal Language Text Mining

Recently, text mining and statistical machine learning have been applied to formal language texts such as software source codes [69,35]. For instance, in software clustering [35], each function or procedure is represented by a bag of words, including reserved words, constants, variables, and function calls. The first two elements are the noise and stopwords to be removed and the last two elements, which carry more information, are features. The identification of stopwords for various formal languages requires an automatic extraction system.

5.2.6 Ontology Learning

Hub words have been introduced in [49]. They are related to many other words. Since one characteristic of stopwords is their correlation with other terms, hub words are viewed as a subset of domain-specific stopwords. Extracting hub words and building sub-domain vocabularies, also known as terminology, is a baseline for learning ontology.

5.3 Stopword Reduction Using Document Frequency

In text information retrieval and in text categorization systems, stopword reduction is usually performed by a stoplist. The first English standard stoplists were published in

the 1970*s*. Since then, many lists with few changes continue to be relied on. In addition to the use of a standard stoplist, most text categorization systems perform low and high document frequency filtering (Low-DF and High-DF). In this section, the benefits and drawbacks of classical DF-based filtering for stopword extraction and removal is discussed.

According to Zipf's law, in a corpus of natural language text, the usage frequency of any word is considered to be inversely proportional to its frequency rank. For example, the most frequently used words occur almost twice as often as the second most frequently word used, and the second one occurs twice as often as the third most frequent word, and so forth. In other words, in a vocabulary, only a few words are very frequent, whereas the majority of words occur only once. Both high frequent and low frequent groups carry some linguistic content, and typically facilitate the understanding of the meaning of the text. For text mining purposes both groups are removed, because they do not relevant enough to contribute to the learning process. Figure (5.1) illustrates this law for the six data sets used in this thesis. In each graph, the left tail indicates the High-DF terms and the right tail shows Low-DF terms.

Document Frequency (DF) is a measure that reflects the contribution of a term in a data set. It is assumed that all the terms in the vocabulary have the same importance. This assumption does not always work, because from a pragmatic point of view, the importance of the terms across the collection and its categories varies. The second assumption in estimating DF is that all the terms are uniformly distributed over the categories [61]. In other words, the DF is biased to uniformly distributed terms across the categories, which means DF can be potentially employed in stopword reduction. Since DF ignores the labels and class information of the documents, it is an unsupervised scoring measure that is widely used in text clustering. Each term is assigned a measure, representing the number of documents, containing the term. Other variants of DF are IDF, variance quality index [3], local DF [54], and CDF (see Chapter 2).

5.3.1 Removing Low Document Frequency Terms and Singletons

In the majority of text classification research, Low-DF terms are removed from the vocabulary. The threshold used for Low-DF filtering varies from one to more than ten, depending on the data sets [95, 22, 23]. Those Low-DF terms, which occur only once in the collection, are called *singletons*. The singletons are sometimes considered as stopwords [4]. Removing Low-DF terms dramatically reduces the vocabulary size, but it does

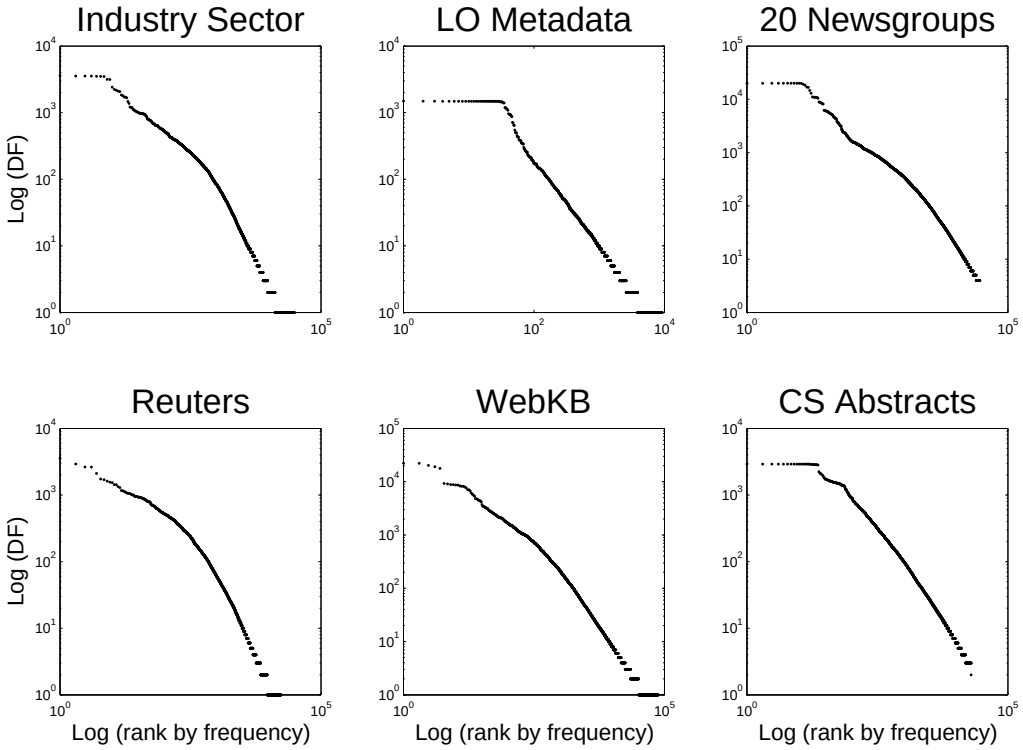


Figure 5.1: Zifp's law for different data sets.

not influence the sparsity index.

Low-DF terms include rare terms or phrases, spelling errors, and those have no significant contribution in the discriminating process of document category. Although these words from the information retrieval point of view have a critical role in indexing and retrieval, in the learning process, they have no information content and are treated as noise to be removed. Furthermore, in a family of feature ranking methods for feature selection, including χ^2 and odds ratio which behave unreliable and non-robust in the case of low frequent features, eliminating Low-DF terms is required.

However, removing Low-DF terms is risky. Sometimes Low-DF terms are useful in the classification process. In text classification, where difficult classes with less sparse vocabulary, or classes with very few samples, Low-DF terms and singletons can have a more predictive role. Another application of Low-DF terms is for the keyword extraction. Low-DF terms can be key indices for query processing to retrieve the appropriate documents. In this case, those Low-DF terms with a high TF which imply the popular $TF \times IDF$ weighting, are desired. It has also been demonstrated that by increasing the

threshold of Low-DF filtering (eliminating more Low-DF terms), precision drops and recall remains almost unchanged, or even, slightly improved [22]. Thus, by removing more Low-DF terms, a high recall is obtained at the cost of precision. The reason for this is that Low-DF terms are individual positive features or good keywords.

5.3.2 Removing High Document Frequency Terms

IDF is an information retrieval ranking measure and widely used in removing high frequent words, which are potentially considered as stopwords. For example, in [22] stopword removal is performed by removing all the terms with $DF \geq n/2$. This rule fails in the case of domain-specific stopword reduction and data sets with high class skew [22].

According to Zipf's law, unlike a Low-DF reduction, removing High-DF terms reduces the vocabulary size by only a small amount, and the database size by a significant amount. Although IDF is popular for stopword reduction, in Section 5.5, we show that it is one of the worst ranking measures. It can confuse some relevant words with stopwords so that, sometimes, the resulting stoplist leads to a misleading high classification accuracy.

5.4 Stopword Extraction Using Feature Ranking Measures

In previous stoplist generations [62, 103], the information retrieval systems have been used to evaluate the generated stoplist. In this thesis, we use a text classification performance measure to evaluate the extracted stoplists. In addition, this framework is employed to compare various extraction algorithms. The text classifier, employed for the stoplist evaluation, must be a weak classifier, sensitive to the noise and stopwords, and scalable and inexpensive as much as possible. The Rocchio classifier, which is used in this research, can meet these requirements [94]. For estimating the text classifier performance, the macro-averaged F-measure is employed [22, 23].

From the text categorization viewpoint, a set of terms $\mathbf{T_1}$ is more relevant than a set of $\mathbf{T_2}$, if it offers a better classification performance that addresses the prediction capacity. In the opposite direction, a set of stopwords $\mathbf{S_1}$ is better than that of $\mathbf{S_2}$, if $\mathbf{S_1}$ offers less prediction capacity or poorer classifier performance. To express these two statements into a unified expression, *the Area Under FLP (AUF)* is defined as

$$A_\tau = \sum_{0<\alpha\leq\tau} J(\mathbf{V}^\alpha), \tag{5.1}$$

and for backward FLP

$$A_\tau = \sum_{0<\alpha\leq\tau} J(\mathbf{\Lambda}^\alpha), \tag{5.2}$$

where $0 < \tau \leq 1$ = the maximum filter level or threshold;
$0 < \alpha \leq 1$ = the variable filter-level;
$\mathbf{\Lambda}$ = the backward feature ranking sequence;
J = estimated classifier performance.

As we discussed in Chapter 4, by FLP approach, the behavior of a ranking measure is analyzed and compared with that of other measures. The major disadvantage of this approach is that it requires a text classifier for estimating the FLP response. Although the Rocchio classifier, employed in this chapter, is simple and inexpensive, especially for large vocabularies, obtaining FLP characteristics is challenging.

One alternative approach is to approximately predict the classifier performance without performing any classification task [8]. Our idea is to analyze the data set characteristics and to estimate the performance trend. The data set *sparsity* is one of the appropriate data set characteristics, which, approximately, represents the classifier performance [6,23]. The sparsity index is calculated by various formulations. In [6], a global sparsity such as (3.4) is discussed.

Figure (5.2-a) depicts the correlation between the backward FLP and global sparsity for the data sets studied in this thesis. Each graph is the average of $9,000$ experiments, including nine ranking measures, 200 filter levels, and 5 different distributions of the training data (by 5-fold cross validation). In all the experiments, the Rocchio text classifier is used. This experiment is also performed with a different classifier to investigate the impact of the classifier model on the correlation between the classifier performance and global sparsity index. Figure (5.2-b) illustrates the correlation between the backward FLP and global sparsity for the data sets by the SVM. Each graph is the average of $1,800$ experiments, including 9 ranking methods, 40 filter levels, and 5 different distributions of the training data. To reduce the computation time, we use a smaller number of filter levels for the SVM-based FLP.

According to both sets of experiments and regardless of the type of classifier model, the FLP response and global sparsity index are strongly correlated, which has been informally addressed in [6]. As a result, for the FLP analysis, instead of obtaining the classifier performance, which can be expensive when sophisticated classifiers such as the SVM are used, we can estimate backward FLP by adopting the global sparsity of the document-term matrix, associated with the training data, when the same filter-level as FLP is applied

as follows:

$$J(\mathbf{\Lambda}^{\alpha}) = \frac{K_D}{[S(\mathbf{\Lambda}^{\alpha})]^{\gamma}}\ ,\ \ K_D > 0,\ \ 0 < \gamma \leq 1 \tag{5.3}$$

where $S(\mathbf{\Lambda}^{\alpha})$ = the sparsity of the filtered document-term matrix in which filter-level α is applied to the feature vector;

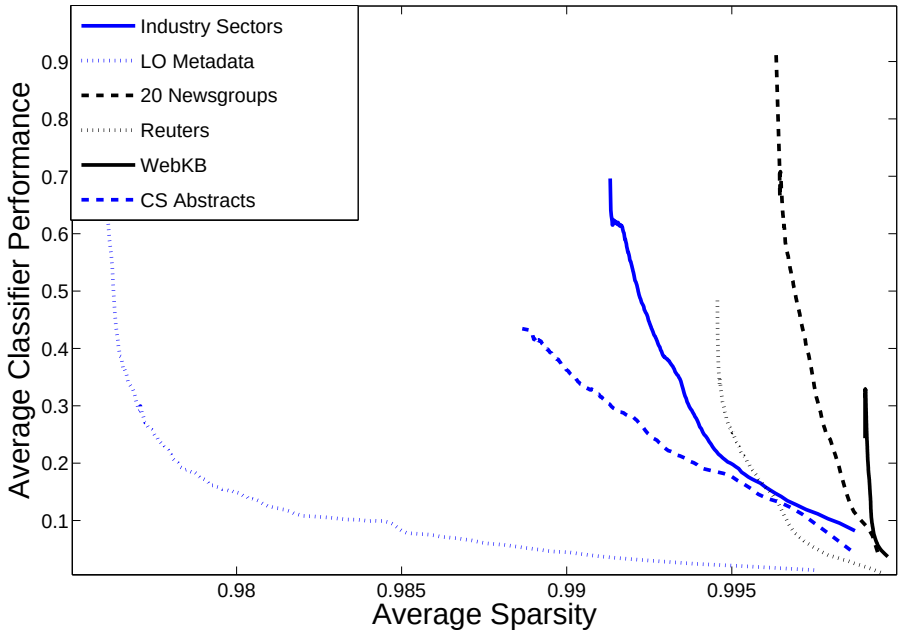
K_D and γ = constants depending on the type of classifier and the data set.

5.5 Experimental Results

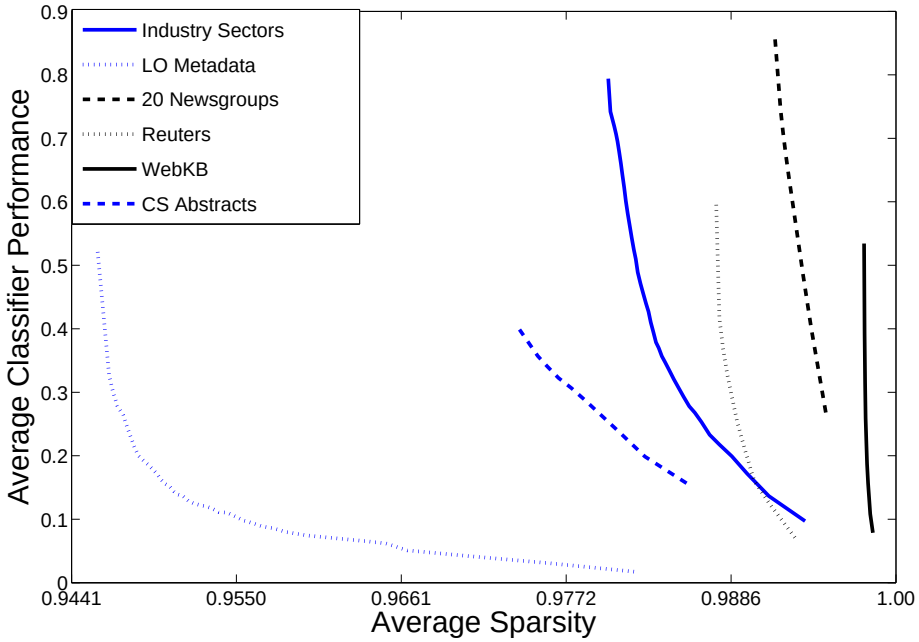
To evaluate the proposed framework, nine feature ranking measures are applied to the six data sets, which are detailed in the Appendix A. The set of feature ranking methods called RankSet, includes *F-measure*, *IG*, *NIG*, *Max(χ^2)*, *Mean(χ^2)*, *Max(OR)*, *CDF*, *IDF*, and *Random*. The set of document collections, represented by DataSet, includes *Industry Sectors*, *LO Metadata*, 20 *Newsgroups*, *Reuters*, *WebKB*, and *CS Abstracts*. To estimate the FLPs, the macro-average F-measure of the Rocchio classifier, a sorted list of features with different filter levels is obtained. All the classification experiments and sparsity estimations are validated by a five-fold cross validation technique.

Figures (5.3) and (5.4) illustrate forward and backward FLPs for all six data sets with $\tau = 0.1$ and 200 filter levels. With respect to the backward FLP responses (graphed in Figure (5.4)), it is evident that the performance of the IDF ranking measure is almost similar to that of Random ranking. In any circumstances, it cannot be the best ranking for stopword extraction. This experiment challenges the use of High-DF filtering for stopword reduction. In addition, the experiment indicates we cannot rely on IDF to filter out the stopwords, since, in most cases, its backward FLP is usually higher than that of others. It is implied that, by IDF ranking among terms with lower ranks, there are still some relevant and informative terms.

The second point is the inconsistency of the behavior of the term ranking measures in forward and backward filtering. In Figure (5.3), the forward FLPs are presented for all the ranking measures versus all the data sets. The results in this figure are not consistent with those in Figure (5.4), which exhibits the backward FLPs of the ranking measures. In other words, a good ranking is supposed to assign not only higher ranks to the relevant words but also lower ranks to the irrelevant stopwords. Table (5.2) summarizes the results in Figures (5.3) and (5.4), describing the behavior of each ranking measure by the AUF index. In Table (5.4), the rank of the ranking methods for each data set, with respect to their forward and backward AUF, is presented. According to Table (5.4), the best ranking



(a)



(b)

Figure 5.2: The correlation between global sparsity and classifier performance: (a) Rocchio, and (b) SVM classifier.

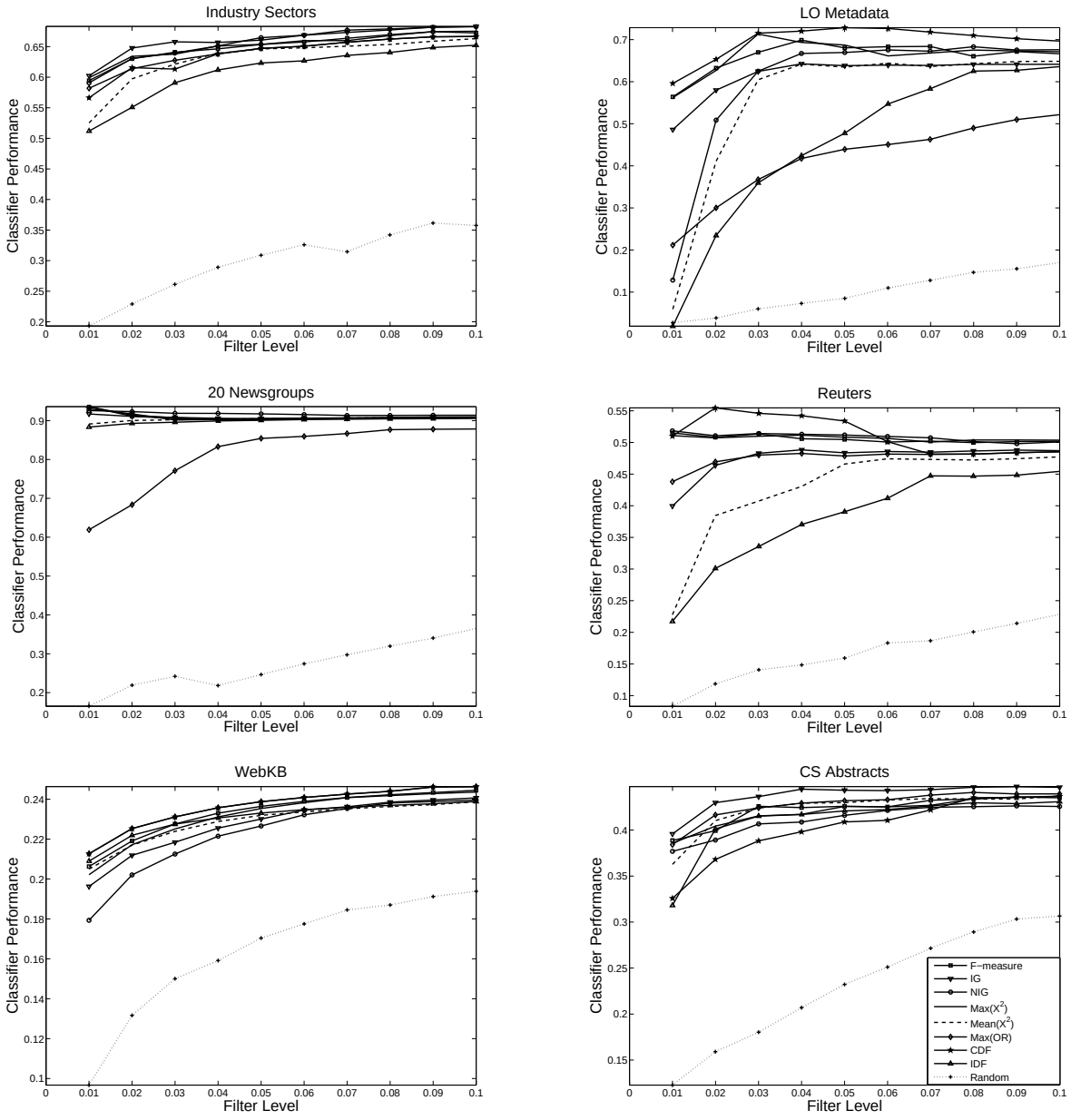


Figure 5.3: Forward filter-level performance of the ranking methods.

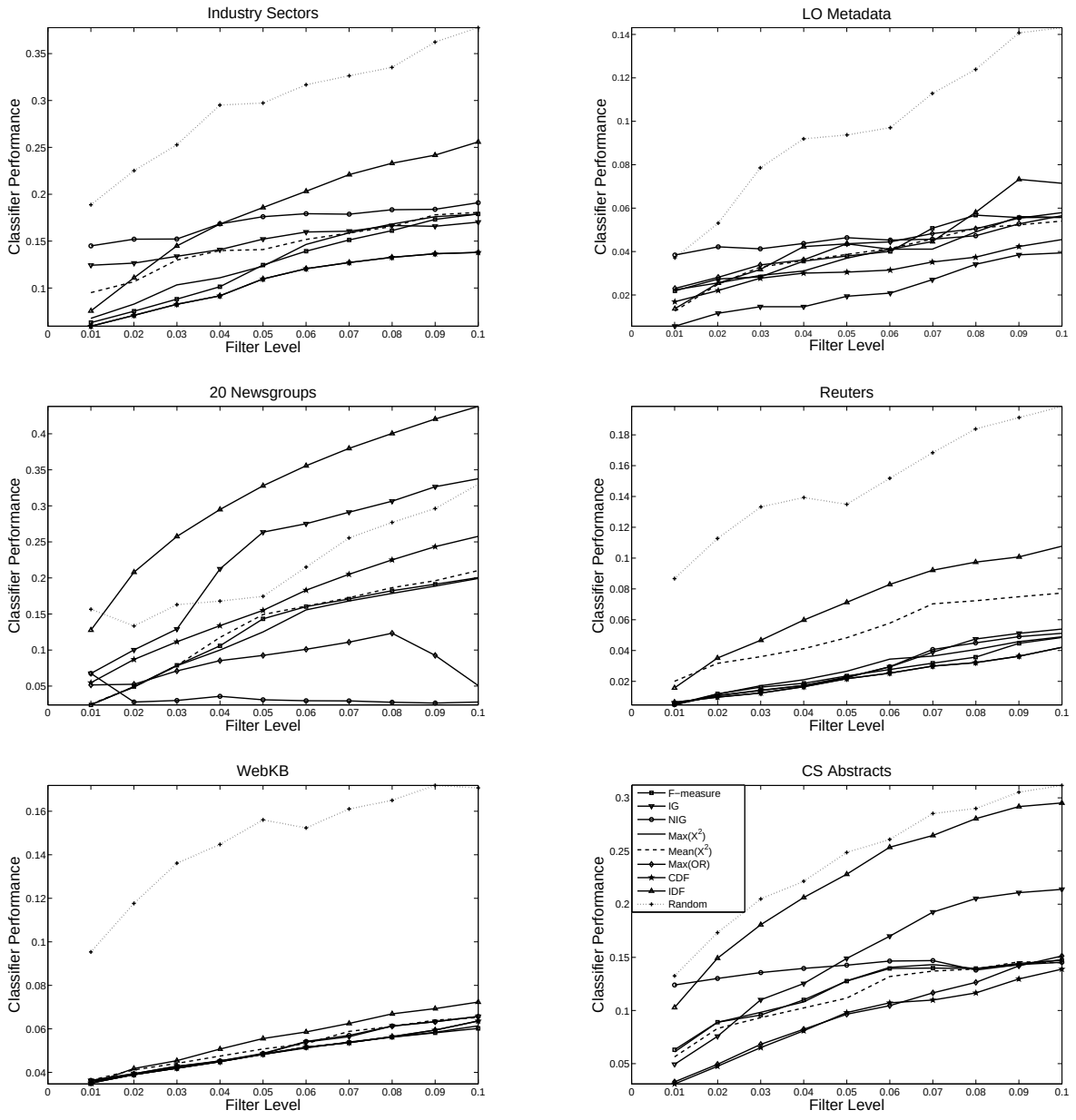


Figure 5.4: Backward filter-level performance of the ranking methods.

Table 5.2: The area under FLP of the feature ranking methods for different data sets using Rocchio classifier (forward,backward).

Data Set	IG	NIG	Max(χ^2)	Mean(χ^2)	Max(OR)	F-measure	CDF	IDF	Random
Industry Sectors	**6.61**,1.50	6.57,1.71	6.51,1.32	6.30,1.45	6.41,1.07	6.50,1.26	6.38,**1.07**	6.09,1.84	2.98,2.98
LO Metadata	6.17,**0.23**	5.98,0.46	6.63,0.39	5.57,0.39	4.17,0.42	6.61,0.41	**6.97**,0.32	4.53,0.45	0.99,0.97
20 Newsgroups	9.09,2.31	**9.17**,0.33	9.09,1.27	9.04,1.35	8.12,**0.83**	9.09,1.31	9.09,1.66	8.99,3.21	2.69,2.17
Reuters	4.75,0.29	5.08,0.29	5.07,0.29	4.29,0.53	4.76,0.23	5.05,2.26	**5.12**,**0.23**	3.82,0.71	1.66,1.50
WebKB	2.27,0.51	2.23,0.51	2.32,0.49	2.29,0.52	**2.36**,**0.49**	2.33,0.49	2.36,0.49	2.310.56	1.64,1.47
CS Abstracts	**4.38**,1.50	4.12,1.39	4.21,1.20	4.23,1.15	4.28,0.97	4.23,1.20	4.03,**0.92**	4.11,2.25	2.32,2.43

Table 5.3: The area under FLP of the feature ranking methods for different data sets using SVM classifier (forward,backward).

Data Set	IG	NIG	Max(χ^2)	Mean(χ^2)	Max(OR)	F-measure	CDF	IDF	Random
Industry Sectors	17.47,5.42	**17.49**,5.73	17.42,5.21	17.30,5.27	17.34,5.21	17.41,**5.19**	17.32,5.21	17.10,7.25	13.51,13.47
LO Metadata	11.85,1.17	11.99,1.74	12.09,1.40	11.83,1.14	10.59,1.48	12.08,1.34	**12.17**,**1.10**	11.37,1.53	6.62,6.38
20 Newsgroups	4.77,2.51	4.77,2.42	**4.78**,2.18	4.78,**2.14**	4.76,2.41	4.78,2.16	4.78,2.26	4.78,2.76	4.12,4.21
Reuters	3.31,0.71	3.32,0.73	**3.33**,0.70	3.31,0.74	3.32,0.64	3.32,0.67	3.32,**0.63**	3.29,0.99	2.57,2.40
WebKB	2.27,0.44	2.27,0.46	**2.28**,0.46	2.23,0.48	2.27,**0.42**	2.28,0.43	2.27,0.42	2.21,0.55	1.59,1.63
CS Abstracts	2.22,1.30	**2.23**,1.22	2.22,1.05	2.22,**1.04**	2.22,1.12	2.21,1.06	2.22,1.13	2.21,1.54	1.97,2.00

measures for feature selection (according to their Forward FLPs) are *IG*, *CDF*, *NIG*, *CDF*, *Max(OR)*, and *IG*, for the six data sets, respectively. On the contrary, the best ranking measures to extract stopwords are not exactly similar to those for term selection. The best ranking, in relation to the data sets are *Max(OR)*, *IG*, *NIG*, *Max(OR)*, *F-measure*, and *CDF*, which is different from the previous meta-ranking except for 20 *Newsgroups* data set. This data set is the easiest (with the minimum class skew), and most homogenous (similar difficulty for all classes) data set.

Also, we examined the previous setup with the SVM classifier. Table (5.3) depicts the AUF indices for the forward and backward FLPs for the SVM classifier, and Table (5.5) shows the rank of the ranking methods for each data set, with respect to their forward and backward AUF. According to the results, even by employing a powerful SVM classifier to estimate backward and forward FLPs, there is still an inconsistency between selecting relevant and rejecting irrelevant terms. This finding addresses the pitfall of using feature ranking methods to remove noise and stopwords.

To compare the proposed approach, which is based on an estimated backward filter-level performance by using the global sparsity measure, for other automatically building stoplists, the following experiment is set up. First, the stoplist, extracted by backward FLP characteristic, is considered as the baseline and optimum list. It should be noted that for all methods, 10% of the most irrelevant terms are selected as stopwords. The stopword list by using the IDF ranking, a classical approach, is also derived. Two well-known feature

Table 5.4: The data sets and the rank of feature ranking measures (meta-ranking) using Rocchio classifier (forward, backward).

Data Set	IG	NIG	Max(χ^2)	Mean(χ^2)	Max(OR)	F-measure	CDF	IDF	Random
Industry Sectors	1,6	2,7	3,4	7,5	5,1	4,3	6,2	8,8	9,9
LO Metadata	4,1	5,8	2,3	6,4	8,6	3,5	1,2	7,7	9,9
20 Newsgroups	5,8	1,1	4,3	6,5	8,2	3,4	2,6	7,9	9,7
Reuters	6,6	2,4	3,5	7,7	5,1	4,3	1,2	8,8	9,9
WebKB	7,5	8,6	4,2	6,7	1,3	3,1	2,4	5,8	9,9
CS Abstracts	1,7	6,6	5,5	3,3	2,2	4,4	8,1	7,8	9,9

Table 5.5: The data sets and the rank of feature ranking measures (meta-ranking) using SVM classifier (forward, backward).

Data Set	IG	NIG	Max(χ^2)	Mean(χ^2)	Max(OR)	F-measure	CDF	IDF	Random
Industry Sectors	2,6	1,7	3,2	7,5	5,3	4,1	6,4	8,8	9,9
LO Metadata	5,3	4,8	2,5	6,2	8,6	3,4	1,1	7,7	9,9
20 Newsgroups	7,7	6,6	1,3	3,1	8,5	2,2	4,4	5,8	9,9
Reuters	6,5	2,6	1,4	7,7	5,2	3,3	4,1	8,8	9,9
WebKB	4,4	3,6	2,5	7,7	5,1	1,3	6,2	8,8	9,9
CS Abstracts	3,7	1,6	6,2	4,1	2,4	7,3	5,5	8,8	9,9

ranking measures, *IG* and *Max(χ^2)* [62] are also considered in the experiment. The best feature ranking measures, which are obtained by forward FLP, are also examined. Finally, the stoplist is extracted by the estimated backward FLP by using global sparsity. All the lists are compared with the baseline stoplist by using the F-measure. According to Table (5.6), which illustrates the results of the comparison, the sparsity-based estimation of the FLP provides the most stopwords that are similar to the baseline stoplist. The IDF offers poor results compared to the other methods. Adopting the best feature selection method by the forward FLP characteristic, and IG perform better than IDF, but they cannot outperform the sparsity-based estimated backward FLP results. Table (5.7) presents a list of the first 50 domain-specific stopwords, which are extracted by the proposed approach by using sparsity measure.

Table 5.6: F-measure of extracted stoplists compared with the baseline stoplist.

Data Set	Global Sparsity	Best Forward FLP	IDF	IG	Max(χ^2)
Industry Sectors	0.9170	0.9170	0.5034	1.0000	0.7823
LO Metadata	1.0000	1.0000	0.5416	0.6891	0.3750
20 Newsgroups	1.0000	1.0000	0.7529	0.9831	0.6092
Reuters	0.6494	0.6494	0.6381	0.9736	0.6494
WebKB	1.0000	0.7724	0.5241	0.9983	0.8589
CS Abstracts	1.0000	0.8751	0.6749	0.8751	0.5458

Table 5.7: Extracted stoplists (first 45 stopwords) using Backward FLP method.

Industry Sectors	LO Metadata	20 Newsgroups	Reuters	WebKB	CS Abstracts
deem	archer	cantaloup	ryan	mead	bibtex
inaccuraci	clickabl	cmu	pesticid	tei	citat
older	colombia	cs	helm	inet	context
av	gord	date	huski	puma	copyright
everydai	manor	id	vista	interplai	document
foster	motor	messag	rehear	sarita	download
purpos	toi	newsgroup	atla	unawar	entri
vari	2y2	path	wallcov	horizon	rate
oppos	mississauga	srv	hyo	judi	relat
creation	murphi	subject	medco	ajit	researchindex
carl	0a9	biscuit	tele	michal	previou
proud	0m5	stylu	lube	cuni	varieti
attribut	albertan	bobcat	proprietari	athena	advanc
stress	alex	ocsmd	absb	rowspan	intend
break	biotech	preffer	initit	marginwidth	typic
10k	bridg	sunil	spun	bud	exist
relianc	burn	abba	weis	pierr	examin
geograph	candid	overtur	caremark	wharton	determin
overse	cariboo	gtri	dyneer	toledo	directli
est	chase	belvilad	dyr	surgic	lot
trail	chevi	mattel	craa	usl	grant
jpg	conflict	fervour	55p	22th	main
unlaw	cyberu	hasta	microfilm	ariane5rep	advantag
teamwork	dole	implor	hazard	asplos7	separ
smaller	downtown	nirvana	produkt	cmg	difficult
unmatch	eat	svh	fslic	durand	propos
satisfact	eleph	knoweth	goodyear	gradi	produc
divers	elmer	adsorb	glendal	grievanc	attempt
undertak	evid	cfr	anacomp	ipps96	difficulti
highwai	fail	gleam	datagraphix	irrit	describ
typograph	gene	namepl	gd	lca	modifi
australian	healthier	addon	reshap	ogden	progress
peter	huronia	kfc	mercer	overdu	origin
freedom	immun	lenienc	goldsmith	untyp	basi
costli	kimberlei	haywir	nynex	win3	chapter
bachelor	kitchen	crise	reincorpor	wendi	achiev
confid	landmark	marina	agrico	dani	paper
russel	manuscript	pdy	tandi	psych	brown
15th	mcmaster	ramakrishnan	avial	abet	section
challeng	muir	saigon	paradis	bhatti	introduc
suffici	ordin	mightili	carson	cs380d	pre
joe	parl	skimpi	leaseback	helm	chu
outgo	parliamentari	license	trelleborg	legion	reli
tenth	pei	cslab	berisford	swept	facilit
manageri	poison	dex	deerfield	tertiari	multipl
increasingli	publiqu	hembruch	simmon	twain	juli
reveal	restor	jhj	forb	reuter	singl
strengthen	rope	mhembruc	macandrew	bogu	lab
dictionari	scene	orchestra	ot	bristol	global
contributor	cex	wega	brooksvill	cs384m	appear

5.6 Conclusion

Standard stoplists, which are used in information retrieval and text categorization, are outdated. Automatically building stoplists are also required in applications such as domain-specific text mining, ontology generation, non-English text processing, and formal language textual data mining. Conventional methods for stopword extraction are based on removing the terms with low and high document frequencies. In this chapter, the risks of the document frequency approach are discussed. For supervised stopword extraction, which uses labeled training data, feature ranking measures such as IG and χ^2 are employed. According to the results in this chapter, if a given feature ranking can perform well for selecting good features, the selection of poor features (stopwords) in the opposite direction, are not guaranteed. The reason is that term rankings behave differently, whereas ranking relevant terms from scoring irrelevant words. This fact is studied by introducing a new evaluation model, called the area under backward filter-level performance.

Using the notion of backward FLP and meta-ranking framework, we can identify the best term ranking measure, which minimizes the prediction capacity of selected terms as candidate stopwords, among a set of candidate ranking methods. The novel optimum solution can extract the most irrelevant stopwords so-called the baseline stoplist. The major disadvantage of this approach is that it employs a classifier to obtain filter-level performance. One alternative approach is to use training data characteristics to estimate the classifier performance. In this chapter, we use the global sparsity index, after term selection, to predict the trend of text classifier performance. According to the experimental results, obtained by applying the proposed approach to the six benchmark textual data sets, sparsity index offers a good estimation of classifier performance. The result of sparsity based estimation is almost better than other feature ranking measures, and entirely outperforms traditional inverse document frequency.

Chapter 6

Learning Term Dependency

6.1 Introduction

During recent years, many researchers in the text classification field have constantly repeated that feature selection is unnecessary for a SVM classifier, which is directly adopted from findings of Joachims [43]. According to [43], even the features ranked the lowest still contain considerable information for SVM and removing those features tends to hurt the performance of the classifier. These findings seem to be against the need for feature selection in text classification problems. However, this is true when using feature ranking methods, evaluating features individually, and ignoring term redundancy. For example, [58] demonstrated that even SVM classifier performs better when redundancy is reduced by a recursive wrapper approach. It also showed that a weaker classifier with feature selection can beat a SVM classifier without feature selection.

In Chapter 3, it was discussed that one of the major disadvantages of feature ranking-based feature selection methods is that they ignore the dependencies among the features [125]. Dependency can be considered between two or more features. It has also been explained that the impact of term redundancy is more destructive when employing a small number of features [127] (aggressive feature selection). Furthermore, there are some reports showing that redundancy reduction can improve the performance of feature selection algorithms [95, 125].

The problem of redundancy reduction is to find an efficient redundancy extraction algorithm in terms of low computational complexities. The major difficulty in redundancy extraction is calculating the correlation between features. This calculation can be expensive. Few simplified term redundancy reductions such as the $\mu-$occurrence mea-

sure have been reported [107]. They propose special cases such as binary class problems or assessing only pair-wise term redundancy without considering the class labels, which can increase the complexity of the problem.

6.2 Mutual Information

Mutual information has been employed as an indicator of relevance and dependency between two or more variables. Because of potential capability of mutual information to measure these dependencies, it has been widely adopted by researchers for feature selection [2, 113, 117, 52, 80, 53, 114, 1]. The problem with mutual information is its difficult calculation. Estimating mutual information can be complicated when dealing with continuous variables [2,52,53]. One well-known approach is dividing the continuous feature space into some discrete partitions and estimating the entropy and mutual information between them using the definition of discrete entropy and mutual information. Since in this thesis, we use the binary term weighting to represent features in vector space model, we do not face this problem.

Most efforts on employing mutual information to extract redundancy and correlation between features have gone into implementing an objective function to maximize the information content in the search-based filter approach feature selection [1, 2, 52, 80]. Due to the complexity of mutual information estimation, especially in the case of continuous features and high dimensional feature space, employing mutual information can be expensive for text classification problems. The minimum cost of calculating the mutual information is at least quadratic for the case of estimating only pair-wise mutual information, while in the filter-based approach we need to calculate the information shared between a candidate feature with all selected features.

Mutual information is a measure of statistical information shared between two probability distributions. Based on the definition in [63], mutual information $I(x;y)$ is computed by the relative entropy of a joint probability distribution, such as $P(x,y)$ and the product of the marginal probability distributions $P(x)$ and $P(y)$

$$I(x;y) = D(P(x,y)||P(x)P(y)) = \sum_x \sum_y P(x,y) log \frac{P(x,y)}{P(x)P(y)} \tag{6.1}$$

which is called the Kullback-Leibler divergence. Mutual information, such as other information theoretic measures, widely used in language modeling, has been applied in text mining and information retrieval for applications such as word association [9] and feature selection [117].

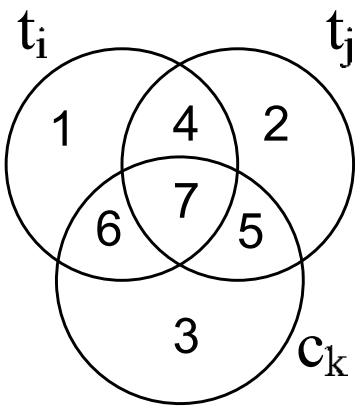


Figure 6.1: Entropies and mutual information between two terms and one class.

Figure (6.1) depicts the scenario. According to the figure, the area $\{4\} + \{7\}$ is associated with the mutual information between two terms regardless of the class information that they share. In most text classification applications, this area is referred as mutual information. However, this is a simplified case. A more complicated case is the mutual information between an individual term and previously selected features. One attempt to involve the class information in mutual information has been reported in [1], in which the main objective is to select a set of features that maximizes the area $\{6\}$.

Similar to most previous works [53, 2, 80], in this thesis mutual information is viewed as the entropy of co-occurrence of two terms. In other words, we are interested in the information shared between two terms. Eq. 6.1 can be rewritten as follows:

$$I(t_i; t_j) = P(t_i, t_j) log \frac{P(t_i, t_j)}{P(t_i)P(t_j)} \tag{6.2}$$

where $I(t_i; t_j) =$ the mutual information of the distribution of terms t_i and t_j.

In other words, $I(t_i; t_j)$ is the entropy of $P(t_i, t_j)$, which is the joint probability distribution of the terms t_i and t_j. If the two terms are completely correlated, then $I(t_i; t_j) = 1$, and $I(t_i; t_j) = 0$ if the two terms are completely uncorrelated.

6.3 Information Theoretic Inclusion Index

In feature selection, the dependency can be asymmetric (term redundancy) or symmetric (term correlation). Mutual information is a symmetric measure and unable to evaluate asymmetric relations. In order to define a measure of redundancy or asymmetric term dependency, we know that the information that is concordantly shared by two terms can

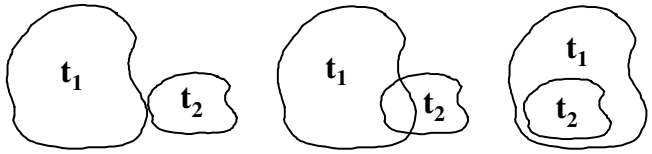


Figure 6.2: Inclusion relation between terms t_1 and t_2, (a) no inclusion relation between t_1 and t_2, (b) t_1 partially includes t_2, (c) t_1 includes t_2.

be estimated by mutual information or $I(t_i; t_j)$. A good asymmetric dependency measure may represent the contribution of each partner in the mutual information.

Let $\theta(t_i; t_j)$ ($\theta(t_j; t_i)$) be the dependency of t_j to t_i (t_i to t_j). One approach is to employ an inclusion index to evaluate the degree of dependency between two terms [67,65]. Inclusion index $\theta(t_i, t_j)$ [97], measuring how much t_i includes t_j, is calculated by:

$$\theta(t_i, t_j) = \frac{||t_i \cap t_j||}{||t_j||} = \frac{n(t_i, t_j)}{n(t_j)}, \ \ \theta(t_i, t_j) \neq \theta(t_j, t_i) \tag{6.3}$$

where $||.|| =$ the cardinal number of the set;
$n(t_i) =$ the number of documents in the training data containing t_i;
$n(t_i, t_j) =$ the number of documents in the training data containing t_i and t_j.

Inclusion index can be also estimated by conditional probability of t_i given t_j as follows:

$$\theta(t_i, t_j) = P(t_i|t_j) = \frac{P(t_i, t_j)}{P(t_j)} \tag{6.4}$$

where $P(t_i, t_j) =$ the joint probability of t_i and t_j.

If t_i completely covers t_j, then $\theta(t_i, t_j) = 1$, which is a full inclusive case. On the contrary, $\theta(t_i, t_j) = 0$ means that there is no overlap between the two terms. There is also partial inclusion when $0 < \theta(t_i, t_j) < 1$. t_j is called more inclusive than t_i if $\theta(t_i, t_j) < \theta(t_j, t_i)$ (see Figure (6.2)).

The inclusion index can be also expressed by information theoretic terms. The amount of information provided by a conditional probability distribution such as $P(t_i|t_j)$ is measured by conditional entropy $H(t_i|t_j)$,

$$H(t_i|t_j) = H(t_i) - I(t_i; t_j) \tag{6.5}$$

The upper bound of $H(t_i|t_j)$ is $H(t_i)$, when $I(t_i; t_j) = 0$. It means there is no overlap between the two terms and they are independently distributed. The lower bound of $H(t_i|t_j)$

is zero when $I(t_i;t_j) = H(t_i)$, addressing a full inclusive case ($t_i \subset t_j$). In order to map the conditional entropy $H(t_i|t_j)$ to unit distance $[0,1]$, it is normalized by $H(t_i)$:

$$H_n(t_i|t_j) = \frac{H(t_i|t_j)}{H(t_i)} = 1 - \frac{I(t_i;t_j)}{H(t_i)} \tag{6.6}$$

Since $H_n(t_i|t_j)$ decreases as we increase the overlap between two terms, we can estimate the *Information theoretic inclusion index* by using the normalized conditional entropy as follows:

$$\theta(t_j,t_i) = 1 - H_n(t_i|t_j) = \frac{I(t_i;t_j)}{H(t_i)} \tag{6.7}$$

and

$$\theta(t_i,t_j) = \frac{I(t_i;t_j)}{H(t_j)} \tag{6.8}$$

Using Figure (6.1), we also have:

$$\theta(t_i,t_j) = \frac{\{4\}+\{7\}}{\{2\}+\{4\}+\{5\}+\{7\}} \tag{6.9}$$

$$\theta(t_j,t_i) = \frac{\{4\}+\{7\}}{\{1\}+\{4\}+\{6\}+\{7\}} \tag{6.10}$$

It is interesting that the information theoretic definition of the inclusion index is exactly similar to the coefficients of constraint [10] and coefficients of uncertainty [86]. Inclusion index can be also linked to the Normalized Mutual Information (NMI) proposed in [109]:

$$NMI(t_i;t_j) = \frac{I(t_i;t_j)}{\sqrt{H(t_i).H(t_j)}} = \sqrt{\theta(t_i,t_j).\theta(t_j,t_i)} \tag{6.11}$$

where $NMI =$ the normalized mutual information;
$\sqrt{\theta(t_i,t_j).\theta(t_j,t_i)} =$ the geometric mean of inclusion indices.

It is worthwhile to mention that mutual information has no upper bound. By normalizing $I(t_i;t_j)$, the maximum value of the mutual information is one.

Using the asymmetric dependency $\theta(t_i;t_j)$, we can evaluate the following statements:

- $\theta(t_i,t_j) > \theta(t_j,t_i)$: The term t_j is dependent to the term t_i and might be redundant.
- $\theta(t_j,t_i)$=$\theta(t_j,t_i) > 0$: The term t_i and t_j are correlated.
- $\theta(t_j,t_i)$=$\theta(t_j,t_i) = 0$: The term t_i and t_j are independent.

From (6.7) and (6.8), we need to calculate several information theoretic terms including: $H(t_i)$, $H(t_j)$, and $I(t_i;t_j)$. In text classification, we usually approximate the probabilities by frequency measures. Table (6.1) depicts some of the most common frequency terms that we use to estimate the probabilities. These measures are easily obtained from the document-term matrix of the training data set. Using Table (6.1), the information

Table 6.1: Estimating the probabilities with document frequency measures.

summarized notation	frequency	probability	comment
a	$n(t_i)$	$n.P(t_i)$	# documents containing t_i
b	$n(t_j)$	$n.P(t_j)$	# documents containing t_j
c	$n(c_k)$	$n.P(c_k)$	# documents in c_k
d	$n(t_i,t_j)$	$n.P(t_i,t_j)$	# documents containing t_i and t_j
e	$n(t_j,c_k)$	$n.P(t_j,c_k)$	# documents in c_k containing t_j
f	$n(t_i,c_k)$	$n.P(t_i,c_k)$	# documents in c_k containing t_i
g	$n(t_i,t_j,c_k)$	$n.P(t_i,t_j,c_k)$	# documents in c_k containing t_i and t_j
n	n	-	# documents in the training data set

theoretic terms are estimated as follows [63]:

$$H(t_i) = P(t_i)\log\frac{1}{P(t_i)} = -\frac{a}{n}\log\frac{a}{n} \tag{6.12}$$

$$H(t_j) = P(t_j)\log\frac{1}{P(t_j)} = -\frac{b}{n}\log\frac{b}{n} \tag{6.13}$$

and

$$I(t_i;t_j) = P(t_i,t_j)\log\frac{P(t_i,t_j)}{P(t_i)P(t_j)} = \frac{d}{n}\log\frac{n.d}{a.b} \tag{6.14}$$

6.4 Term Dependency Tree

Let $\mathbf{T_q} = \{t_1, t_2, ..., t_q\}$ be the set of $q \ll m$ (m is the number terms or the size of vocabulary) most relevant terms according to a ranking measure ϕ, such as information gain. By estimating pair-wise inclusion index of T_q, the *term dependency matrix* is obtained. Table (6.2) depicts an example of term dependency matrix for six terms. The objective is to mine the matrix and extract the dependency links between the terms. Given the dependency links, we are able to identify term redundancies and correlations. At the final stage, using a criterion, we make a decision about retaining or removing (merging) the redundant (correlated) terms.

Table 6.2: Term dependency matrix.

	rec	hockei	motorcycl	bike	nhl	playoff
rec	0	0.0988	0.0995	0.0333	0.0139	0.0140
hockei	0.4426	0	0	0	0.1366	0.1240
motorcycl	0.4372	0	0	0.2891	0	0
bike	0.2839	0	0.5609	0	0	0
nhl	0.2067	0.4530	0	0	0	0.0460
playoff	0.2059	0.4067	0	0	0.0455	0

Table 6.3: Adjacency matrix representing the term dependency tree in Figure (6.3).

	rec	hockei	motorcycl	bike	nhl	playoff
rec	0	0	0	0	0	0
hockei	1	0	0	0	0	0
motorcycl	1	0	0	0	0	0
bike	0	0	1	0	0	0
nhl	0	1	0	0	0	0
playoff	0	1	0	0	0	0

A graph representation is also proposed for visualizing the dependency links between terms. *Term Dependency Tree (TDT)* is a rooted, directed, incomplete, and acyclic graph in which both vertices or nodes of any edge are assigned to terms such as $t_1 =$ "*hockei*" and $t_2 =$ "*nhl*". An edge, connecting t_1 to t_2, states that t_2 is dependent on t_1 (or t_1 includes t_2). Figure (6.3) shows an example. The direction of each edge depends on the value of $\theta(t_i, t_j)$ and $\theta(t_j, t_i)$. If $\theta(t_i, t_j) < \theta(t_j, t_i)$ then the direction is from j^{th} to i^{th} node, otherwise the direction is reversed. In Algorithm (1), the process of generating an adjacency matrix, which represents a term dependency tree, is detailed. Table (6.3) illustrates the adjacency matrix of the dependency matrix in Table (6.2), and in Figure (6.3), the corresponding term dependency tree is demonstrated.

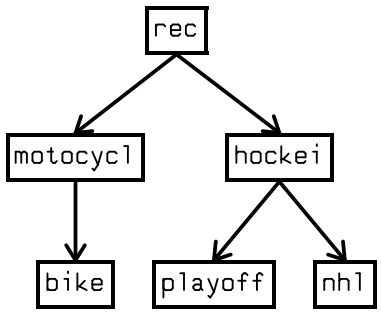
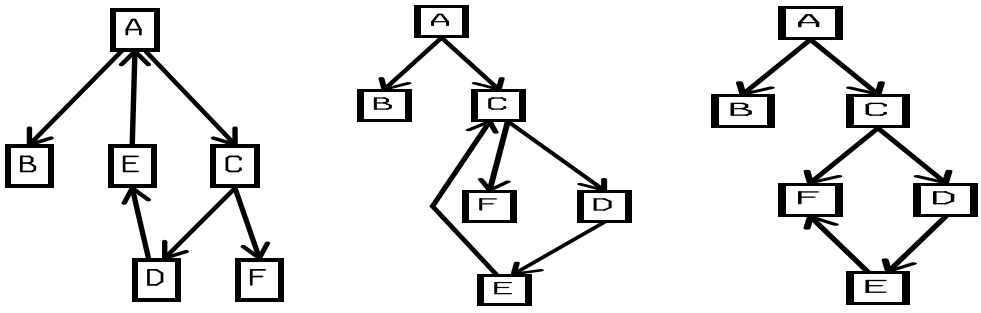


Figure 6.3: Term dependency tree.



(a) (b) (c)

Figure 6.4: Cycles and Pseudo-cycle.

6.4.1 Graph De-cycling

The major attribute of a term dependency structure is its acyclic property. In rare cases, the Algorithm (1) fails to construct a cycle-free graph. Algorithm (2) presents a general algorithm to detect a cycle in a graph. In the case of term dependency tree, we face the following situations:

- Cycle type 1: The graph has no root (Figure (6.4-a)). In term dependency tree, the node with zero indegree, by which any other node in the tree is reachable, is labled as the *root*. To ensure that the generated graph has a root, it should have at least one node with zero indegree. To resolve this problem, the node with maximum coverage is chosen as root and all incoming edges to this node are disconnected (see Algorithm (3)). In the case of having more than one node with zero indegree, the graph includes some isolated sub-graphs.

- Cycle type 2 and Pseudo-cycle: There is at least one node with $indegree > 1$. This problem either causes a cycle (Figure (6.4-b)) or a Pseudo-cycle (Figure (6.4-c)). Both cases are treated as a cycle. By identifying the node with $indegree > 1$, the weaker link(s) is disconnected. The weaker link is the one which exchanges less information (dependency) between the two terms. In Algorithm (4), the approach to fix this problem is detailed.

Algorithm 1 Extracting adjacency matrix from term dependency matrix.

Require: $\mathbf{\Theta} = \{0 \leq \theta(i,j) \leq 1 | 1 \leq i,j \leq q\}$: term dependency matrix.
Ensure: $\mathbf{A} = \{a(i,j) \in \{0,1\} | 1 \leq i,j \leq q\}$: adjacency matrix.

for all $1 \leq i,j \leq q$: $a(i,j) \leftarrow 0$
$Z \leftarrow 1$
while $Z \neq 0$ **do**
    for all $1 \leq j \leq q$: $SUM(j) \leftarrow \sum_{i=1}^{q} \theta(i,j)$
    $x \leftarrow \arg\min_{i=1}^{q}$ [Non-zero elements of SUM]
    $y \leftarrow \arg\max_{i=1}^{q} [\theta(i,x)]$
    $a(x,y) \leftarrow 1$
    for all $1 \leq i \leq q$: $\theta(x,i) \leftarrow 0$ and $\theta(i,x) \leftarrow 0$
    $Z \leftarrow$ # Non-zero elements of $\mathbf{\Theta}$
end while

Algorithm 2 General algorithm to detect cycles in the adjacency matrix.

Require: $\mathbf{A} = \{a(i,j) \in \{0,1\} | 1 \leq i,j \leq q\}$: adjacency matrix.

$\mathbf{B} \leftarrow \mathbf{A}$
if the depth of the graph is given **then**
    $q \leftarrow depth$
end if
for $i = 2$ to q **do**
    $\mathbf{B} \leftarrow \mathbf{B} \times \mathbf{A}$
end for
$N \leftarrow$ # non-zero elements in Diag($\mathbf{B}$)
if $N > 0$ **then**
    there is at least one cycle in the adjacency matrix $\mathbf{A}$.
end if

Algorithm 3 Finding a root for the extracted graph (No node with zero indegree).

Require: $\mathbf{\Theta} = \{0 \leq \theta(i,j) \leq 1 | 1 \leq i,j \leq q\}$: term dependency matrix.
$\mathbf{A} = \{a(i,j) \in \{0,1\} | 1 \leq i,j \leq q\}$: adjacency matrix.
Ensure: $\mathbf{A} = \{a(i,j) \in \{0,1\} | 1 \leq i,j \leq q\}$: rooted adjacency matrix.

for all $1 \leq j \leq q$: $SUM(j) \leftarrow \sum_{i=1}^{q} \theta(i,j)$
$K \leftarrow \arg\max_{i=1}^{q} [SUM(i)]$
for all $1 \leq j \leq q$: $a(k,j) \leftarrow 0$

Algorithm 4 De-cycling and removing pseudo-cycles in the extracted graph.

Require: $\mathbf{\Theta} = \{0 \leq \theta(i,j) \leq 1 | 1 \leq i,j \leq q\}$: term dependency matrix.
$\mathbf{A} = \{a(i,j) \in \{0,1\} | 1 \leq i,j \leq q\}$: adjacency matrix.
Ensure: $\mathbf{A} = \{a(i,j) \in \{0,1\} | 1 \leq i,j \leq q\}$: de-cycled adjacency matrix.

1: $cycle \leftarrow TRUE$
2: **while** *cycle* **do**
3: for all $1 \leq i \leq q$: $indegree(i) \leftarrow \sum_{j=1}^{q} a(i,j)$
4: $k \leftarrow$ find the first element with $indegree > 1$
5: **if** k={} **then**
6: $cycle \leftarrow FALSE$
7: **else**
8: $l \leftarrow \arg\max_{i=1}^{q} [\theta(i,k)]$
9: $a(k,j) \leftarrow 0$ for all $1 \leq j \leq 1$ except j=l
10: **end if**
11: **end while**

6.5 Substitution Cost

Let $\mathbf{T} = \{t_1, t_2, ..., t_m\}$ be the sorted terms of the vocabulary according to a feature ranking measure ϕ such that $\phi(t_1) \geq \phi(t_2) \geq ... \geq \phi(t_m)$. Let $\mathbf{T_q} \subset \mathbf{T}$ be the selected features including the $q \ll m$ most relevant terms. q is the ranking threshold or the desired number of features.

In feature ranking-based feature selection, the search for redundant and correlated terms is sequential and based on individual terms. Let t_j, $1 \leq j \leq q$, with the ranking measure $\phi(t_j)$ be a redundant (correlated) term to be removed (to be merged). By removing (or merging) t_j, the length of T_q is decreased to $q-1$. To have q features, t_{q+1} with ranking measure $\phi(t_{q+1})$ is added to T_q. In order to provide an objective measure to facilitate the decision making about term removal, *Substitution Cost* is defined as the cost of substituting a relevant but redundant (or correlated) term with a less relevant term.

$$\Delta_\phi(t_j) \equiv \phi(t_j) - \phi(t_{q+1}) \tag{6.15}$$

where $\Delta_\phi(t_j)$ = the cost of removing or merging the term t_j.

A high substitution cost addresses a risky redundant term removal (or correlated term merging), because by substituting the term with a poor term in terms of information and relevance, the total performance of the feature vector can be reduced.

The main reason can be understood intuitively as follows: Referring to Figure (6.5), illustrating the sorted information gain for the first 100 best terms, when q is less than

Algorithm 5 Redundancy reduction algorithm.

Require: $\mathbf{\Theta} = \{0 \leq \theta(i,j) \leq 1 | 1 \leq i,j \leq q\}$: term dependency matrix.
$\mathbf{A} = \{a(i,j) \in \{0,1\} | 1 \leq i,j \leq q\}$: adjacency matrix.
$T = \{t(i) | 1 \leq i \leq m\}$: ranked set of terms
$\Phi = \{\phi(i) | 1 \leq i \leq m\}$: sorted term scores (ranking measures)
$q \ll m$: the desired number of terms
$SubstitutionThreshold > 0$: substitution cost threshold
$CorrelationThreshold > 0$: correlation threshold
Ensure: T_q: the set of selected, non-redundant terms.

for all $1 \leq j \leq q$: $outdegree(j) \leftarrow \sum_{i=1}^{q} a(i,j)$
$DependentSet \leftarrow \arg_{i=1}^{q}[outdegree(i) \leq 1]$
$T_q \leftarrow t(1)$
$k \leftarrow 2$
while length of $T_q < q$ **do**
  $SubstitutionCost \leftarrow \phi(k) - \phi(k+q)$
  $l = \arg_{j=1}^{q}[a(k,j) = 1]$: $t(l)$ is the node connected to $t(k)$
  if [$t(l) \notin DependentSet$] or [$SubstitutionCost > SubstitutionThreshold$] **then**
    $T_q \leftarrow T_q + \{t(k)\}$
  else if $|\theta(l,k) - \theta(k,l)| < CorrelationThreshold$ **then**
    merge $t(k)$ with $t(l)$: $t(k)$ and $t(l)$ are correlated
    $T \leftarrow T - \{t(k)\}$
  else
    $T \leftarrow T - \{t(k)\}$: $t(k)$ is redundant
  end if
  $k \leftarrow k + 1$
end while

10, term redundancy reduction is being held in the sharp slope region of the curve (for example, between points "A" and "B"). It means that by removing a redundant term from the feature vector, most likely a much less informative term will be substituted (t_{q+1} with feature ranking measure $\phi(t_{q+1})$), but in the case of working in a smooth region of the curve (for example, between points "B" and "C"), the proposed method may outperform information gain. As a result, according to this figure, removing or merging any term in areas between "A" and "B", and also between "C" and "D" is costly, since the graph shows a sharp decline in these areas. On the contrary, depending on q, we probably can remove the redundant terms from "B" to "C" and "D" to "E".

The substitution cost can be compared with a threshold such as τ_s to make a decision on redundancy term removal or merging. The removal or merging is performed if $\Delta_\phi(t_i) < \tau_s$. In this thesis, we heuristically use the average feature ranking measure as

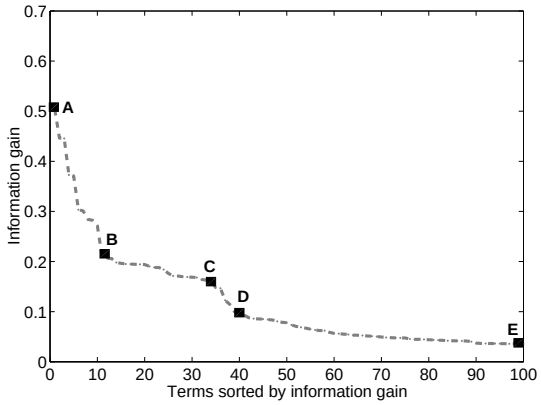


Figure 6.5: Sorted information gain for the 100 best terms.

the substitution cost threshold:

$$\tau_s = \frac{1}{q}\sum_{i=1}^{q}\phi(t_i) \tag{6.16}$$

6.6 Redundancy Reduction Algorithm

Term dependency tree and dependency matrix provide a framework to evaluate the features based on their dependencies to each other. Figures (6.6) to (6.9) illustrate the dependency tree of the data sets employed in this chapter. The algorithm (see Algorithm (5)) for redundancy reduction is performed in the following steps:

1. Using the feature ranking measure ϕ and the ranking threshold q, the q best terms are selected. By estimating pair-wise inclusion index of the terms, and using Algorithm (1), the term dependency tree is extracted. The nodes of the tree are clustered into the *Links* and *Hubs* based on their measure of outdegree:

   - *Link nodes*: the nodes that their outdegree is equal to one.

   $$\forall\ 1 \leq j \leq q,\ \ t_j \in \mathbf{T_q},\ \ outdegree(t_j) = 1 : t_j \in LINKS \tag{6.17}$$

   - *Hub nodes*: the nodes that their outdegree is greater than one.

   $$\forall\ 1 \leq j \leq q,\ \ t_j \in \mathbf{T_q},\ \ outdegree(t_j) > 1 : t_j \in HUBS \tag{6.18}$$

For example, in Figure (6.8), the set of hub and link nodes are as follows:

$$hubs = \{shr, japan, market\}$$
$$links = \{acquir, corp, rev, note, ct, net, japanese, trade, export, oil, rate, bank, franc, stg\}$$

A link node is most likely redundant, because it shares its maximum information with only one child node. On the contrary, a hub node shares its information with many child nodes. In a classification task, each of these child nodes might be relevant to a particular class. As a result, hub nodes are hardly redundant. It should be noted that the indegree of all nodes is one, with the exception of the root whose indegree is zero.

2. Starting from the highest ranked term t_1 according to the feature ranking measure ϕ, a link node is redundant if it satisfies the substitution cost criterion.

3. Before removing a redundant term, we should ensure that it is not correlated with its co-dependent term. Let t_k be a candidate redundant term. Let t_l be its co-dependent term, or parent node in the term dependency tree. By introducing *correlation threshold* τ_c, the correlation between the two terms t_k and t_l can be approximately evaluated. If $|\theta(t_k, t_l) - \theta(t_l, t_k)| \leq \tau_c$, they are most likely correlated. In this case, instead of removing t_k, it can be merged with t_l to make a multi-word term. Otherwise, the term t_k is removed from the set of terms. In this thesis, τ_c is empirically set to $0.1 \times \tau_s$.

6.7 Experimental Results

The proposed approach is applied to four data sets including: Industry Sectors, 20 Newsgroups, Reuters, and WebKB using SVM with linear kernel and Rocchio text classifiers. Recently, SVM has outperformed most classifiers in text categorization [43, 26]. There are some reports showing that feature selection for SVM classifiers is not only unnecessary but also can reduce the performance [95, 43]. However, in this chapter, we show that for a very small size of feature vector, SVM performance can be improved by feature selection through redundancy reduction [26].

The proposed approach has been evaluated by comparing its results with those of stand-alone information gain ranking (without redundancy reduction). A five-fold cross

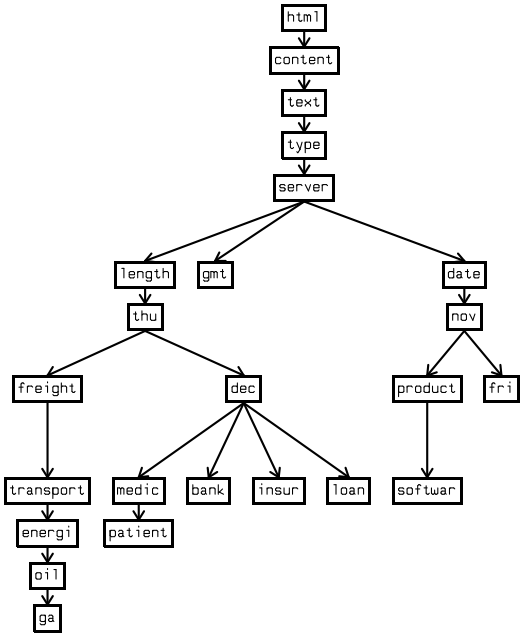


Figure 6.6: Term dependency tree of first 0.2% of terms for Industry Sectors data.

validation is used for better estimation of classifier performance. Each method has been applied to the SVM and Rocchio classifiers with 10 levels of aggressive feature selection including $\{0.002, 0.004, ..., 0.02\} \times m$. It is worthwhile to mention that in *aggressive feature selection*, the number of features is drastically reduced, and only 2% to 5% of features are retained [95].

The performance of both classifiers are presented in Figure (6.10) and (6.11). Redundancy reduction improves the Rocchio classifier performance in most data sets. Figure (6.11) shows that the result of redundancy reduction method is better than that of information gain ranking. In the 20 Newsgroups data set, both methods perform similarly for the first levels of filtering. This can be explained by the notion of substitution cost. In 20 Newsgroups, the sorted information gain starts with a sharp decline (see Figure (6.5)), in which any redundancy reduction cannot be affordable. In the smooth region of the curve ("B" to "C"), redundancy reduction can outperform the information gain ranking.

Generally speaking, in the 20 Newsgroups data, the redundancy reduction method for both classifiers cannot offer better results compared to the information gain ranking. The main reason can be explained as follows: The 20 Newsgroups seems an easy data set. It includes 20 uniformly distributed classes and is a homogenous data set, such that the distribution of features in classes is almost uniform. It means that in a very small feature

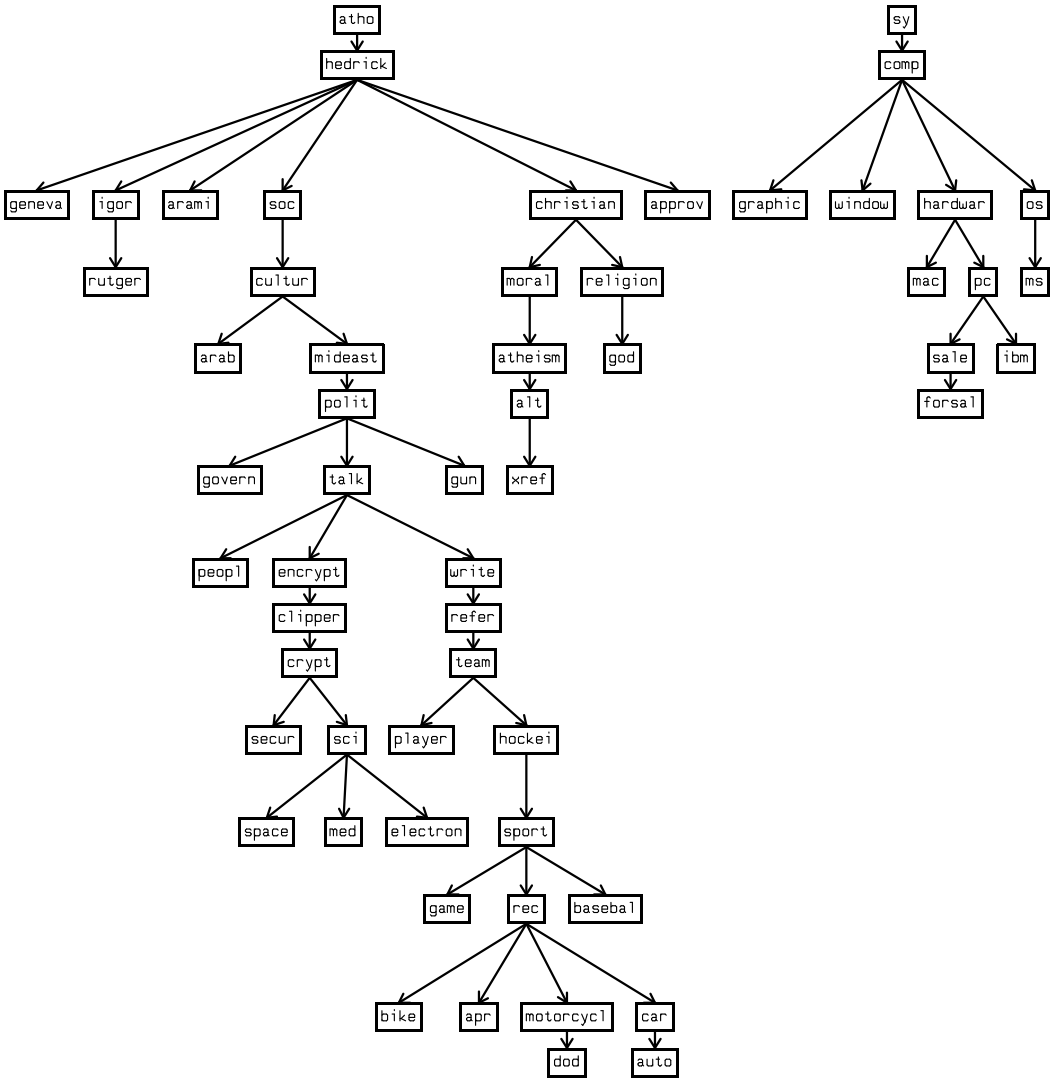


Figure 6.7: Term dependency tree of first 0.2% of terms for 20 Newsgroups data.

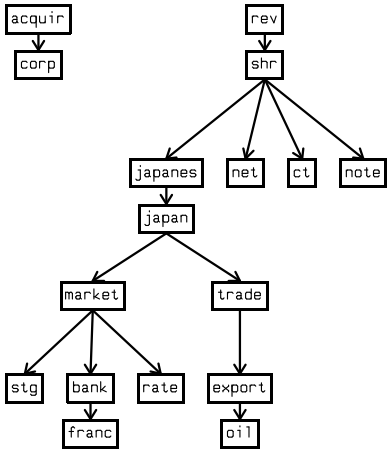


Figure 6.8: Term dependency tree of first 0.2% of terms for Reuters data.

vector (aggressive feature selection), the terms are uniformly scattered over all classes. In this case, the terms are rarely redundant.

6.8 Conclusion

In aggressive feature selection, the length of feature vector is quite short. The text classifiers, working with very small feature vectors, are very sensitive to noise and redundancies. Because of these restrictions, improving any classical feature selection method, such as feature ranking for aggressive reduction is absolutely necessary. Feature ranking methods, such as information gain and χ^2, ignore term dependencies (including correlation and redundancy), which can cause serious complications.

To deal with redundancy, a method for improving aggressive feature selection by feature ranking for text classifiers was proposed. The method was based on extracting and removing term redundancy using an information theoretic inclusion measure. The pairwise inclusion measures are used to estimate the dependency link between terms. A heuristic technique was proposed to search the links for redundancy and correlation. Finally, using the correlation threshold, we can either remove the redundant term or merge it with the correlated one. The proposed approach and the stand-alone IG ranking were compared in SVM and Rocchio text classifier frameworks. Results showed that the proposed approach outperformed the aggressive feature selection by the stand-alone information gain for most data sets with a weak classifier such as Rocchio.

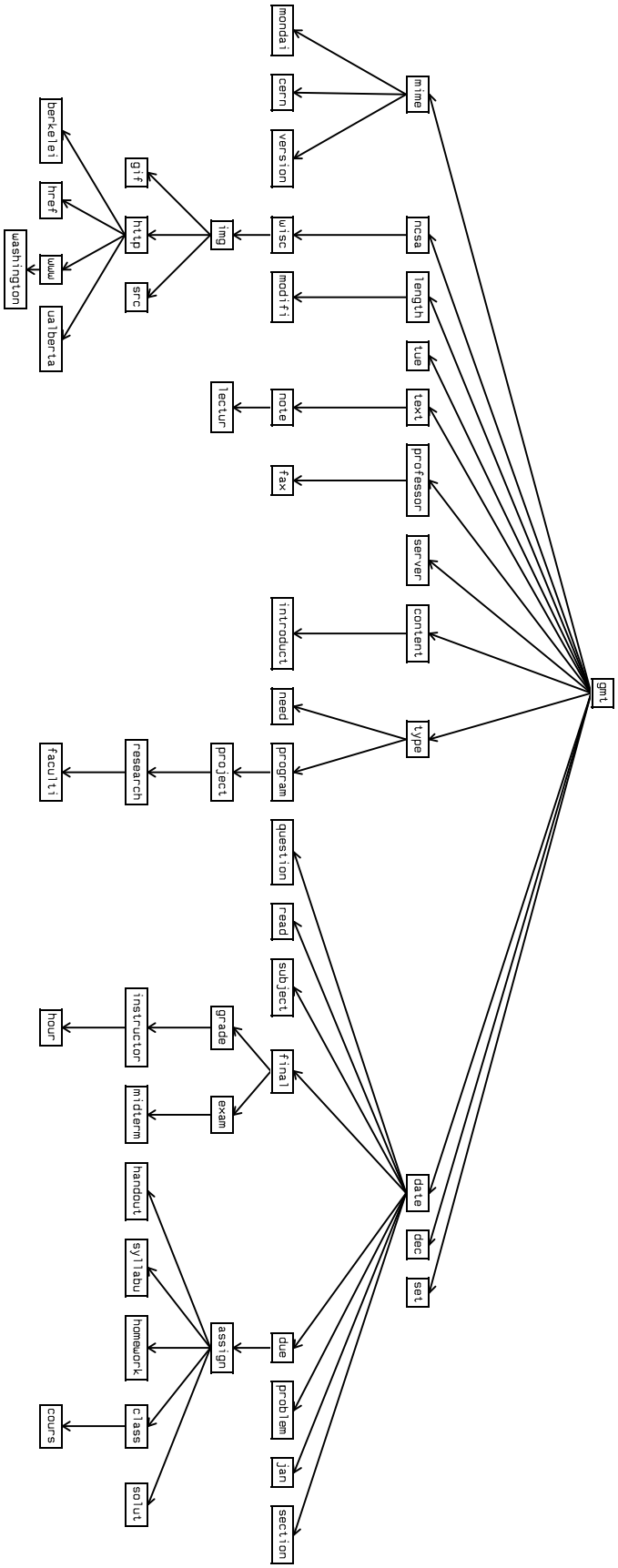


Figure 6.9: Term dependency tree of first 0.2% of terms for WebKB data.

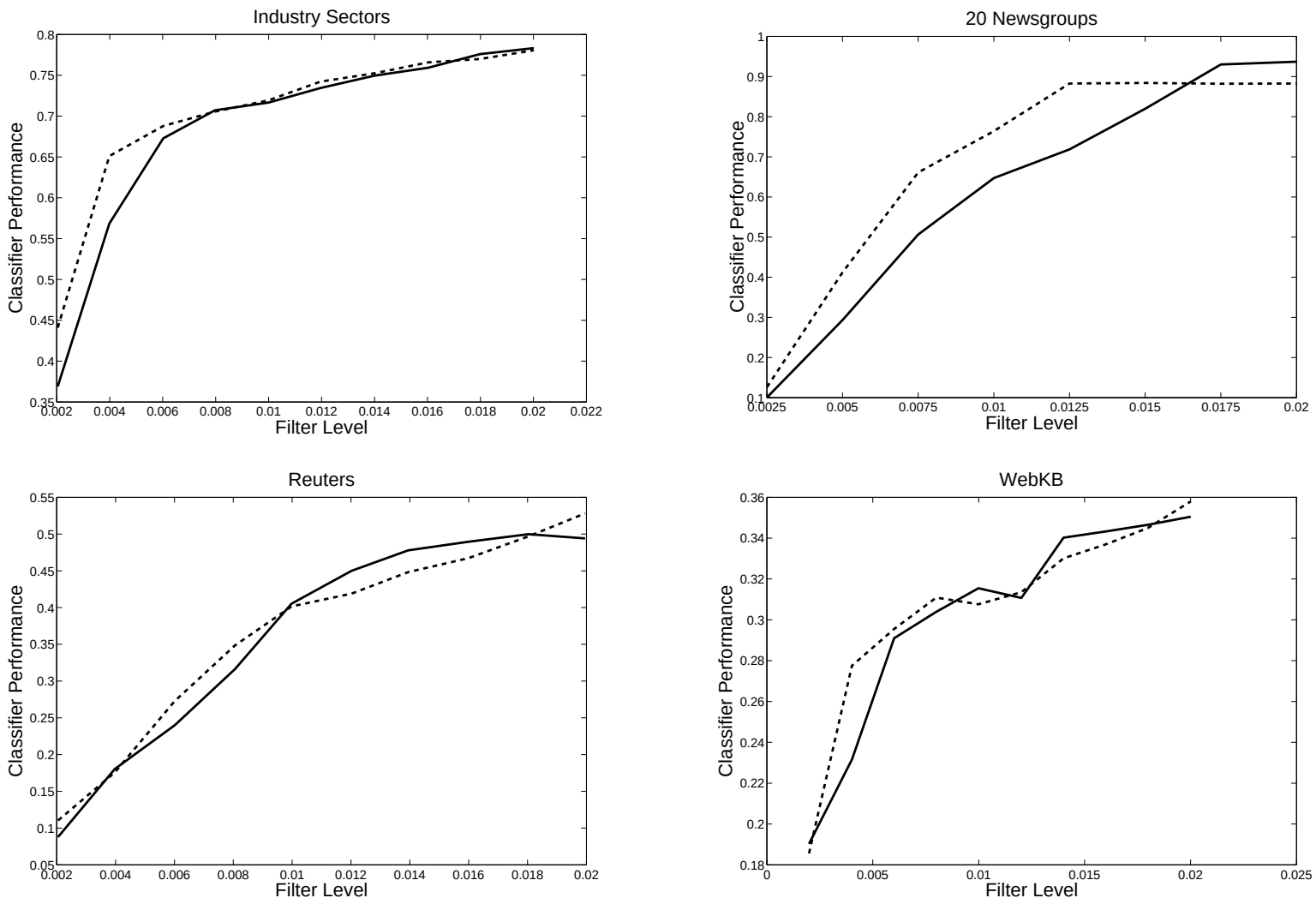


Figure 6.10: SVM classifier performance vs. the number of features for two aggressive feature selection methods. Dashed lines represent the proposed method and solid lines represent information gain feature ranking.

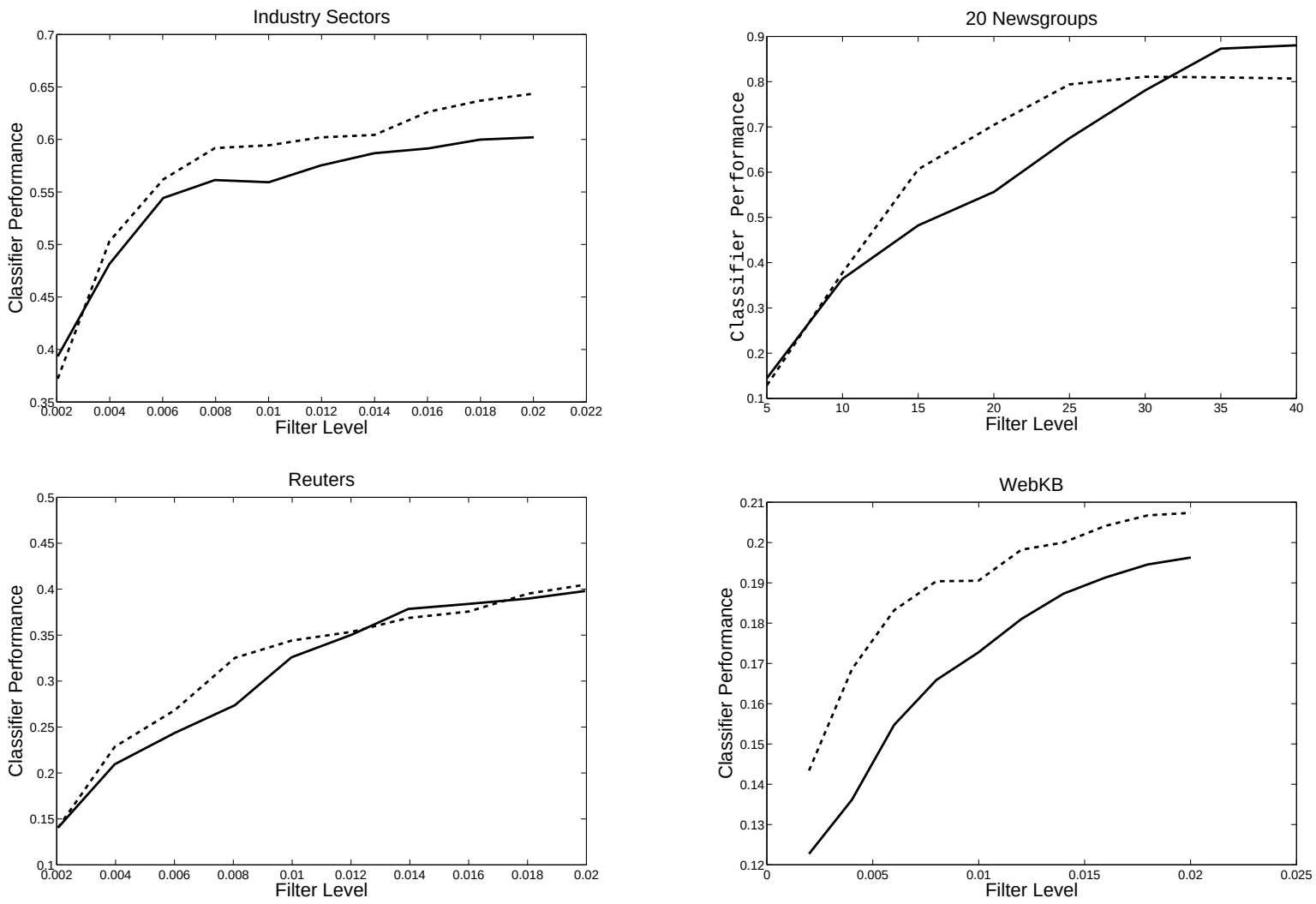


Figure 6.11: Rocchio classifier performance vs. the number of features for two aggressive feature selection methods. Dashed lines represent the proposed method and solid lines represent information gain feature ranking.

Chapter 7

Local Feature Ranking for Unbalanced Data

7.1 Introduction

From Chapter 3, the majority of feature ranking methods fail when applied to multiple class problems with non-uniform class distributions. According to [23], feature ranking methods can select relevant features for simple classes, while being unable to learn difficult or small classes. They only pay more attention to the easy and large classes to compensate for their weakness against the difficult classes. By means of this trick, they improve their overall accuracy even though there is a large negative number of results for small and difficult classes.

One potential approach to decrease the destructive impact of imbalance is to make use of local feature ranking instead of a global scheme [23]. In this chapter, this approach is detailed and applied to an extremely unbalanced data, which is the problem of social network extraction from textual data.

7.2 Social Networks

A social network is defined as a map of relationship between individuals. Relationships and individuals involved in a social network are also called ties and actors (nodes or members), respectively. In social network analysis, the point of focus is ties rather than actors. In order to have an actual social network, its size should be limited to around 150 actors. This number, the so-called Dunbar's number, comes from the idea of the

maximum size of a village in sociology [29].

Social networks on the Internet use a web site to connect people together who share their ideas about personal or professional interests. Unlike real social networks, the size of social networks in a virtual world can number in the thousands. During the last three years, social networks such as orkut[1], Facebook[2], Friendster [3], LinkedIn[4], hi5network[5], MySpace[6], and Yahoo!360[7], have had substantial growth in terms of web traffic. The majority of these networks are for personal and socialization purposes. However, they are interesting for marketing and advertising due to their exponentially increasing traffic.

Unlike the virtual networks, which are for public, with broad commercial applications, the Friend Of A Friend (FOAF) project is initiated by professionals in the semantic web and information technology community [108]. FOAF is merely a metadata standard for describing people and their relationship, and to build a social network or online community [18]. FOAF is one of the semantic web initiatives attempting to build a social network framework. Similar to the other types of metadata, a FOAF file is data about an object, in this case an individual person in a community. One important part of FOAF is the friend-list, including the name of friends and their contact information [18].

Finding a person's friends' network is a more personal and private issue. However, in a small community such as a virtual classroom in an e-learning system, by creating the social network online, we can offer the students a list of individuals of similar interest who can share their knowledge, questions, comments and interests towards educational matters. Scenarios might include preparing a course paper, developing a course note or getting feedback about a lecture. In all cases, the system can provide the user a list of potential friends who can help her to perform the task.

This thesis proposes an approach to generating automatically a social network from a collection of web documents. In order to generate (semi)automatically a social network, we need to represent each actor (individual person) by a set of features or attributes. Using web resources, every person can be represented by her corresponding documents, which is represented by a vector space model. Using vector space document representation, each person is described by a set of single-word terms from the vocabulary. As-

[1]www.orkut.com/
[2]www.facebook.com
[3]www.friendster.com/
[4]www.linkedin.com/
[5]www.hi5network.com/
[6]www.myspace.com/
[7]360.yahoo.com/

sociating people in the community to the terms in the vocabulary, the new structure is called *actor-term matrix*. Similar to the document-term model, the new model suffers from high dimensionality of the vocabulary. By a set of preprocessing tasks such as stemming, stopword removal and document frequency threshold, the vocabulary is downsized up to 80%.

The next step is learning social relations from the actor-term data base. Similar to other machine learning applications, if there is any training data, a social network can be extracted using a supervised learning or a classification approach. In this case, the training data is a set of known relationships among some actors. If no pair-wise relation is known, the learning is unsupervised, which is based on pair-wise similarities and clustering.

In this thesis, we assume that the social network is partially explored. Using the revealed relations in the social network as training data, a classifier is employed to extract the missing relations to complete the social network.

7.3 Related Work

Recent researches on machine learning and data mining has provided developed methods and algorithms to construct statistical models of network data, including social networks, web-page networks, email tracks, citation networks, and so on. The models can be constructed either directly from data using information extraction algorithms, which are applied mostly on structure data (for example, relational databases) and semi-structured data (for example, XML data), or indirectly from unstructured textual data using text mining techniques.

In the first case, the concept of acquaintanceship (or a link between two actors) can be extracted from information in the data object, for example, the receiver of an email, the author of a cited paper, a web page link, and "knows" relationship of FOAF vocabulary in an FOAF file. In this case, the extracted network is mostly descriptive rather than predictive. In descriptive social network problems, the goal is to visualize the extracted network for analysis and tracking purposes [41, 17]. One interesting work was exploring the social network in Enron by tracking the 1.5 million messages [17]. In 2003, the US Federal Energy Regulatory Commission posted the company's e-mail on its Web site. After removing duplicates, the data included a half-million e-mails from about 161 users, including those of the company's top executives.

In the second case, which is predictive, acquaintanceship is translated into the similarity of two actors. This pair-wise similarity is extracted from the textual resources of the

actors. The resources can be articles, papers, news, resumes, CVs, and so on [79]. In [91], social networks are generated by mining knowledge-sharing sites to support generating a marketing plan. In the knowledge-sharing networks, customers share their opinions with others. The social network is generated by a probabilistic model of the network.

FOAF, as a semantic web initiative with RDF format, is a vocabulary to create machine readable information about people. One element of the FOAF vocabulary is *knows* triple, by which the owner of FOAF describes her friends. In [74], this relationship is extended to achieve a context aware social network by involving the concepts as well as actors. The relationship between actors and concepts is estimated by Google web page frequency [108, 73]. In [46], using text-based segmentation of chat-room conversations, social and contextual environment of the given chat-rooms are analyzed.

Data mining has also been used for mining social networks. For instance, in [70], using the influence diffusion model, the social networks in a message board are extracted. The model is simply based on the frequency of terms which are propagated between two individuals. The social networks are categorized to study some social and psychological issues such as interactivity among members. Another approach in extracting social networks is using usage and log data instead of textual contents. One application is exploring the social networks in instant messaging systems to study the network related issues such as system traffic [90].

In [71], the number of retrieved web pages, including the names of both actors, is considered as a degree of acquaintanceship to build a social context. The method has been applied to extract social relationships of conference participants. One drawback of querying the name of network members in a search engine is the personal identification problem. For social network extraction we need to assign a unique identity to each actor, given that many individuals may have same identities and a person may be known by more than one name. Using FOAF person metadata, this problem is resolved because each member is associated with an Uniform Resource Identifier (URI), which cannot be redundant. Acquaintanceships and ties in a social network might also be extracted using citation, web link, and co-authorship. In [34], the social network of a software reverse engineering community is extracted from co-authorship graph. It shows that the community behaves like a *small world* [20].

7.4 Problem Statement

A social network is considered as a descriptive framework to study and analyze social relations and measure social metrics. In this thesis, our view of social networks is more predictive rather than descriptive. The problem is, in the lack of an explicit social network, how we can predict and learn the network while knowing only a small number of relations between actors. In other words, the goal is to complete a partially known social network. The set of known relations can be obtained using questionnaires, FOAF files, and searching web documents using information extraction techniques. For example, using BibTex metadata, we can extract co-authorship, which is a strong tie between two or more individuals.

Let $\mathbf{A} = \{a_1, a_2, ..., a_n\}$ be a finite set of n actors, who are interconnected through a finite set of relations or ties. These interacting actors form a connected social network or some isolated sub-networks. Since we assume the social relations to be symmetric, there are $M = n(n-1)/2$ possible ties among the actors. Practically, the set of existing relations is a very small subset of the possible relations. This fact addresses an important metric of social network called sparsity:

$$S_N = 1 - \frac{2n_r}{n(n-1)} \tag{7.1}$$

where $n_r =$ the number of ties in the network.

Social networks are usually very sparse graphs. High sparsity can create isolated sub-networks or even isolated actors. On the other hand, low sparsity increases the density of the network, in which every actor is connected to the others. In this case, the network seems to have no social benefits for its members.

Social networks are represented either by graphs called socio-graph or matrices called socio-matrix (adjacency matrix). Let $\mathbf{R}_T = \{r_k \in \{0,1\} | 0 \leq k \leq q\}$, $q \ll M$ be the incomplete set of known ties among the actors of $\mathbf{A}$. The objective is, using the textual data $\mathbb{D} = \{\mathbf{D_1}, \mathbf{D_2}, ..., \mathbf{D_n}\}$, which are associated with the actors, to learn new relations in order to complete and enrich the social network(s). In other words, the adjacency matrix $\mathbb{A}_T$ is given (Table (7.1-A)), the goal is learning the adjacency matrix $\mathbb{A}_E$, which is the complete social network (Table (7.1-B)).

Table 7.1: Two socio-matrices: (a) incomplete matrix (training examples), (b) complete matrix (learned socio-matrix).

(A)					(B)				
0	?	0	?	1	0	1	0	1	1
?	0	?	?	0	1	0	0	0	0
0	?	0	1	?	0	0	0	1	1
?	?	1	0	0	1	0	1	0	0
1	0	?	0	0	1	0	1	0	0

7.5 Learning Social Networks from Unbalanced Text Data

Learning a social network is to explore the relationships among the actors of a community. In addition to the structural methods, which require structured data, statistical approaches such as data mining techniques may provide more comprehensive frameworks by which to extract social structures. By employing text mining techniques, the social network generation system may learn non-trivial patterns, similarities, and association among the actors in the community.

In order to extract information about people, we need sufficient resources from or about them. The information may include news, personal home pages, web logs, publications, resumes, and so on. In the proposed approach, we assume that two input data are available:

1. a subset of relations represented by $\mathbf{R}_T$;
2. the textual data $\mathbb{D}$ associated with the actors.

Learning social networks is performed in three steps:

1. actor representation;
2. relationship representation;
3. classifier design to learn unknown relations of the network.

7.5.1 Actor Representation

In this approach, each actor is represented by her web documents, including home page, blog, CV and so on. All documents associated with an individual are merged together to

build a unique document vector. Every document $\mathbf{D}_i \in \mathbb{D}$ is associated with an individual actor where $\mathbb{D}$ is the actor-term matrix. Using the vector space model, each actor is represented by a set of single-word terms as follows:

$$\mathbf{D}_i = \{ w_{i,j} t_j | 1 \leq i \leq n, 1 \leq j \leq m \} \tag{7.2}$$

where $w_{i,j}$ = the weight of term j in the web documents associated with the actor i;
t_i = the i^{th} term in the vocabulary $\mathbf{T}$.

The weight $w_{i,j}$ is calculated by a weighting technique. All weighting techniques include two parts, global and local weighting. The global weighting assigns a weight to the term based on its significance across the collection, while the local one estimates the importance of the term in a particular document. In this research, the $TF \times IDF$ weighting scheme is used. It consists of a TF local weighting and an IDF global weighting. A normalized version of $TF \times IDF$ is as follows

$$TF \times IDF(i,j) = \frac{TF_{i,j}}{log\frac{n(t_j)}{n}} / \sqrt{\sum_{i=1}^{n(\mathbf{D}_i)} (\frac{TF_{i,j}}{log\frac{n(t_j)}{n}})^2} \tag{7.3}$$

where $n(t_j)$ = the number of documents in which the term j appears;
$TF_{i,j}$ = the number of occurrences of the j^{th} term within the i^{th} document;
$n(\mathbf{D}_i)$ = the length of the document vector $\mathbf{D}_i$ associated with the actor a_i.

After performing two preprocessing tasks, including stemming and general stopword reduction, two DF thresholds are applied. All terms with document frequency less than 5 and more than 100 are removed.

Another approach to representing the actors is Latent Semantic Indexing (LSI) or generalized vector space model [64]. LSI has been inspired from Singular Value Decomposition (SVD) to capture major associative patterns in the data. LSI not only reflects the significant associations, which can be interpreted as semantic relations, but also ignores minor influences. By applying this technique, an effective dimensionality reduction is achieved in addition to extracting more relevant features at the same time [16]. The other approach is to represent each actor with its associated keywords. In this approach, we need to extract keywords from the corresponding documents [79].

7.5.2 Relationship Representation

One simple approach to representing the relation between two actors is to estimate the similarity of their document vectors. The similarity measure is calculated by various

formulations such as Cosine, Jaccard, and Correlation [79, 64]. This approach offers very poor results because it describes each relation by only one feature, which is not enough for this classification problem.

A better approach is to aggregate the document vectors of the actors in both sides of the relation and create a new aggregated vector. By this operation, the new vector represents the relation instead of the actors. Let $\mathbf{D}_i$ and $\mathbf{D}_j$ be the document vectors associated with the actors a_i and a_j, respectively. The relation between the two actors is represented by their aggregated vectors. The vectors are aggregated by an operator such as MIN, MAX, or Product. While working with binary features, operators such as AND or OR may also be used. Here, due to the use of weighted features, a MAX operator is employed to generate aggregated vectors.

We call the aggregated vector *relation vector*. The relation vector is obtained as follows:

$$\mathbf{W}_{i,j} = \{\ max(w_{i,k}, w_{j,k}) | 1 \leq i, j \leq n, 1 \leq k \leq m\} \tag{7.4}$$

The vector aggregation using MAX operator offers better results compared to MIN and Product. The reason can be explained as follows: When applying MAX operator on two vectors, a resulting, sparse vector shows the closeness and similarity of two aggregated document vectors. On the other hand, a scattered result reflects existing dissimilarity and difference between the two vectors. As a result, this model offers a more discriminating feature vector, which is required to design the classifier. Therefore, MIN and Product cannot generate discriminating feature vectors, because they tend to represent more compact vectors.

7.5.3 Classifier Design for Imbalance Social Network Data

Given n actors in a social network with symmetric relations, they can be connected to each other by $M = n(n-1)/2$ possible relations. The problem of learning social networks is a search problem in this large space. Using vector space model representation, each relation is modeled by a vector of m terms. Since by this formulation, every relation is represented by an aggregated document vector, the problem of learning social networks is translated into a text classification problem.

From (7.4), we define relation-term matrix $\mathbb{W} \in \mathbb{R}^{M \times m}$, in which every column is the distribution of the term across the relation space. In a classification model, the label of each relation vector can be either 1 or 0 addressing *relation* (connected link) or *no relation* (broken link), respectively.

Let $\mathbb{W}$ be the relation space modeled by document aggregation, and $\mathcal{R} = \{0, 1\}$ be the set of classes. Let us suppose there is a classifier such as h, which is defined as

$$h : \mathbb{W} \rightarrow \mathcal{R} \tag{7.5}$$

where $h =$ a many-to-one mapping of the documents to the binary class space.

As labels of training data, the labels of q relations out of M possible relations are given ($q \ll M$). The label of each relation is either $connected = 1$ or $broken = 0$, which are mapped to the positive and negative classes, respectively.

Using a classifier, $M - q$ remaining relations are predicted. Due to one of the essential properties of social networks among M relations, very few are positive or connected, which addresses the sparsity of the social network structure. Since the training data is randomly sampled from the real social network, its sparsity is directly linked to the class imbalance of the classifier. From (7.1), only n_r relations are positive and $M - n_r$ are negative, which means the class imbalance is $n_r : M - n_r$ or approximately $n_r : M$ since $n_r \ll M$.

Learning social networks is a binary class problem. The training data for this problem is extremely unbalanced, such that the data samples are non-uniformly distributed across the two classes. The class imbalance, also known as class skew, may dominate the support vectors in SVM and does not let the samples in the minority class effectively contribute in the training process.

In social network analysis, the focus is mostly on the connected ties or positive classes ($class = 1$). For example, in crime investigation and policing, tracking disease transmission, and other critical issues, investigating the connections is more important than the broken ties. Then class imbalance may hurt the performance of learning social networks, because it ignores the positive examples in the minority class and will focus primarily on the majority class, including negative samples.

A common approach to dealing with class imbalance is to artificially re-balance the training data. Two well-known techniques are up-sampling the minority and down-sampling the majority classes [59, 66]. A multiple-resampling approach can also be applied to improve the classifier performance [68].

In this thesis, we consider the problem of learning social networks as a case study to deal with imbalance data at a feature level rather than data and algorithm levels [111]. In Chapter 3, the impact of feature ranking-based feature selection on the classifier performance of imbalance data was explained. It was empirically stated that the majority of

feature ranking methods fail in the case of unbalanced class distribution. In the next section, we propose a new technique to deal with class imbalance at the feature level using *local feature selection*.

7.6 Reverse Discrimination: Re-balancing Unbalanced Data at Feature Level

The major motivation to re-balance unbalanced data at the feature space level is to improve the performance of feature ranking methods when applied to unbalanced data. In Chapter 3, we empirically stated that, in unbalanced data, feature ranking methods tend to select features from majority and simple classes rather than minority and difficult classes. In other words, feature ranking methods are unable to be fair in the case of class imbalance. By increasing the level of imbalance, the chance of having representative features from the minority class decreases. The consequence is a high misclassification rate of the minority class, which is primarily the target class.

Inspired from social science, *reverse (positive) discrimination* [56] in feature ranking is proposed. From Wikipedia [119]:

> "Affirmative Action refers to concrete steps that are taken to promote access to education or employment aimed at a historically socio-politically non-dominant group; typically people of color or women. Motivation for Affirmative Action policies (sometimes referred to as Positive or Reverse Discrimination) is to redress the effects of past discrimination and to encourage public institutions such as Universities, Hospitals and Police forces to be more representative of the population."

In the reverse discrimination approach, the goal is to design a fair representation of all classes in feature selection using a feature ranking scheme. Simply defined, in the reverse discrimination approach, features are locally ranked and selected for each class. In the next step, for each category and using the corresponding feature vector, a classifier is designed. Finally, the results of all classifiers are combined by an aggregation method such as maximum weight or majority vote [50].

To implement a reverse discrimination-based feature ranking, a local feature ranking measure, such as OR or χ^2, is required. Local feature ranking measures produce a category-term ranking matrix instead of a ranked feature vector. Let ϕ be a local feature ranking measure. We have

$$\mathbb{F} = \begin{bmatrix} \phi(t_1, c_1) & \phi(t_2, c_1) & ... & \phi(t_m, c_1) \\ \phi(t_1, c_2) & \phi(t_2, c_2) & ... & \phi(t_m, c_2) \\ & ... & & \\ \phi(t_1, c_C) & \phi(t_2, c_C) & ... & \phi(t_m, c_C) \end{bmatrix} \tag{7.6}$$

where $\mathbb{F}$ = the category-term ranking matrix.

From Chapter 2, $\phi(t_j, c_k)$ is the weight of feature t_j in class c_k. The features are sorted C times according to their scores in the rows of $\mathbb{F}$. The result includes C feature vectors $\mathbb{V} = \{\mathbf{V}_1, \mathbf{V}_2, ..., \mathbf{V}_C\}$, each of which is suited for one separate class. In the next step, a threshold α_k is applied to each feature vector $\mathbf{V}_k$. Using C different thresholds, we can control the feature distribution over classes. In other words, by choosing appropriate threshold values, an efficiently balanced data might be achieved. Intuitively, a feature vector of a larger class needs a lower threshold, while the vector of a smaller one requires a higher threshold value. In this thesis, we use only one threshold value for all vectors, *i.e.*, $\alpha_1 = \alpha_2 = ... = \alpha_C$.

Let $\mathbf{h} = \{h(V_1^{\alpha_1}), h(V_2^{\alpha_2}), ..., h(V_C^{\alpha_C})\}$ be the set of classifiers. Here, we use same classifier algorithms for all feature vectors. The aggregated classifier result is estimated as follow:

$$h^* = \Gamma(h(V_1^{\alpha_1}), h(V_2^{\alpha_2}), ..., h(V_C^{\alpha_C})) \tag{7.7}$$

where Γ = the aggregation function.

In this research, a maximum operator is employed as the aggregation function.

7.7 Experimental Results

A Rocchio text classifier is applied to learn social relations from a subset of the FOAF data set described in Appendix A. Because of highly unbalanced training data, reverse discrimination-based feature ranking is applied. Figure (7.1) shows the comprehensive results of the classifier using the reverse discrimination (local) approach and the regular (global) feature ranking. The experiments are performed for 200 filter levels. After preprocessing, the total number of terms in the vocabulary is $2,910$. Considering the overall macro-averaged F-measure of the classifier, the proposed method outperforms the global feature ranking method. Although the macro-average of χ^2 in both methods is better than that of OR ranking measure, we prefer to use OR ranking because it offers superior results for learning the minority class.

From Chapter 3, the behavior of class imbalance can be analyzed by sparsity measure. Let SP_N and SN_N be the sparsity of training documents belonging to the minority (positive) and majority (negative) class, respectively. Figure (7.2) depicts the normalized sparsity of the minority class ($SP_N/(SP_N + SN_N)$. We learn from Chapter 3 that the lower sparsity level implies a more compact learning model, offering better performance and accuracy. For both feature ranking measures (OR and χ^2), the reverse discrimination approach guarantees lower class sparsity than that of the global ranking approach.

7.8 Conclusion

A text classifier framework to predict social relations using web documents was proposed. It has been shown that the sparse characteristic of social networks causes high class imbalance, which degrades the performance of supervised learning. To deal with the class imbalance, a feature-level re-balance scheme, feature ranking based on reverse discrimination, was proposed. In the proposed approach, the feature ranking method is compelled to select features fairly from all classes, with especial attention to the minority class (here, the positive class).

The Rocchio text classifier enjoys better performance using the proposed method. However, our experiments show that feature ranking based on reverse discrimination cannot improve SVM classifier performance. It can be explained as follows: SVM classifier is sensitive to training data rather than feature space. In SVM classifier, only a small portion of training examples, also known as support vectors, are really used to train the classification boundary. In the lack of the support vectors, balancing feature space cannot compensate for this shortage. On the other hand, multiple re-sampling techniques, due to the generating of new data samples, may provide support vectors [66,68]. We also expect that combining a feature level re-balance approach with data-level multiple samplings may offer promising results.

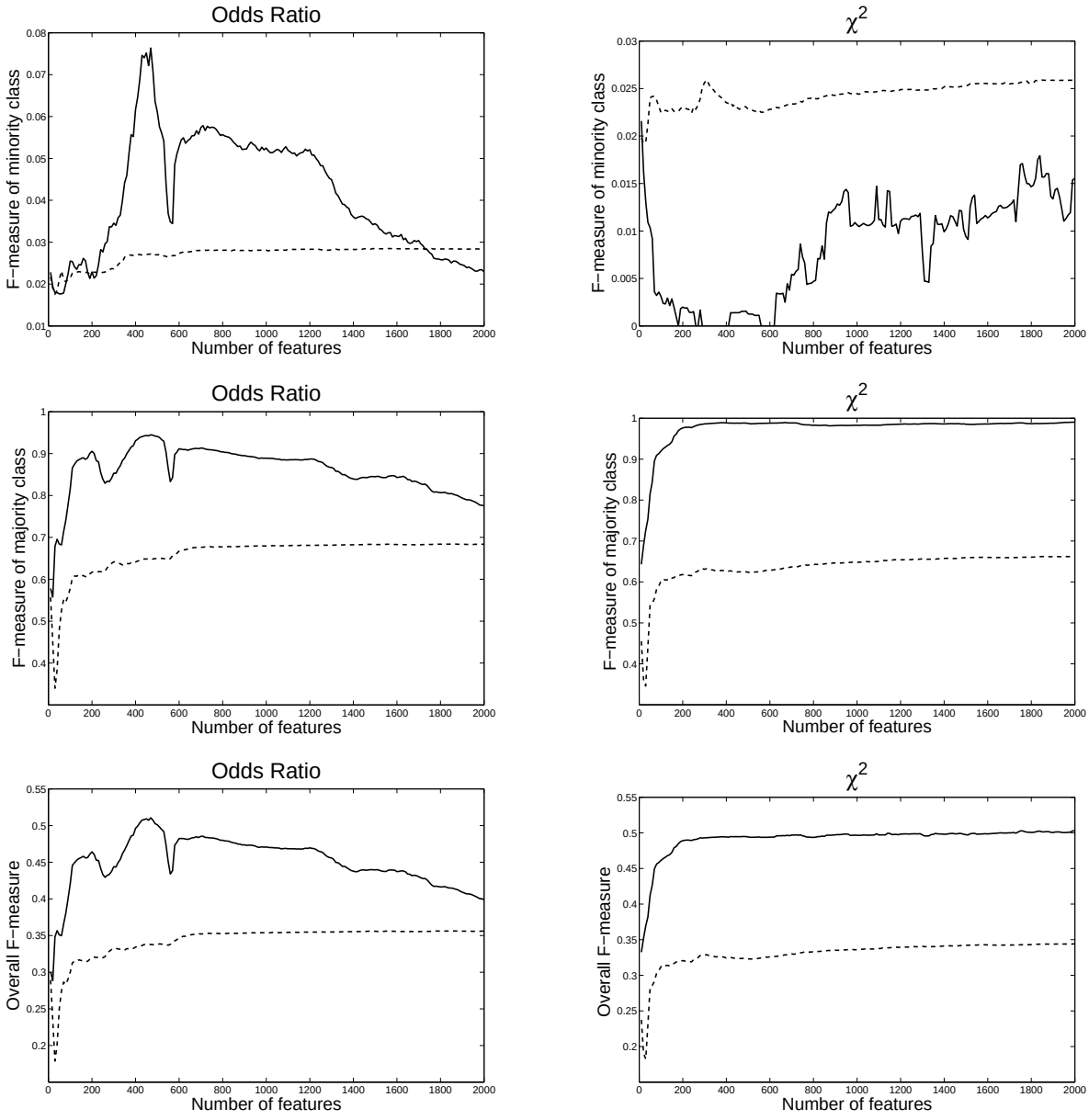
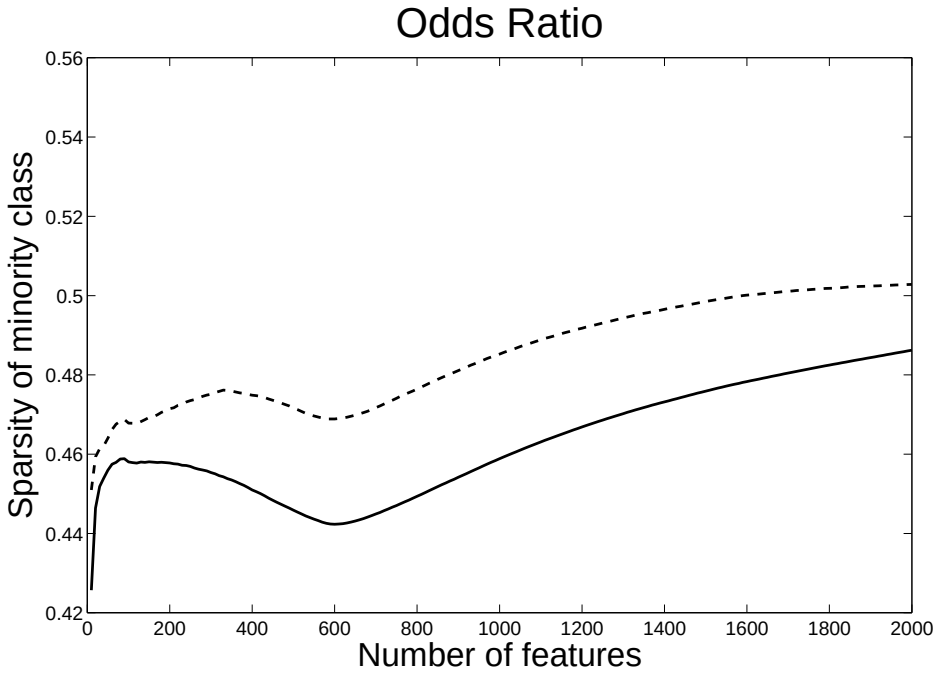
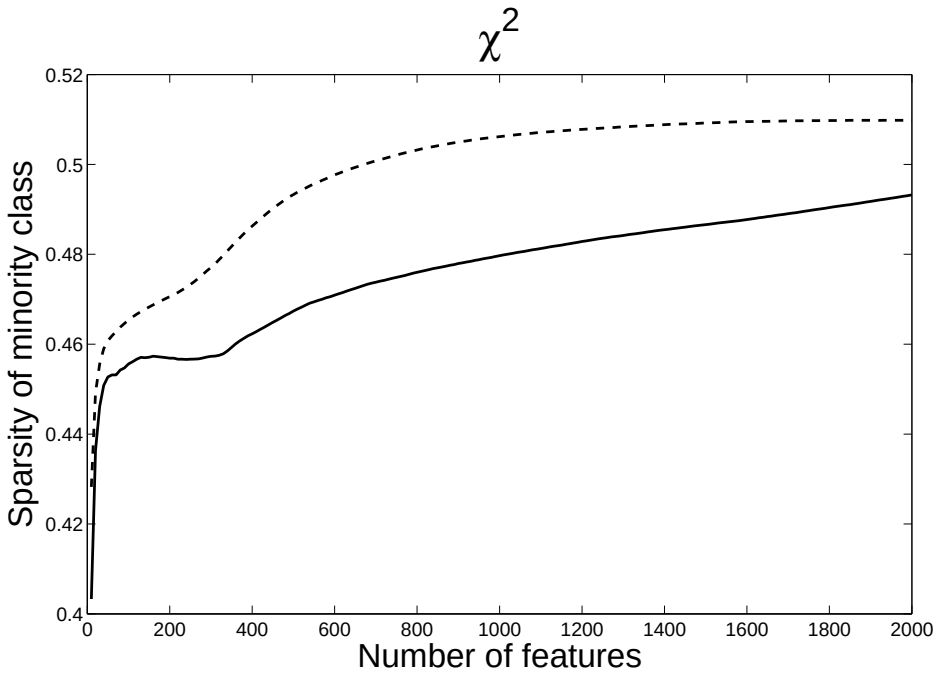


Figure 7.1: Macro-averaged F-measure and F-measure of the majority and minority class using reverse discrimination or local ranking (solid lines), and global (dashed) feature ranking method for OR and χ^2.



(a)



(b)

Figure 7.2: Sparsity of the minority class vs. filter level using global feature ranking (dashed lines), and reverse discrimination or local feature ranking (Solid lines).

Chapter 8

Conclusion and Future Work

The overall aim of this thesis was mainly to characterize the behavior of feature ranking methods in text classifiers, identify their drawbacks, and propose effective models, techniques, and algorithms to handle their deficiencies.

The first part of the thesis was devoted to highlighting the advantages and disadvantages of feature ranking methods. Focusing on the disadvantages, we studied the behavior of feature ranking methods by a set of experiments in different scenarios. According to the findings, the feature ranking-based feature selection techniques fail in two major cases: *(i)* filtering out the term redundancies; and *(ii)* the problem of unbalanced data classification. Additionally, the ranking methods are very domain specific and suffer from a high problem dependency.

In the second part, which includes the major contributions of the thesis, a set of models, algorithms, and measures were proposed to enhance the performance of feature ranking methods for text classifiers.

8.1 Contributions

- **Experiments to show the drawbacks of feature ranking methods.**
  - According to the findings in Section 3.2.2, redundant terms not only have no discriminating benefits for the classifier, but also reduce the chance that other less informative but non-redundant terms can contribute to the classification process.
  - A new ranking method called Single Term Prediction (STP), which ranks the terms based on their individual discriminant capacity, was proposed. In Sec-

tion 3.2.1 we showed that STP ranking always performs very poorly as compared to the other methods, including random ranking. It suggests ignoring the correlation and dependency between terms is as destructive as noise in feature ranking.

- In Section 3.3, failing in multi-class problems, especially with high class distribution imbalance was discussed. We showed that any class distribution imbalance affects classifier performance through increasing or decreasing the degree of local (class) sparsities. Because of high dimensional feature space in text classification problems, more important than having a balanced training samples is the need to have all classes in the training data balanced and compact.

- **Two new feature ranking measures.** In Section 2.5, two new ranking measures NIG and CDF were proposed. We explained that by using NIG, we can bound IG measure with the weight of the term (*e.g.* its document frequency). In Section 4.6, the effectiveness of NIG was experimentally demonstrated. It outperformed the well-known IG and Max(χ^2) ranking measures in four out of six data sets. The other proposed measure, CDF, is technically a supervised version of IDF. In one case (out of six data sets), it outperformed other ranking measures.

- **DFLP Method for Evaluating Feature Ranking Measures.** To deal with high problem dependency of feature ranking methods, in Chapter 4 the notion of backward and forward FLP as hybrid models was proposed to evaluate the performance of feature ranking measures. Based on this objective framework, feature meta-ranking strategy was introduced to rank the candidate feature ranking measures based on their DFLP measures and select the one best-suited to the problem. We also proved that, in the ideal case, the classifier performance increases as we increase the number of features, regardless of how we ranked them.

- **Extracting Domain-Specific Stopwords.** In Chapter 5, the concept of domain-specific stopwords was introduced. Using an approximate estimation of FLP measure using the global sparsity of training data, domain-specific stoplists for six data sets were extracted. We proposed a machine learning evaluation measure and the amount of misclassification rate, instead of information retrieval evaluation framework to assess the extracted stoplists.

- **A New Asymmetric Dependency Measure: Information Theoretic Inclusion Index.** In Chapter 6, a new measure of asymmetric dependency based on mutual

information, which is called inclusion index, was introduced. This measure is a weighted version of mutual information between two features t_i and t_j. The mutual information for feature t_i is weighed against the information provided by the other feature t_j.

- **Redundancy Reduction Using Term Dependency Tree.** An algorithm to extract redundant terms was proposed in Section 6.6. The algorithm was based on a newly proposed term dependency tree, which is constructed by estimating the pair-wise inclusion measures. All nodes in the tree are clustered according to their outdegree values. The nodes with outdegree less than two are grouped as Link nodes. We showed that these nodes are potentially redundant. By proposing the notion of substitution cost in section 6.5, an objective function was introduced to decide whether a Link node should be removed or not. Experimental results showed that the proposed approach outperformed the aggressive feature selection by the stand-alone IG for most data sets with a weak classifier such as Rocchio.

- **Reverse Discrimination in Feature Ranking.** In Chapter 7 a local feature ranking for unbalanced data was proposed, based on the idea of reverse discrimination to re-balance the local (class) sparsity of training data. In this scheme, feature ranking is forced to select fairly the representative features from all classes, in particular the small ones. According to the experimental results in Section 7.7, the Rocchio text classifier enjoys better performance using the proposed method.

- **A Supervised Learning Framework for Learning Social Networks from Text data.** The problem of extracting social networks, which is a classification problem of highly unbalanced data, was investigated as a case study to experiment with the local feature rankling scheme. In Section 7.5, first we suggested a text classification approach to learning social networks, and next, a classification framework is proposed for learning unexplored social links from an incomplete, small set of known relations. We also proposed the document aggregation model instead of the document similarity for representing the links between actors in a community.

8.2 Future Work

This thesis is ended here, but we believe that the research in this area should be continued. Working on feature ranking schemes for text classifiers directly and indirectly opened

some new opportunities in text mining research.

- **DNA Microarray and Gene Expression Data.** In this research, we mainly focused on the text mining application. We are also interested in applying the DFLP measure and meta-ranking framework to other high dimensional data such as DNA microarray and gene expression data. The idea of stopword extraction in text mining can be adopted by bioinformatics applications to extract noise and irrelevant features.

- **Feature Ranking for Unsupervised Feature Selection.** The supervised dimensionality reduction techniques, compared to the unsupervised examples are more successful and well-appreciated by researchers in text categorization. In the lack of efficient feature ranking methods for text clustering, the IDF ranking method is widely used. We showed in this thesis that relying on IDF measure for feature ranking can be very risky, especially when dealing with unbalanced categories. One interesting topic can be designing efficient term ranking measures for text clustering applications.

- **Ontology Learning and Taxonomy Extraction Using Term Dependency Tree.** The idea of term dependency tree may provide a framework to extract taxonomies for ontology learning. Let $\mathbf{T} = \{t_1, t_2, ..., t_q\}$ be the set of terms, realizing domain-specific concepts, also known as terminology. Terms can be extracted from a corpus in domain $\mathbb{D}$. The problem of taxonomy extraction is to learn links between the terms in $\mathbf{T}$. Using the idea of term dependency tree and an information theoretic inclusion index, we are able to extract approximately the links.

  One practical application of taxonomy extraction is in metadata enrichment. Recently, Folksonomy, user generated taxonomy, has become a popular approach to generating metadata for large databases. One well-known example is Flickr[1] photo-sharing web site. People can post and share their photos on the web site and generate their own metadata (tag) for the photos. One problem with this type of tagging is that it generates flat (not hierarchial) metadata. To provide semantic search and retrieval, we need to give depth to the tag list by building a taxonomy (hierarchial metadata) from the given tags.

[1] http://www.flickr.com/

8.3 Publication Resulting from this Work

- Book Chapter
  - M. Makrehchi, M. S. Kamel. **Aggressive Feature Selection by Feature Ranking**. In Computational Methods of Feature Selection, Huan Liu and Hiroshi Motoda, Eds.; Chapman and Hall/CRC Press, 2007 *(in press)*.
- Journal
  - M. Makrehchi, M. S. Kamel. **Differential Filter Level Performance for Evaluating Feature Ranking Measures in Text Classifiers**, *Submitted to Knowledge and Information Systems Journal.*
  - M. Makrehchi, M. S. Kamel. **Extracting Domain-Specific Stopwords Using Sparsity Based Estimation of Classifier Performance**, *Submitted to The IEEE Transaction on Knowledge and Data Engineering Journal.*
  - M. Makrehchi, M. S. Kamel. **Impact of Term Dependency and Class Imbalance on Performance of Feature Ranking Methods for Text Classifiers**, *Submitted to the Journal of Pattern Analysis and Applications.*
  - M. Makrehchi, M. S. Kamel. **Learning Social Network from Content**, *To be submitted to Knowledge and Information Systems Journal.*
- Conference
  - M. Makrehchi, M. S. Kamel. **Automatic Taxonomy Extraction Using Google and Term Dependency**, *Accepted in the 2007 IEEE/WIC/ACM International Conference on Web Intelligence (WI'07), to be held in Silicon Valley, CA, USA, 2 - 5 Nov. 2007.*
  - M. Makrehchi, M. S. Kamel. **A Text Classification Framework with a Local Feature Ranking for Learning Social Networks**, *Accepted in the 2007 IEEE International Conference on Data Mining, (ICDM 2007), to be held in Omaha, NE, USA, 28 - 31 Oct. 2007.*
  - M. Makrehchi, M. S. Kamel. **Learning Term Dependency Links Using Information Theoretic Inclusion Measure**, *Accepted in the 2007 ICDM Workshop on Mining Graphs and Complex Structures (MGCS2007), to be held in Omaha, NE, USA, 28 - 31 Oct. 2007.*

- M. Makrehchi, M. S. Kamel. **Combining Feature Ranking for Text Classification**, *Accepted in 2007 IEEE International Conference on Systems, Man and Cybernetics, to be held in Montreal, Canada, 7 - 10 Oct. 2007.*
- M. Makrehchi, M. S. Kamel. **Learning Social Networks Using Multiple Resampling Method**, *Accepted in 2007 IEEE International Conference on Systems, Man and Cybernetics, to be held in Montreal, Canada, 7 - 10 Oct. 2007.*
- M. Makrehchi, M. S. Kamel. **Learning Social Networks from Web Documents Using Support Vector Classifiers**, *The 2006 IEEE/WIC/ACM International Conference on Web Intelligence, Hong Kong, Dec. 18 - 22, 2006.*
- M. Makrehchi, M. S. Kamel. **Building Social Networks from Web Documents: A Text Mining Approach**, *The 2nd LORNET Scientific Conference, Vancouver, BC, Canada, Nov. 14-18, 2005.*
- M. Makrehchi, M. S. Kamel. **Text Classification Using Small Number of Features**, *Machine Learning and Data Mining Conference (MLDM2005), Leipzig, Germany, pp. 580–589, July 9–11, 2005.*
- M. Makrehchi, M. S. Kamel. **A Fuzzy Set Approach to Extracting Keywords from Abstracts**, *NAFIPS-IEEE Annual Meeting of the Fuzzy Information Processing, Banff, Canada, pp. 528–532, June 27–30, 2004.*

Appendix A

Data Sets

A.1 Document Data Sets

Six document data sets, including four well-known benchmark collections and two data sets created by the authors, have been used in this paper. All data sets are preprocessed by Porter stemmer, and stopword reduction using a general stoplist. In a few cases, empty or very small documents and classes containing less than three documents have been removed from the data base. Table A.1 depicts some statistics of the data sets used in this paper.

A.1.1 Industry Sectors

This data set contains company web pages, which are hierarchically classified into 71 categories [72]. To reduce the number of classes, the documents in the classes of the same hierarchy are merged into seven larger categories. The resulting set of categories consists

Table A.1: The data sets statistics.

Parameter	Industry Sectors	LO Metadata	20 Newsgroups	Reuters	WebKB	CS Abstracts
Number of documents	4018	1525	19997	7181	48852	2912
Number of terms	30943	9206	28983	17138	75636	19722
term per document	329.18 ± 648.55	164.14 ± 48.98	105.63 ± 79.90	46.51 ± 40.89	86.53 ± 416.00	217.95 ± 61.58
Number of classes	7	31	20	20	7	17
document per class	574.00 ± 295.94	49.19 ± 13.47	999.85 ± 0.67	359.50 ± 447.48	6978.86 ± 5183.52	171.29 ± 30.31
term per class	107.95 ± 11.98	84.85 ± 13.85	105.63 ± 19.31	57.95 ± 13.90	26.24 ± 7.83	219.06 ± 11.16
sparsity (%)	99.6571	99.0860	99.6355	99.7286	99.9629	98.8949
low-DF threshold	1	1	3	not available	1	3
Number of terms removed by stopword	419	382	1350	not available	432	997
Number of terms removed by stemming	9491	4049	16319	not available	15330	9173

of “materials”, “energy”, “financial”, “health-care”, “technology”, “transportation”, and “utilities”.

A.1.2 20 Newsgroups

This data set was first employed in [42]. The collection includes about 20,000 documents, which are uniformly distributed into 20 classes. This data set is a good example of a homogeneous and uniform data set (with minimum class distribution imbalance), which makes it for the classification easy even for weak classifiers.

A.1.3 Reuters

The Reuters document collection is a well-known benchmark data set for text categorization algorithms [57]. The original version, in terms of class imbalance, the distribution of terms in documents and classes, and the number of classes, is a difficult data set for classification. In this paper, a small version of the Reuters is employed, including only 20 classes.

A.1.4 WebKB

This data set [13] is one of the worst collections for categorization for two reasons: *(i)* very scattered vocabulary (the distribution of features across classes); and *(ii)* high imbalance. The data set includes 48,852 documents distributed across seven categories.

A.1.5 The Learning Object Metadata (LO Metadata)

This data set has been collected from the Schoolnet Canada[1] for the Learning Objects Repositories NETwork (LORNET) project, supported by the National Science and Engineering Research Council of Canada. The collection contains 1,525 learning object metadata that are classified into 31 categories.

A.1.6 Computer Science Abstracts

This data set have been also collected by the authors from CiteSeer Computer Science Directory[2] for keyword extraction research. The collection contains 2,912 documents in

[1] *http://www.schoolnet.ca/home/e/resources/*
[2] *http://citeseer.ist.psu.edu/directory.html*

17 categories.

A.2 Social Network Data

In this research, a real FOAF database contains 210,611 RDF triples is used (see Appendix A) [18,108]. We searched the database using two queries to explore two types of information: *(i)* Relations between the individuals; and *(ii)* Any web resource addresses and URLs related to the individuals. The first information, which represents a set of true social networks and can be called descriptive social network, is used for training and evaluation purpose. The second queried information is used to construct vector space model of actors by downloading their web resources, which includes, home pages, blogs, resumes, articles, papers, and so on. We use downloaded data to build actor-term matrix and model the actor-actor relationship based on merging their document vectors. Using a small portion of relations in the true social network as training data, the missing relations are approximately predicted. This resulting network can be considered as predictive social network.

The set of actors includes 34,275 individuals, which are connected together through 33,419 ties out of 587,370,675 possible relationship. The database represents an extremely sparse (%99.99) social network. If we use this data set as the training data, and assume a relation between two actors as positive class and a no-relation as negative class, it addresses a very unbalanced ($1 : 17,575$) classification problem. One alternative solution is breaking the database into small sub-graphs. For example, Figure (A.1-a) represents a subset of the social networks, in which both actors of all relations have downloadable web resources. It includes 2,933 actors, which build some social networks with 2,641 ties out of 42,99,778 relations. The sparsity index is negligibly decreased to %99.94, while the class imbalance is dramatically reduced to ($1 : 1,627$).

The resulting database is still very big. It is downsized once more into a small subset of the social database by removing very large and very small (with less than 20 and more than 70 members) social networks. Figure (A.1-b) illustrates the final networks for training and evaluation purpose. The number of actors is 254, which are interconnected by 246 ties out of 32,131. The sparsity index is %99.21 with class imbalance of $1 : 130$, which is still challenging for classification algorithms.

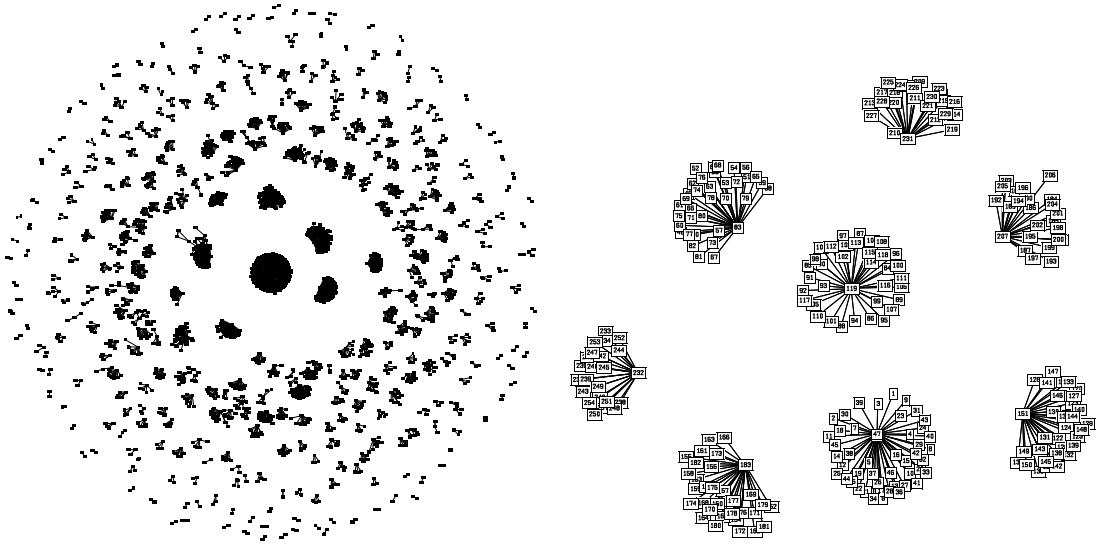
In the FOAF metadata collection, the relationships are described by various vocabularies. Table A.2 shows the vocabulary used for querying the relations. Table A.2 depicts the keywords to search URLs associated to each individual.

Table A.2: The vocabulary for querying the relations among the individuals.

http://purl.org/vocab/relationship/knowByRep
http://purl.org/vocab/relationship/knowByReputation
http://purl.org/vocab/relationship/knowInPass
http://purl.org/vocab/relationship/knowInPassing
http://purl.org/vocab/relationship/wouldLikeToKnow
http://purl.org/vocab/relationshipknowsByReputation
http://purl.org/vocab/relationshipwouldLikeToKnow
http://www.perceive.net/schemas/relationship/knowsOf
http://www.w3.org/1999/02/22-rdf-syntax-ns#knows
http://xmlns.com/foaf/0.1/knowns http://xmlns.com/foaf/0.1/knows

Table A.3: The vocabulary for querying the web URLs of the individuals.

http://owl.mindswap.org/2003/ont/owlweb.rdf#mindswapHomepage
http://www.w3.org/2000/01/rdf-schema#homepage
http://www.w3.org/2000/10/swap/pim/contact#homePage
http://xmlns.com/foaf/0.1/groupHomepage
http://xmlns.com/foaf/0.1/homepage
http://xmlns.com/foaf/0.1/projectHomepage
http://xmlns.com/foaf/0.1/publication
http://xmlns.com/foaf/0.1/publications
http://xmlns.com/foaf/0.1/schoolHomepage
http://xmlns.com/foaf/0.1/weblog
http://xmlns.com/foaf/0.1/workInfoHomepage
http://xmlns.com/foaf/0.1/workOrgHomepage
http://xmlns.com/foaf/0.1/workplaceHomepage



(a) (b)

Figure A.1: FOAF data set, (a) all social networks, (b) a subset of FOAF data set.

Degree measure, which is estimated by the the number of ties associated to an actor, is an important measure for social network analysis [118]. According to [108], degree measure in social network data bases follows the Zifp's law, in which a few actors have many ties while many others have very few. (Figure (A.2)). Due to this fact, the networks with very small and vary large number of actors are removed from the database.

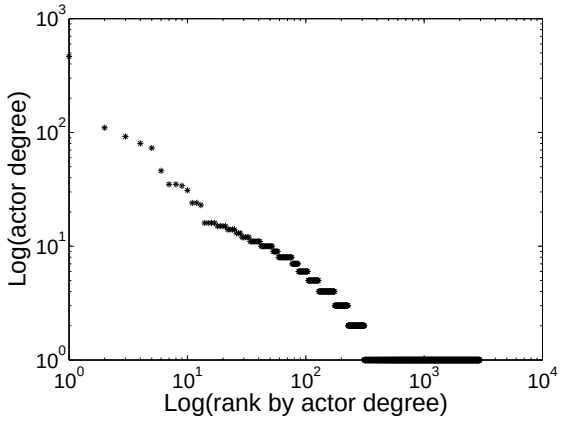


Figure A.2: The Zipf's law for the FOAF data set.

Appendix B

List of Symbols

$\mathbf{A}$	set of actors in a community
A_τ	area under FLP
a_i	actor i in a community
$\mathbf{C}$	set of category labels
C	number of categories
$CDF(t_j, c_k)$	category-document frequency
c_k	category k
$\mathcal{D}$	document space
$\mathbb{D}$	Training data set (represented by document-term matrix)
$\mathbf{D_i}$	i^{th} document of the training data set
$DFV(t_j)$	document frequency variance
$\mathbb{F}$	category-term ranking matrix
F_n	False-Negative
F_p	False-Positive
$F_{measure}$	F-measure
$\mathcal{G}$	generalization ability or learn-ability of a classifier

$\mathbb{H}$	classifier space
$H(t_j)$	term entropy
$H(t_1, t_2, ..., t_m)$	joint entropy
$H(c_k\|t_j)$	conditional entropy
$H_n(t_i\|t_j)$	normalized conditional entropy
$H_s(\mathbf{V})$	head of sequence $\mathbf{V}$
$\mathbf{h}$	a subset of classifiers $\mathbf{h} \subset \mathbb{H}$
h	classifier
$I(x;y)$	mutual information
$IDF(t_j)$	inverse document frequency
$IG(t_j, c_k)$	local information gain
$IG(t_j)$	global information gain
$J(\mathbf{T})$	classifier performance using feature vector $\mathbf{T}$
K	resolution
K_D	constants depending on the type of classifier and the data set in estimating classifier performance by global sparsity
K_{JH}	scaling factor
$LDF(t_j)$	local document frequency
M	number of all possible ties
$MI(t_1, t_2, ..., t_m)$	the summation all mutual information values
m	number of terms in the vocabulary
$NIG(t_j)$	normalized information gain
$NMI(t_i; t_j)$	normalized mutual information
n	number of documents in the training data set
$n(t_i, t_j)$	the number of documents in the training data containing t_i and t_j
$n(t_i)$	number of documents in the training data containing t_i
n_l	the size of larger class in class imbalance scenario
n_s	the size of smaller class in class imbalance scenario
n_k	the number of samples in class k
n_r	number of ties (relations)
OC	outlier count
$odds(t_j\|c_k)$	odds value
$OR(t_j, c_k)$	odds ratio

$\mathcal{P}(\mathbf{V})$	power sequence of $\mathbf{V}$
$P(x)$	probability
$P(t_i, c_k)$	joint probability distribution of a term and a category
$P(c_k \mid t_j)$	the conditional probability of c_k given term t_j
$P_c(t_j, c_k)$	precision of classification rule $t_j \rightarrow c_k$
P_c	precision
q	number of selected features ($q < m$)
$\mathbf{R}_T$	the set of training ties
$\mathcal{R}$	the set of relation labels $\{0, 1\}$
$R_c(t_j, c_k)$	recall of classification rule $t_j \rightarrow c_k$
R_c	recall
RND	random feature ranking
$\mathbf{S}$	subsequence of a feature ranking sequence
S	global sparsity of training data
S_N	social network sparsity
SP_N	sparsity of training documents belong to the minority or positive class
SN_N	sparsity of training documents belong to the majority or negative class
STP	single term prediction ranking
$S(c_k)$	local sparsity of class k
$s\chi^2(t_j, c_k)$	simplified χ^2
$\mathcal{T}(t_j, c_k)$	threshold function
$\mathbf{T}$	vocabulary (list of terms)
$TF \times IDF$	term frequency, inverse document frequency weighting
T_n	True-Negative
T_p	True-Positive
$T_s(\mathbf{V})$	tail of sequence $\mathbf{V}$
t_j	term j of the vocabulary
$\mathbb{V}$	the set of local feature vectors
$\mathbf{V}$	feature ranking sequence
$\mathbb{W}$	relation-term matrix
$\mathbf{W}_{i,j}$	aggregated vector of actors i and j
$w_{i,j}$	weight of term j in document i
x_i	number of distinct words in the i^{th} document or the number of non-zero entries in the i^{th} row of the document-term matrix

α	filter level (feature ranking threshold)
β	F-measure parameter
χ^2	χ^2 distribution
$\Delta_\phi(t_j)$	substitution cost of t_j
δ_{FLP}	differential filter level performance measure
$\epsilon_{\mathbf{h}}$	classifier errors corresponding to the set of classifiers $\mathbf{h}$
Γ	aggregation function
γ	constant depending on the type of classifier and the data set in estimating classifier performance by global sparisty
$\mathbf{\Lambda}$	backward feature ranking sequence
μ_{IG}	mean of information gain scores
$\mathbf{\Phi}$	term ranking vector
$\mathbf{\Phi}^-$	inverted feature score vector
$\phi(t_j, c_k)$	category-term weight representing the relevance of the term j to the class k
ϕ	feature ranking measure
$\phi(t_j)$	feature ranking score of term t_j
σ_{IG}	standard deviation of information gain scores
τ	maximum filter level or threshold
τ_c	category-term frequency threshold
τ_d	correlation threshold
τ_s	substitution cost threshold
$\theta(t_i, t_j)$	inclusion index
ζ	sort function

Bibliography

[1] Jan Bakus and Mohamed S. Kamel. Higher order feature selection for text classification. *Knowl. Inf. Syst.*, 9(4):468–491, 2006.

[2] Roberto Battiti. Using the mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5(4):537–550, 1994.

[3] Michael W. Berry. *Survey of Text Mining: Clustering, Classification, and Retrieval*. Springer, 2004.

[4] Michael W. Berry and Murray Browne. *Understanding Search Engines: Mathematical Modeling and Text Retrieval*. SIAM, Society for Industrial and Applied Mathematics, second edition, 2005.

[5] Avrim Blum, Adam Kalai, and John Langford. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *Computational Learing Theory*, pages 203–208, 1999.

[6] J. Brank, M. Groblenik, N. Milic-Frayling, and D. Mladenic. Interaction of feature selection methods and linear classification models. In *Workshop on Text Learning held at ICML-2002, Sydney, Australia*, 2002.

[7] Aitao Chen and Fredric C. Gey. Building an Arabic stemmer for information retrieval. In *TREC*, 2002.

[8] H.-C. Chiang, R. L. Moses, and L. C. Potter. Classification performance prediction using parametric scattering feature models. In E. G. Zelnio, editor, *Proc. SPIE Vol. 4053, p. 546-557, Algorithms for Synthetic Aperture Radar Imagery VII, Edmund G. Zelnio; Ed.*, volume 4053 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 546–557, August 2000.

[9] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Comput. Linguist.*, 16(1):22–29, 1990.

[10] C. Coombs, R. M. Dawes, and A. Tversky. *Mathematical Psychology: An Elementary Introduction*. Prentice-Hall, 1970.

[11] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley-Interscience, 1991.

[12] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. John Wiley, New York, 1990.

[13] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, pages 509–516, 1998.

[14] Dan Crow and John DeSanto. A hybrid approach to concept extraction and recognition-based matching in the domain of human resources. In *ICTAI*, pages 535–539, 2004.

[15] Ido Dagan, Lillian Lee, and Fernando C. N. Pereira. Similarity-based models of word cooccurrence probabilities. *Machine Learning*, 34(1-3):43–69, 1999.

[16] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[17] J. Diesner and K. M. Carley. Exploration of communication networks from the enron email corpus. In *Proc. of Workshop on Link Analysis, Counterterrorism and Security at SIAM International Conference on Data Mining 2005. Newport Beach, CA, April 21-23, 2005*, pages 3–14, 2005.

[18] Li Ding, Lina Zhou, Tim Finin, and Anupam Joshi. How the semantic web is being used: An analysis of foaf documents. *Proceedings of the 38th International Conference on System Sciences, January 03, 2005*, pages 1–10, 2005.

[19] Edward R. Dougherty. Feature-selection overfitting with small-sample classifier design. *IEEE Intelligent Systems*, 20(6):64–66, 2005.

[20] D.Watts and S. Strogatz. Collective dynamics of smallworld networks. *Nature*, (363):202–204, 1998.

[21] Deniz Erdogmus and Jose C. Principe. Lower and upper bounds for misclassification probability based on renyi's information. *J. VLSI Signal Process. Syst.*, 37(2-3):305–317, 2004.

[22] George Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305, 2003.

[23] George Forman. A pitfall and solution in multi-class feature selection for text classification. In *in Proceedings of ICML-04, Twenty-first international conference on Machine learning*, pages 297–304, 2004.

[24] Christopher J. Fox. Lexical analysis and stoplists. In *Information Retrieval: Data Structures & Algorithms*, pages 102–130. 1992.

[25] Kenji Fukumizu, Francis R. Bach, and Michael I. Jordan. Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces. *Journal of Machine Learning Research*, 5:73–99, 2004.

[26] Evgeniy Gabrilovich and Shaul Markovitch. Text categorization with many redundant features: Using aggressive feature selection to make SVMs competitive with C4.5. In *Proceedings of ICML-04, Twenty-first international conference on Machine learning*, pages 321–328, Banff, Alberta, Canada, 2004. Morgan Kaufmann.

[27] Gerald Gavin and Olivier Teytaud. Lower bounds for training and leave-one-out estimates of the generalization error. In *ICANN '02: Proceedings of the International Conference on Artificial Neural Networks*, pages 583–588, London, UK, 2002. Springer-Verlag.

[28] Xin Geng, De-Chuan Zhan, and Zhi-Hua Zhou. Supervised nonlinear dimensionality reduction for visualization and classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 35(6):1098–1107, 2005.

[29] Malcolm Gladwell. *The Tipping Point: How Little Things Can Make a Big Difference*. Little, Brown, March.

[30] Jovan DJ. Golic. On the relationship between the information measures and the bayes probability of error. *IEEE Trans. Inf. Theor.*, 33(5):681–693, 1987.

[31] T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M. Caligiuri, C. Bloomfield, and E. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, October 1999.

[32] Jihun Ham, Daniel D. Lee, Sebastian Mika, and Bernhard Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *ICML*, 2004.

[33] Eui-Hong Han and George Karypis. Fast supervised dimensionality reduction algorithm with applications to document categorization & retrieval. In *CIKM*, pages 12–19. ACM, 2000.

[34] Ahmed E. Hassan and Richard C. Holt. The small world of software reverse engineering. In *WCRE '04: Proceedings of the 11th Working Conference on Reverse Engineering (WCRE'04)*, pages 278–283, Washington, DC, USA, 2004. IEEE Computer Society.

[35] Jane Huffman Hayes, Alex Dekhtyar, and Senthil Sundaram. Text mining for software engineering: how analyst feedback impacts final results. In *MSR '05: Proceedings of the 2005 international workshop on Mining software repositories*, pages 1–5, New York, NY, USA, 2005. ACM Press.

[36] Sherwood Haynes. Stemming and stopwording effects on word frequency. In *Proceedings of the Thirteenth Midwest Artificial Intelligence and Cognitive Science Conference: MAICS 2002, S. Conlon, ed., Chicago, IL*, pages 71–75, 2002.

[37] Vera Hollink, Jaap Kamps, Christof Monz, and Maarten de Rijke. Monolingual document retrieval for european languages. *Inf. Retr.*, 7(1-2):33–52, 2004.

[38] http://www.ranks.nl/stopwords/index.html.

[39] Shen Huang, Zheng Chen, Yong Yu, and Wei-Ying Ma. Multitype features coselection for web document clustering. *IEEE Transactions on Knowledge and Data Engineering*, 18(4):448–459, 2006.

[40] Anil K. Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[41] D. Jensen and J. Neville. Data mining in social networks. *National Academy of Sciences Symposium on Dynamic Social Network Analysis, November 7-9, 2002, Washington, DC: National Academy Press.*, 2002.

[42] Thorsten Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 143–151, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[43] 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.

[44] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *International Conference on Machine Learning*, pages 121–129, 1994. Journal version in AIJ, available at http://citeseer.nj.nec.com/13663.html.

[45] M. Kawahara and H. Kawano. Mining association algorithm with threshold based on roc analysis. In *Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 3*, pages 3010–3017. IEEE Computer Society, 2001.

[46] F. Khan, T. Fisher, L. Shuler, T. Wu, and W. Pottenger. Mining chat-room conversations for social and semantic interactions, 2002.

[47] R. Kohavi and G. John. *The Wrapper Approach*, pages 33–50. Feature Selection for Knowledge Discovery and Data Mining. Kluwer Academic Publishers, 1998.

[48] Daphne Koller and Mehran Sahami. Toward optimal feature selection. In *International Conference on Machine Learning*, pages 284–292, 1996.

[49] Sang Ok Koo, Soo Yeon Lim, and Sang-Jo Lee. Building an ontology based on hub words for information retrieval. In *Web Intelligence*, pages 466–469, 2003.

[50] L. I. Kuncheva. A theoretical study on six classifier fusion strategies. *IEEE Transactions on Pattern Analalysis and Machine Intelligence*, 24(2):281–286, February 2002.

[51] Tien-Fang Kuo and Y. Yajima. Ranking and selecting terms for text categorization via SVM discriminate boundary. *Granular Computing, 2005 IEEE International Conference on*, 2:496– 501 Vol. 2, 2005.

[52] Nojun Kwak and Chong-Ho Choi. Improved mutual information feature selector for neural networks in supervised learning. In *IEEE International Conference on Neural Networks (IJCNN'99)*, volume II, pages 1313–1318, Washington DC, July 1999. IEEE.

[53] Nojun Kwak and Chong-Ho Choi. Input feature selection by mutual information based on parzen window. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1667–1671, 2002.

[54] Savio L. Lam and Dik L. Lee. Feature reduction for neural network based text categorization. In *Proceedings of DASFAA-99, 6th IEEE International Conference on Database Advanced Systems for Advanced Application*, pages 195–202, 1999.

[55] Wai Lam, Miguel E. Ruiz, and Padmini Srinivasan. Automatic text categorization and its applications to text retrieval. *IEEE Transactions on Knowledge and Data Engineering*, 11(6):865–879, 1999.

[56] Jonathan S. Leonard. Anti-discrimination or reverse discrimination: The impact of changing demographics, title vii and affirmative action on productivity. Nber working papers, National Bureau of Economic Research, Inc, November 1983.

[57] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[58] Fan Li and Yiming Yang. Using recursive classification to discover predictive features. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1054–1058, New York, NY, USA, 2005. ACM Press.

[59] C.X. Ling and C. Li. Data mining for direct marketing: Problems and solutions. In *Proceedings of The Forth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, AAAI*, pages 73–79, 1998.

[60] Huan Liu and Hiroshi Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[61] Tao Liu, Shengping Liu, Zheng Chen, and Wei-Ying Ma. An evaluation on feature selection for text clustering. In *Proceedings of ICML 2003*, pages 488–495, 2003.

[62] Rachel TszWai Lo, Ben He, and Iadh Ounis. Automatically building a stopword list for an information retrieval system. *the Journal on Digital Information Management: special issue on the 5th Dutch-Belgian Information Retrieval Workshop (DIR'05)*, 3(1):3–8, 2005.

[63] David Mackay. *Information Theory, Inference and Learning Algorithms*. Cambridge University, 2003.

[64] Masoud Makrehchi and Mohamed S. Kamel. Building social networks from web documents: A text mining approach. In *The 2nd LORNET Scentific Conference, Vancouver, BC, Canada, Nov. 14-18, 2005*, 2005.

[65] Masoud Makrehchi and Mohamed S. Kamel. Text classification using small number of features. In *Proceedings 4th International Conference Machine Learning and Data Mining in Pattern Recognition, MLDM 2005, Leipzig, Germany, July 9-11, 2005*, pages 580–589, 2005.

[66] Masoud Makrehchi and Mohamed S. Kamel. Learning social networks from web documents using support vector classifiers. In *2006 IEEE / WIC / ACM International Conference on Web Intelligence (WI 2006), 18-22 December 2006, Hong Kong, China*, pages 88–94, 2006.

[67] Masoud Makrehchi and Mohamed S. Kamel. Aggressive feature selection by feature ranking. In Huan Liu and Hiroshi Motoda, editors, *Computational Methods of Feature Selection*. Chapman and Hall/CRC Press, 2007.

[68] Masoud Makrehchi and Mohamed S. Kamel. Learning social networks using multiple resampling method. 2007. Submitted to 2007 IEEE International Conference on Systems, Man and Cybernetics, to be held in Montreal, Canada, 7 - 10 Oct. 2007.

[69] J. I. Maletic and N. Valluri. Automatic software clustering via latent semantic analysis. In *Proceedings 14th IEEE International Conference on Automated Software Engineering (ASE'99), Cocoa Beach Florida*, pages 251–254, October 1999.

[70] N. Matsumura, D. Goldberg, and X. Llora. Mining directed social network from message board. In *In WWW '05: Special interest tracks and posters of the 14th inter-*

national conference on World Wide Web, New York, NY, USA, 2005. ACM Press., pages 1092–1093, 2005.

[71] Yutaka Matsuo, Hironori Tomobe, K. Hasida, and Mitsuru Ishizuka. Mining social network of conference participants from the web. In *WI '03: Proceedings of the IEEE/WIC International Conference on Web Intelligence*, pages 190–193, Washington, DC, USA, 2003. IEEE Computer Society.

[72] Andrew K. McCallum, Ronald Rosenfeld, Tom M. Mitchell, and Andrew Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In Jude W. Shavlik, editor, *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 359–367, Madison, US, 1998. Morgan Kaufmann Publishers, San Francisco, US.

[73] P. Mika. Bootstrapping the foaf-web: An experiment in social network mining, 2004.

[74] Peter Mika. Ontologies are us: A unifed model of social networks and semantics. In *Fourth International Semantic Web Conference (ISWC 2005)*, pages page 122–136, Galway, Ireland.

[75] Pabitra Mitra, C. A. Murthy, and Sankar K. Pal. Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analalysis and Machine Intelligence*, 24(3):301–312, 2002.

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

[77] Elena Montanes, Irene Diaz, Jose Ranilla, Elias F. Combarro, and Javier Fernandez. Scoring and selecting terms for text categorization. *IEEE Intelligent Systems*, 20(3):40–47, 2005.

[78] R. Lopez De Montaras. A distance-based attribute selection measure for decision tree induction. *Mach. Learn.*, 6(1):81–92, 1991.

[79] Junichiro Mori, Yutaka Matsuo, Mitsuru Ishizuka, and Boi Faltings. Keyword extraction from the web for foaf metadata. *1st Workshop on Friend of a Friend, Social Networking and the Semantic Web, 1-2 September 2004, Galway, Ireland*, 2001.

[80] J. Novovivcova, A. Malik, and P. Pudil. Feature selection using improved mutual information for text classification. In *Structural, Syntactic, and Statistical Pattern Recognition, Lecture Notes in Computer Science 3138*, pages 1010–1017, Lisbon, Portugal, 18-20 August 2004.

[81] Douglas W. Oard and Fredric C. Gey. The trec 2002 arabic/english CLIR track. In *TREC*, 2002.

[82] Sankar K. Pal and Pabitra Mitra. *Pattern Recognition Algorithms for Data Mining*. Chapman and Hall, 2004.

[83] Vivien Petras, Natalia Perelman, and Fredric C. Gey. UC berkeley at clef-2003 - Russian language experiments and domain-specific retrieval. In *CLEF*, pages 401–411, 2003.

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

[85] Rudy Prabowo and Mike Thelwall. A comparison of feature selection methods for an evolving rss feed corpus. *Inf. Process. Manage.*, 42(6):1491–1512, 2006.

[86] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1988.

[87] Foster J. Provost, David Jensen, and Tim Oates. Efficient progressive sampling. In *Knowledge Discovery and Data Mining*, pages 23–32, 1999.

[88] Prabhakar Raghavan. Information retrieval algorithms: a survey. In *SODA '97: Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, pages 11–18, Philadelphia, PA, USA, 1997. Society for Industrial and Applied Mathematics.

[89] A. Rakotomamonjy. Variable selection using svm-based criteria, 2003.

[90] J. Resig and A. Teredesai. A framework for mining instant messaging services. In *In Proceedings of the 2004 SIAM DM Conference*, 2004.

[91] Matthew Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70, New York, NY, USA, 2002. ACM Press.

[92] C. J. Rijsbergen, D. J. Harper, and M. F. Porter. The selection of good search terms. *Information Processing and Management*, pages 77–91, 1981.

[93] Ellen Riloff. Little words can make a big difference for text classification. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 130–136, 1995.

[94] J. Rocchio. *Relevance Feedback in Information Retrieval*. Prentice Hall, 1971.

[95] Monica Rogati and Yiming Yang. High-performing feature selection for text classification. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 659–661, 2002.

[96] Sajama and Alon Orlitsky. Supervised dimensionality reduction using mixture models. In Luc De Raedt and Stefan Wrobel, editors, *ICML*, pages 768–775. ACM, 2005.

[97] Gerard Salton. Recent trends in automatic information retrieval. In *SIGIR '86: Proceedings of the 9th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 1–10, 1986.

[98] Jacques Savoy. A stemming procedure and stopword list for general French corpora. *Journal of the American Society for Information Science*, pages 944–952, 1999.

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

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

[101] Kazuhiro Seki and Javed Mostafa. An application of text categorization methods to gene ontology annotation. In *SIGIR*, pages 138–145, 2005.

[102] Rudy Setiono and Huan Liu. Improving backpropagation learning with feature selection. *Applied Intelligence*, 6(2):129–139, 1996.

[103] Mark P. Sinka and David W. Corne. Evolving better stoplists for document clustering and web intelligence. *Design and application of hybrid intelligent systems*, pages 1015–1023, 2003.

[104] Mark P. Sinka and David W. Corne. Towards modernised and web-specific stoplists for web document analysis. In *Proceedings of the IEEE/WIC International Conference on Web Intelligence*, pages 396–402. IEEE Computer Society, 2003.

[105] Ian Soboroff and Charles Nicholas. Collaborative filtering and the generalized vector space model (poster session). In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 351–353, 2000.

[106] Sa Kwang Song, Yun Jin, and Sung Hyon Myaeng. Abbreviation disambiguation using semantic abstraction of symbols and numeric terms. In *Proceedings of 2005 IEEE International Conference on Natural Language Processing and Knowledge Engineering, IEEE NLP-KE '05, 30 Oct.-1 Nov. 2005*, pages 14–19. IEEE Computer Society, 2005.

[107] Pascal Soucy and Guy W. Mineau. A simple feature selection method for text classification. In Bernhard Nebel, editor, *Proceeding of IJCAI-01, 17th International Joint Conference on Artificial Intelligence*, pages 897–902, Seattle, US, 2001.

[108] S. Staab, P. Domingos, P. Mike, J. Golbeck, Li Ding, T. T. Finin, A. Joshi, A. Nowak, and R.R. Vallacher. Social networks applied. *IEEE Intelligent Systems*, 20:80–83, 2005.

[109] Alexander Strehl and Joydeep Ghosh. Cluster ensembles – a knowledge reuse framework for combining partitionings. In *Proceedings of AAAI 2002, Edmonton, Canada*, pages 93–98. AAAI, July 2002.

[110] Masashi Sugiyama. Local fisher discriminant analysis for supervised dimensionality reduction. In William W. Cohen and Andrew Moore, editors, *ICML*, pages 905–912. ACM, 2006.

[111] Yanmin Sun, Mohamed S. Kamel, and Yang Wang. Boosting for learning multiple classes with imbalanced class distribution. In *ICDM*, pages 592–602, 2006.

[112] Kazem Taghva, Jeffrey S. Coombs, Ray Pareda, and Thomas A. Nartker. Language model-based retieval for Farsi documents. In *ITCC (2)*, pages 13–17, 2004.

[113] Kari Torkkola. Feature extraction by non parametric mutual information maximization. *Journal of Machine Learning Research*, 3:1415–1438, 2003.

[114] Thomas Trappenberg, Jie Ouyang, and Andrew Back. Input variable selection: Mutual information and linear mixing measures. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):37–46, 2006.

[115] K. Tumer, K. D. Bollacker, and J. Ghosh. A mutual information based ensemble method to estimate the Bayes error. In C. Dagli et al., editor, *Intelligent Engineering Systems through Artificial Neural Networks*, volume 8, pages 17–22. ASME Press, 1998.

[116] C. J. Van Rijsbergen. *Information Retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow, 1979.

[117] Gang Wang and Frederick H. Lochovsky. Feature selection with conditional mutual information maximin in text categorization. In *CIKM '04: Proceedings of the Thirteenth ACM conference on Information and knowledge management*, pages 342–349, 2004.

[118] Stanley Wasserman, Katherine Faust, and Dawn Iacobucci. *Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences)*. Cambridge University Press, November 1994.

[119] Wikipedia. Affirmative action — wikipedia, the free encyclopedia, 2007.

[120] S. K. M. Wong and Vijay V. Raghavan. Vector space model of information retrieval: a reevaluation. In *Proceedings of the 7th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 167–185, 1984.

[121] Eric P. Xing, Michael I. Jordan, and Richard M. Karp. Feature selection for high-dimensional genomic microarray data. In *Proc. 18th International Conf. on Machine Learning*, pages 601–608. Morgan Kaufmann, San Francisco, CA, 2001.

[122] Jinxi Xu and W. Bruce Croft. Corpus-based stemming using cooccurrence of word variants. *ACM Transactions Inf. Syst.*, 16(1):61–81, 1998.

[123] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 412–420, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.

[124] Yiming and Xin Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49, 1999.

[125] H. Yoon, K. Yang, and C. Shahabi. Feature subset selection and feature ranking for multivariate time series. *IEEE Transactions on Knowledge and Data Engineering,*, 17(9):1186–1198, 2005.

[126] C. T. Yu, K. Lam, and G. Salton. Term weighting in information retrieval using the term precision model. *J. ACM*, 29(1):152–170, 1982.

[127] Lei Yu and Huan Liu. Redundancy based feature selection for microarray data. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 737–742, New York, NY, USA, 2004. ACM Press.

[128] J. L. Yuan and T. L. Fine. Neural-network design for small training sets of high dimension. *IEEE Transactions on Neural Networks*, 9(2):266–280, March 1998.

Index