AN EVALUATION OF EXISTING AND NEW FEATURE SELECTION METRICS IN AUTOMATIC TEXT CATEGORIZATION

by

Şerafettin Taşcı

B.S., Computer Engineering, Boğaziçi University, 2006

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2008

AN EVALUATION OF EXISTING AND NEW FEATURE SELECTION METRICS IN AUTOMATIC TEXT CATEGORIZATION

APPROVED BY:

Assoc. Prof. Tunga Güngör (Thesis Supervisor)	. . . . .
Prof. Fikret Gürgen	. . . . .
Assist. Prof. Murat Saraçlar	. . . . .

DATE OF APPROVAL: 30.06.2008

ACKNOWLEDGEMENTS

First and foremost, I want to express my thanks to Assoc. Prof. Tunga Güngör, for his contribution to this work, for sharing his experience and guiding me in this study. In particular, I am grateful to him for helping me and spending his time for me whenever I want.

I also thank Prof. Fikret Gürgen and Assist. Prof. Murat Saraçlar, for being a part of my thesis jury and giving me feedback about the thesis.

I am also grateful to TÜBİTAK-BİDEB for awarding me their graduate scholarship, which helped me to concentrate better on this thesis.

Finally, I would like to thank my family for their love, support, patience and understanding.

ABSTRACT

AN EVALUATION OF EXISTING AND NEW FEATURE SELECTION METRICS IN AUTOMATIC TEXT CATEGORIZATION

In recent years, the amount of available documents in the electronic medium such as electronic books, digital libraries and email messages increased rapidly. Therefore, the task of organizing and manipulating these resources has gained more importance and has become more difficult. Automatic text categorization is widely used for organizing and manipulating these documents in the electronic medium. However, since the data in text categorization is very high-dimensional, feature selection is crucial to make the task more efficient and precise.

In this study, we make an extensive evaluation of the feature selection metrics used in text categorization by using local and global policies. For the experiments, we use seven datasets which vary in size, complexity and skewness. We use SVM as the classifier and tf-idf weighting for term weighting. We observed that almost in all metrics and datasets, the local policy outperforms others when the number of keywords is low and global policy outperforms others as the number of keywords increases.

In addition to the evaluation of the existing feature selection metrics, we propose new metrics which have shown high success rates especially with low number of keywords. Moreover, we propose a keyword selection framework called *Adaptive Keyword Selection* (AKS). It is based on selecting different number of keywords for different classes and it improved the performance significantly in skew datasets.

ÖZET

METİN SINIFLANDIRMADA KULLANILAN ESKİ VE YENİ ÖZNİTELİK SEÇME METRİKLERİNİN DEĞERLENDİRMESİ

Son yıllarda elektronik ortamda bulunan elektronik kitap, dijital kütüphane ve e-posta mesajları gibi dökümanların miktarı hızla arttı. Bu nedenle, bu kaynakları düzenleme ve idare etme işi daha çok önem kazanmakla birlikte daha da zorlaştı. Metin sınıflandırma, elektronik ortamdaki bu dökümanların düzenlenmesi ve idaresi için geniş ölçüde kullanılmaktadır. Bununla birlikte, metin sınıflandırmada kullanılan veri çok boyutlu olduğu için öznitelik seçme işlemin daha verimli ve kusursuz yapılmasında çok önemlidir.

Bu çalışmada, metin sınıflandırmadaki öznitelik seçme metriklerinin yerel ve genel politika kullanarak kapsamlı bir değerlendirmesini yapıyoruz. Yaptığımız deneyler için; boyutları, karmaşıklıkları ve çarpıklıkları farklılık gösteren yedi adet veri kümesi kullandık. Terim ağırlıklandırması için tf-idf metodu, sınıflandırıcı olarak da SVM (Destek Vektör Makinası) kullandık. Hemen hemen tüm veri kümeleri ve metriklerde, az sayıda anahtar sözcük için yerel politikanın, anahtar sözcük sayısı arttırıldığındaysa genel politikanın daha başarılı olduğunu gözlemledik.

Mevcut öznitelik seçme metriklerinin değerlendirilmesine ek olarak, özellikle az sayıda anahtar sözcük kullanıldığında yüksek başarım sergileyen yeni metrikler tasarladık. Ayrıca, *Uyarlamalı Anahtar Sözcük Seçimi (AKS)* adını verdiğimiz bir anahtar sözcük seçme sistemi tasarladık. Bu yöntem, farklı sınıflar için farklı sayıda anahtar sözcük seçimine dayanıyor ve özellikle çarpık veri kümelerindeki başarımı farkedilir derecede geliştirdi.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii

ABSTRACT iv

ÖZET v

LIST OF FIGURES viii

LIST OF TABLES x

LIST OF SYMBOLS/ABBREVIATIONS xii

1. INTRODUCTION 1

1.1. Text Categorization 1
1.2. Related Work 2
1.3. Motivation 4
1.4. Thesis Organization 5

2. DOCUMENT REPRESENTATION 6

2.1. Input Format and Preprocessing 6
2.2. Term Weighting 9
2.2.1. Boolean Weighting 9
2.2.2. Tf-idf Weighting 10

3. FEATURE SELECTION 12

3.1. Existing Feature Selection Metrics 12
3.1.1. Information Gain (IG) 13
3.1.2. Chi-square Statistics 13
3.1.3. Document Frequency Thresholding 14
3.1.4. TFIDF Keyword Selection 15
3.1.5. Accuracy2 (Acc2) 16
3.2. Proposed Feature Selection Metrics 16

3.2.1. M_1 Method 16
3.2.2. M_2 Method 17
3.2.3. M_3 Method 18
3.2.4. M_4 Method 18
3.2.5. Adaptive Keyword Selection 19
3.2.6. Other Heuristics and Methods Studied 21
3.3. Local vs. Global Policy 22
4. PROCESS DESCRIPTION 23
4.1. Preprocessing Phase 24
4.2. Feature Selection Phase 28
4.3. SVM Classification 29
4.4. Transform SVM Output 31
4.5. Calculate Results 31
5. EXPERIMENTAL SETTINGS 33
5.1. Classifier 33
5.2. Datasets 33
5.3. Performance Measures 36
5.4. External Resources 38
6. RESULTS AND DISCUSSION 39
6.1. General Analysis 39
6.2. Analysis of the Proposed Methods 48
7. CONCLUSIONS AND FUTURE WORK 56
APPENDIX A: EXPERIMENTAL PROCEDURE 57
APPENDIX B: FIGURES OF THE EXISTING FEATURE SELECTION METRICS 61
REFERENCES 70

LIST OF FIGURES

Figure 4.1. Steps of the preprocessing phase 27

Figure 4.2. Steps of the feature selection phase 30

Figure 4.3. Steps of phases 4.3, 4.4 and 4.5 32

Figure 6.1. Micro-averaged F-measure results for Wap dataset for new methods 50

Figure 6.2. Macro-averaged F-measure results for Wap dataset for new methods 50

Figure 6.3. Micro-averaged F-measure results for Hitech dataset for new methods 51

Figure 6.4. Macro-averaged F-measure results for Hitech dataset for new methods 51

Figure B.1. Micro-averaged F-measure results for Hitech dataset for global policy 61

Figure B.2. Macro-averaged F-measure results for Hitech dataset for global policy 61

Figure B.3. Micro-averaged F-measure results for Hitech dataset for local policy 62

Figure B.4. Macro-averaged F-measure results for Hitech dataset for local policy 62

Figure B.5. Micro-averaged F-measure results for Hitech dataset for new methods 63

Figure B.6. Macro-averaged F-measure results for Hitech dataset for new methods 63

Figure B.7. Micro-averaged F-measure results for Reuters dataset for global policy 64

Figure B.8. Macro-averaged F-measure results for Reuters dataset for global policy 64

Figure B.9. Micro-averaged F-measure results for Reuters dataset for local policy 65

Figure B.10. Macro-averaged F-measure results for Reuters dataset for local policy 65

Figure B.11. Micro-averaged F-measure results for Reuters dataset for new methods..... 66

Figure B.12. Macro-averaged F-measure results for Reuters dataset for new methods.... 66

Figure B.13. Micro-averaged F-measure results for Wap dataset for global policy 67

Figure B.14. Macro-averaged F-measure results for Wap dataset for global policy..... 67

Figure B.15. Micro-averaged F-measure results for Wap dataset for local policy..... 68

Figure B.16. Macro-averaged F-measure results for Wap dataset for local policy..... 68

Figure B.17. Micro-averaged F-measure results for Wap dataset for new methods 69

Figure B.18. Macro-averaged F-measure results for Wap dataset for new methods..... 69

LIST OF TABLES

Table 2.1. Format of the datasets that are used in this study 7

Table 2.2. Some of the stopwords that are used in this study..... 9

Table 4.1. Text files that are used to store fundamental matrices 28

Table 5.1. Summary of the used datasets..... 34

Table 5.2. Class distributions of the training sets of the datasets 35

Table 5.3. Class distributions of the test sets of the datasets 36

Table 6.1a. Micro-averaged F-measure results for Classic3 and Wap datasets 42

Table 6.1b. Macro-averaged F-measure results for Classic3 and Wap datasets..... 43

Table 6.2a. Micro-averaged F-measure results for Hitech and LA1 datasets..... 44

Table 6.2b. Macro-averaged F-measure results for Hitech and LA1 datasets..... 45

Table 6.3a. Micro-averaged F-measure results for Reviews and Reuters datasets..... 46

Table 6.3b. Macro-averaged F-measure results for Reviews and Reuters datasets..... 47

Table 6.4. Micro and Macro-averaged F-measure results for RCV1 48

Table 6.5. Micro-averaged F-measure results for Adaptive Keyword Selection (AKS)... 53

Table 6.6. Macro-averaged F-measure results for Adaptive Keyword Selection (AKS) .. 54

LIST OF SYMBOLS/ABBREVIATIONS

c	Category
c_i	Category i
d	Document vector
d_j	Document j
f_{max}	Globalization function that selects the maximum of the inputs
M_i	Proposed method i
n	Total number of documents
n_i	Number of documents term i appears in
$P_r(c_i)$	Probability of a document to have class label c_i
$P_r(t)$	Probability of a term to appear in a document
$P_r(c_i \mid t)$	Probability of a document to have class label c_i given that term t appears in that document
$P_r(c_i \mid \bar{t})$	Probability of a document to have class label c_i given that term t does not appear in that document
t_k	Term k
tf_i	Frequency of term i
tf_{ij}	Frequency of term i in document j
w_i	Weight of term i
$w_{i,j}$	Weight of term i in document j
Acc2	Accuracy2
AKS	Adaptive Keyword Selection
CHI	Chi-square Statistics
DF	Document Frequency
FN	False Negatives

FP	False Positives
GR	Gain Ratio
HTML	Hyper Text Markup Language
idf	Inverse Document Frequency
IG	Information Gain
k-NN	k-nearest neighbor
LLSF	Linear Least Squares Fit
RBF	Radial Basis Function
RCV1	Reuters Corpus Volume 1
STW	Supervised Term Weighting
SVM	Support Vector Machines
tf	Term Frequency
TN	True Negatives
TP	True Positives
TREC	Text Retrieval Conference

1. INTRODUCTION

1.1. Text Categorization

In recent years, the amount of available documents in the electronic medium such as electronic books, digital libraries and email messages increased rapidly. Therefore, the task of organizing and manipulating these resources have gain more importance and became more difficult. For this task, many machine learning and information retrieval methods have been proposed and promising results were taken by some of these methods.

Text categorization is the task of automatically assigning documents to some predefined categories. For text categorization, supervised machine learning techniques are widely used in recent years such as bayesian methods, decision trees, neural networks, support vector machines, etc.. SVM among these techniques have a special importance since recently it is shown by different researchers that it gives the best results for text categorization problem.

In text classification, one typically uses a ‘bag of words’ model: each position in the input feature vector corresponds to a given word or phrase. Since generally there are thousands of words (features) in a document corpus, the data is of a very high dimensionality. This high dimensionality is an important challenge for most of the learning methods. Therefore, feature selection is broadly used in text categorization systems for the purpose of reducing the dimensionality. Dimensionality reduction has meny benefits such as improving the interpretability of data, reducing the time and storage requirements and speeding up the learning process. Moreover, it may improve the classification accuracy since it can prevent overfitting by eliminating the terms that are useless or misleading for the classifier.

Feature selection on textual data is mostly based on feature ranking in which all features are ranked by a metric that estimates their importance and then the ones with the highest

ranks are selected. In literature, there are many feature selection methods most of which are borrowed from the information theory. We will study some of these methods in this thesis.

A fundamental factor that affects the success of text categorization is the dataset which may vary in size, number of terms and skewness. Especially, skewness in class distributions is a major determinant of the classification performance. High skew datasets are particularly hard since the common classes may dominate the rare classes. Therefore, feature selection and document classification can be distorted in a way to classify common classes perfectly while ignoring the rare classes.

The next phase after feature selection is term weighting in which document weights for the selected features are computed. In term weighting, the most widespread method is Tf*idf which uses frequencies of terms in a document and inverse of the number of documents a feature is seen for the computation of the weights of the selected keywords in all documents. Besides Tf*idf Weighting, Boolean Weighting is also used very frequently for text categorization since it is simpler and gives satisfactory results in most cases. After term weighting phase is complete, the learning algorithm is used to learn classification rules from training data and then predict the categories of the test data.

1.2. Related Work

Text categorization is a learning task where the aim is to find the categories of some unlabeled documents by using a labeled training set of documents. Therefore, most of the machine learning algorithms such as support vector machines (SVM), neural networks, Naive Bayes and k-nearest heighbor algorithm have been used for text categorization. There are studies in the literature where these learning algorithms have been compared [9, 12]. Most of these studies [6, 9, 11] have shown that SVM is generally the top performer in text classification. Therefore we use SVM as our learning algorithm in this study.

However, regardless of the learning algorithm, text classification is still a very hard problem since the dimensionality of the data is very high in text categorization. Due to this

reason, feature selection is a fundamental issue in most classification problems including non-text domains. When the dimensionality of the data is low, wrapper methods such as genetic search are used [7]. However, when dimensionality is large as in text categorization, feature scoring methods are used since wrapper methods are impractical with high-dimensional data.

There are numerous study on feature selection in text categorization. Most of the popular feature selection metrics have been evaluated and compared in these studies [1, 2]. However there are many variations in these experiments such as dataset selection, policy, classifier, etc...

For example, in the study of Yang and Pedersen [1], five of the popular feature selection metrics are evaluated on the Reuters and Ohsumed datasets. In this study, they use *k*-NN and LLSF as classifier instead of SVM which is an important deficiency of their study. Moreover, this study is based on feature selection with global policy and no comparison with local policy exists.

In the famous study of Forman [2], local policy is considered and it makes a comprehensive evaluation of many feature selection metrics for the high-dimensional text classification domain. He uses SVM as classifier and includes many type of datasets including skew datasets as well as homogenous ones. Another important focus of this paper is the contribution of a new feature selection metric called “Bi-normal Seperation” which is claimed to be very successful in this study. However, despite dataset and metric diversity this study also lacks the comparison of local and global policy. It is also clear that we cannot use the results of [1] and [2] for a comparison of local and global policy since their experimantal settings (datasets, classifiers) are totally different.

An interesting study that includes the comparison of local and global policy is done by Debole and Sebastiani [3]. In this study, they focus on term weighting using the feature selection scores of the features and they compare the results of different feature selection metrics with local and global policy. They report that global policy performs better than local policy but they do not give a detailed comparison using different number of keywords. In

addition, they only use Reuters dataset and it is hard to generalize their findings to other datasets with different class sizes and skewness.

Finally in [8], Ozgur et al. makes the comparison of local and global policy in keyword selection by using SVM with datasets of different skewness and sizes. They compare the results of selecting different number of keywords.Iin this study, they compare the local and global versions of a simple feature selection metric which they call 'tf-idf keyword selection' that is based on selecting the features with the highest tf-idf scores. However, classical feature selection metrics is not considered in this study any. Therefore, it is not possible to say anything about the performances of local and global policy using the popular feature selection metrics such as IG, CHI, etc..

In addition to the studies aimed at the comparison of the existing feature selection metrics, there are also ones that try to propose new methods for keyword selection in text categorization. One such example is [2] where Forman proposes a method called Bi-normal Seperation which is especially successful in high-skew datasets. Another example is Gain Ratio (GR), which is acquired by normalizing IG score of a term by its entropy.

Finally there are newer studies that propose the use of supervised techniques for term weighting where the scores of terms in feature selection phase is also used in term weighting phase [3, 13]. In [3], Debole and Sebastiani proposes a method called Supervised Term Weighting (STW) where they replace the idf part of the tf-idf term weighting function by the score of the term that is calculated in feature selection phase by a method such as IG, CHI or GR. Likewise in [13], Soucy and Mineau introduce a method called 'ConfWeight' which is a weighting method based on statistical estimation of a word importance for a particular categorization problem.

1.3. Motivation

In this thesis, I made an extensive comparison of popular feature selection methods on different datasets that have different class distributions, number of categories and sizes. While

making the comparison, I also compared the policies used in feature selection: local policy and global policy. In local policy, each category has a different set of keywords (features) while in global policy the reduced feature set is the same for all categories. Each feature selection method that I examined in this thesis has a version for each policy. In fact, the global version of a feature selection method is calculated from its local version by using some globalization techniques such as taking sum, weighted sum or maximum of the local scores of the features in different categories. In this thesis, I used $f_{max}(t_k) = \max_{i=1}^{|c|} f(t_k, c_i)$ as the globalization method where t_k is a term and c_i is a category. It was claimed by Debole and Sebastiani [3] that f_{max} consistently outperformed other globalization techniques.

In addition to the evaluation of the feature selection policies, we propose some new feature selection metrics which resemble the Acc2 metric that was studied by Forman[2]. These newly proposed metrics are two-sided metrics (i.e. they consider the negative features as well as positive features) and are based on the difference of the distributions of a term in the documents belonging to a class and the documents not belonging to that class. They are all local metrics and have shown good performances even at a small number of keywords. This makes them precious especially when the practitioner is constrained to use a small number of keywords.

We also propose a feature selection framework called Adaptive Keyword Selection (AKS) which selects different number of keywords for classes that have different sizes. It is inspired from the observation that classification performances are better with high number of keywords in datasets that contain an abundant number of examples for each class while the performances are better with low number of keywords in skew datasets that contain very few examples for some of the classes. In accordance with our expectation, it has shown significant improvements on skew datasets that have a limited number of training instances for some of the classes.

I have also conducted experiments on Supervised Term Weighting (STW) for checking the success of the methods proposed in this thesis with STW and improving the simple

supervised method proposed in [3]. However, I could not get any further improvement on these experiments.

1.4. Thesis Organization

The outline of the thesis can be summarized as follows:

In the next section, we describe the preprocessing steps that are applied to a raw dataset and then give the details of term weighting. In Section 3, we give information about the current and newly proposed term selection methods. We talk about the logic behind them and also give their formulations. We also talk about the local and global policy in feature selection. In Section 4, we describe the steps of the text categorization procedure that were applied in more detail. In Section 5, we describe our experimental settings; the document datasets we have used in the experiments, evaluation metrics and classifier. In Section 6, we give the results we have obtained and also give a comparative and detailed discussion of the results. Then we conclude our thesis and give future directions of study in Section 7.

2. DOCUMENT REPRESENTATION

2.1. Input Format and Preprocessing

In order to use learning algorithms on textual data, first of all, this data should be transformed into a suitable format that can be handled by these algorithms. The most widely used format for representing textual data is Vector Space Model that was introduced by Salton & Buckley[14].

In Vector Space Model, each document is represented as a vector. Each dimension corresponds to a separate term. If a term occurs in the document, its value in the vector is non-zero. Several different ways of computing these term weights have been developed. One of the most known methods is tf-idf weighting which is also used in this thesis.

Table 2.1. Format of the datasets that are used in this study

documentstart	documentstart
1	10
topicstart	topicstart
cisi	cisi
economy	topicend
topicend	bodystart
bodystart	the
the	purpose
present	...
world	science
...	bodyend
study	documentend
abroad	documentstart
bodyend	13
documentend	...

Input of a text categorization system will be datasets that contain documents and their category informations. In this thesis, all datasets are expected to conform to the format shown in Table 2.1.

In this format, there are specific words for representing parts of the input data. For example, ‘documentstart-documentend’ block represents the beginning and ending points of a document. The number below the ‘documentstart’ line shows the ID of the document. ‘topicstart-topicend’ block enumerates the topics of the document while ‘bodystart-bodyend’ block contains the body of the document.

If the dataset is not in this format, it is converted into this format by means of a small program doing the conversion task. Of course this format is not complete and needs more processing to convert the dataset into Vector Space Model. Moreover, some additional filtering methods are required to decrease the high dimensionality of the data and to arrive at a standard format. For preparing the datasets, the following steps are performed:

- Parse the documents and remove HTML tags if necessary
- Convert all letters to lowercase to avoid the duplicates of the same words (i.e. both of the words ‘book’ and ‘Book’ are converted to the word ‘book’.)
- Discard non-alphabetic characters such as numbers, date, etc..
- Remove stopwords (see Table 2.2) since they occur so frequently that they cannot be the real discriminators of any class (Note: Occasionally, this procedure may remove some of the important words. For example, ‘can’ may be important when it means ‘a metal box’ and the document is related with ‘industry’)
- Use Porter’s Stemmer to stem the words: this task is a common method since it reduces dimensionality of data by merging various word forms such as plurals and verb conjugations into one distinct term. It generally eliminates derivational morphemes as

well as inflectional morphemes. (Example: the words ‘computer’, ‘computing’ and ‘computes’ are all stemmed into the word ‘comput’).

- Calculate the weights of the words for each document by means of a term weighting method so as to represent each document d as $d = (w_1, w_2, w_3, ..., w_n)$ where w_i is the weight of the i^{th} term in the document d.

Table 2.2. Some of the stopwords that are used in this study

a	be	from	of	this	who
about	by	how	off	to	will
an	com	in	on	was	with
are	de	is	or	what	und
as	en	it	that	when	the
at	for	la	the	where	www

2.2. Term Weighting

In text categorization, term weighting procedure is generally done by methods that are taken from Information Retrieval community. Some of these methods were evaluated in [4]. Most popular methods are Boolean Weighting and tf-idf Weighting.

2.2.1. Boolean Weighting

Boolean Weighting is a very simple term weighting method. In Boolean Weighting, the weight of a term is 1 if the term appears in a document and 0 if it does not appear in the document. Below is the formulation of Boolean Weighting where $w_{i,j}$ denotes the weight of a term t_i in a document d_j and $count(t_i, d_j)$ is the number of occurrences of term t_i in document d_j.

$$w_{i,j} = \begin{cases} 1 \text{ if } count(t_i, d_j) > 0 \\ 0 \text{ otherwise} \end{cases} \quad (2.1)$$

The primary advantage of Boolean Weighting is that it is very easy to compute. On the other hand, it makes no discrimination between a word that occurs many times in a document and a word that occurs only once. Moreover, it does not consider the number of documents in which the word appears. Due to these reasons, it is not as successful as tf*idf weighting[4]. Therefore we did not use it in our experiments.

2.2.2. Tf-idf Weighting

Opposed to the Boolean Weighting, Tf-idf Weighting is a more complex weighting approach that is used widely in text categorization research. It does not have the deficiencies of Boolean Weighting since it conforms to the monotonicity assumptions below, that were described by Zobel and Moffat[15].

a) Rare terms are more important than frequent terms
b) Multiple appearances of a term in a document is more important than single appearances.
c) Length of the document should not affect the importance of the terms.

The observations above are named as Inverse Document Frequency (idf) assumption, Term Frequency (tf) assumption and length-normalization assumption.

Tf-idf Weighting takes into account not only the frequency of a term within a document but also the frequency of a term throughout all the documents in the corpus. By this way, if a word exists in too many documents, its importance in a document is decreased proportionally. In addition, normalization procedure helps us to normalize the lengths of different documents to a unit length. Below is the formula of Tf-idf Weighting:

$$w_i = tf_i \cdot \log(\frac{N}{df_i}) \tag{2.2}$$

where w_i is the weight of a term in document d, tf_i is the frequency of the term i in that document, N is the total number of documents and df_i is the number of documents term i appears in.

3. FEATURE SELECTION

3.1. Existing Feature Selection Metrics

Since many classification methods including SVM are computationally hard and their computational cost is proportional to the length of document vectors, it is of key importance to use methods that can decrease the dimensionality of the document vectors. In text categorization task, this is done by evaluating the importance of terms by using some metrics and then selecting the subset of all terms corresponding to these more important terms. The metrics that are used for selecting the more important terms from all terms is called feature selection metrics.

Feature selection can help the classifier to avoid overfitting which is a common problem seen on high dimensional data. In addition, it helps to reduce the time and space requirements of the task and also speeds up the learning process.

Most of the feature selection metrics exploit the idea that a term in a category c_i is more important if it is distributed most differently between positive and negative examples of a category. However, the application of this idea may differ greatly which caused the proposal of many feature selection methods. In text categorization, one computes the scores of each term using a feature selection metric, then list them in the decreasing order and select the ones with the highest scores as keywords.

In this thesis, I will consider some of the most successful and recent feature selection methods that can be found in the literature. Some examples are Information Gain (IG), Chi-square (CHI), Document Frequency (DF) Thresholding, Acc2 that is explained in [2] and Tf-idf Keyword Selection which was used by [4] as a simple and rational method for comparison of other methods.

3.1.1. Information Gain (IG)

Information Gain is a very popular term-goodness criterion that is used in the machine learning community. It measures the change in the entropy when a feature exists or not. Therefore, we can say that it measures the number of bits of information obtained for category prediction by knowing the existence of a term in a document.

Let $\{c_1, c_2, ..., c_m\}$ represent the set of categories. Then score of a term t by Information Gain method can be calculated by the formula:

$$\begin{aligned} IG(t) = &-\sum_{i=1}^{m} P_r(c_i).\log P_r(c_i) + P_r(t)\sum_{i=1}^{m} P_r(c_i \mid t).\log P_r(c_i \mid t) \\ &+ P_r(\bar{t})\sum_{i=1}^{m} P_r(c_i \mid \bar{t}).\log P_r(c_i \mid \bar{t}) \end{aligned} \tag{3.1}$$

where $P_r(c_i)$ is the probability of a document to have class label c_i, $P_r(t)$ is the probability of a term to appear in a document, $P_r(c_i \mid t)$ is the probability of a document to have class label c_i given that term t appears in that document and $P_r(c_i \mid \bar{t})$ is the probability of a document to have class label c_i given that term t does not appear in that document.

3.1.2. Chi-square Statistics

Another popular feature selection method is Chi-square statistics metric. In statistics, the Chi-square test is applied to test the independence of two random variables, where two events A and B are defined to be independent if $P(AB) = P(A)P(B)$ or, equivalently, $P(A \mid B) = P(A)$ and $P(B \mid A) = P(B)$.

In text categorization, the two random variables are occurrence of the term t and occurrence of the class c. Chi-square method tests the independence between t and c. Below is the formula for calculating Chi-square score of a term t in a class c:

$$X^2(t,c) = \frac{N.(AD - CB)^2}{(A+C)(B+D)(A+B)(C+D)} \quad (3.2)$$

where A is the number of documents term t appears in and category is c, B is the number of documents term t appears in and category is not c, C is the number of documents term t does not appear in and category is c, D is the number of documents term t does not appear in and category is not c and N is the total number of documents. It is noticeable that in the above formula the CHI score of a term will be zero in a class if the class and term are independent and will be nonzero otherwise.

Since this formula calculates the score of a term for a specific category, we need to combine the scores of a term for different categories. For this purpose, globalization techniques are used to calculate a global score for a term. In this study, I used the globalization technique called "maximum" in which the maximum of the class-based scores of a term is selected as the score of the term since it was claimed that it outperforms the other globalization techniques in [3].

3.1.3. Document Frequency Thresholding

This method is based on the assumption that infrequent terms are not reliable and effective in the category prediction globally. Reason of this assumption is that rare terms may be noise terms that are misleading for the classifier.

Document frequency is the number of documents a term appears in and the terms whose document frequencies are the highest are selected as features to be used in classification. It can be formulated as:

$$DF(t) = n_t \quad (3.3)$$

where n_t is the number of documents term t appears in.

This method has a very low computational cost, since the formula is very simple and the document frequencies are already calculated while performing the term weighting. On the other hand, generally this method is not used as a principle criterion in feature selection. This is due to the important fact that infrequent terms may be better representatives of categories since they are not found in most documents. Therefore they must not be eliminated.

3.1.4. Tf-idf Keyword Selection

This approach is used by Ozgur et al [5] as an easy-to-compute and rational keyword selection method, since their main task was comparing global and local policy on keyword selection, not the comparison of feature selection metrics.

This method is similar to DF Thresholding and based on the idea that terms which have higher tf-idf scores are more informative and discriminative in classification of documents. In this approach, tf-idf scores of a term in all documents are summed up to find the global score of the word. This approach favors the words that occur in fewer documents but have a high frequency within documents.

Below is the formula of tf-idf feature selection:

$$w_{ij} = tf_{ij} . \log(\frac{n}{n_i}) \tag{3.4}$$

$$tfidf(w_i) = \sum_{j=1}^{n} w_{ij} \tag{3.5}$$

where w_{ij} is the weight of a term in document j, tf_{ij} is the frequency of term i in document j, n is the total number of documents and n_i is the number of documents term i appears in.

3.1.5. Accuracy2 (Acc2)

This metric is aimed to find the terms that are the best discriminators of a class. It is based on the difference of distributions of a word in the documents belonging to a class and the documents not belonging to that class. It is symetric in the sense that it uses the absence of words in a class as well as the presence of words in a class. Below is the formula for calculating Acc2 score of a term t in a class c by local policy:

$$\mathrm{Acc2}(t,c) = \frac{A}{n_1} - \frac{B}{n_2} \tag{3.6}$$

where A and B are as defined in CHI; n_1 and n_2 correspond to the number of documents, respectively, belonging to class c and not belonging to class c.

Acc2 metric was studied by Forman[2] and it was reported that it has a performance comparable to that of IG and CHI when the local policy is used.

3.2. Proposed Feature Selection Metrics

In this thesis, the main purpose is to propose a new feature selection metric that is more successful than the existing metrics. For finding such a metric, I concentrated on the answer of the question: “How can we find the keywords for a category that best discriminate the category from others?”. The proposed metrics that can be considered as successful are described in the following sections.

3.2.1. M_1 Method

This method resembles the method that was called as Acc2 in Forman[2]. It is based on the difference of distributions of words in the documents belonging to a class and the documents not belonging to that class. However in Acc2 method, only the number of

documents in which the term exists is taken into account without considering the number of actual occurrences of the term in the documents.

In this method, we multiply two scores calculated by Acc2 method: one score calculated by using the number of documents a term exists and another score calculated by using the actual occurrences of the term in the documents. Below is the formula for calculating M_1 score of a term t in a class c by local policy:

$$M_1(t,c) = \left[\frac{A}{n_1} - \frac{B}{n_2}\right].\left[\frac{C}{t_1} - \frac{D}{t_2}\right] \tag{3.7}$$

where A, B, n_1 and n_2 are as defined in Acc2; C and D are the number of occurrences of the term t in the documents, respectively, belonging to class c and not belonging to class c; t_1 and t_2 correspond to the number of terms, respectively, in class c and in other classes.

3.2.2. M_2 Method

This method is a different version of the Acc2 method where we measure the correlation between a term and a class in a different way. In Acc2, the proportion of documents in a class that contain or not contain a term is considered. However, here we take the documents in the whole corpus in which the term exists as a group and we find the proportion of documents with class label c in this group.

In addition, we multiply it by the term frequency of term t in the whole corpus since without such a modification a very infrequent term can have a similar score with a frequent term that is found in many documents. Needless to say, in such a case the frequent term must have a higher score since it will be useful in the classification of more documents. Below is the formula for calculating M_2 score of a term t having the category label c by local policy:

$$M_2(t,c) = DF(t).\left[\frac{A}{d_1} - \frac{C}{d_2}\right] \quad (3.8)$$

where A and C are as defined in CHI; d_1 and d_2 correspond to the number of documents in which, respectively, the term occurs and the term does not occur.

3.2.3. M_3 Method

In our experiments we have seen that DF Thresholding with local policy gives very good results in skew datasets where there are many categories in the dataset. Therefore we thought that the performance of M_1 Method can be further increased by incorporating the document frequency information of a term. We multiplied the score gathered from M_1 Method with the document frequency of the term to calculate the score of a term in a class. As we expected, in most cases it increased the performance of M_1 Method. We will show these results in subsequent sections. Here we give the formula for calculating M_3 score of a term t in a class c by local policy:

$$M_3(t,c) = DF(t).M_1(t,c) \quad (3.9)$$

3.2.4. M_4 Method

This method also emerged from the analysis of the results of the experiments. In these experiments we have realized that despite the fact that M_1 Method gives very good results for low number of keywords, it is not as good as corpus-based methods when the number of selected keywords is high. We thought that this deficiency of M_1 Method can be handled if we make use of the keywords found by global IG metric. For this purpose, for a given class we selected the first n keywords by M_1 Method where n is the number of documents in that class. Then we selected the remaining keywords from the list of keywords found by global IG

keyword selection. Our aim was to prevent the decrease of success of M_1 Method for high number of keywords and the results justified our expectation. As we thought, in most cases it increased the performance of M_1 Method. Below we give the formula for calculating M_4 score of a term t in a class c by local policy:

$$\begin{aligned} &\text{if } i < n : \text{select keyword(i) by } M_1 \text{ method} \\ &\text{else : select keyword(i) by global IG} \end{aligned} \tag{3.10}$$

3.2.5. Adaptive Keyword Selection (AKS)

The difficulty of different text categorization problems generally varies due to some factors such as class skew, similarity of classes, very large vocabulary and insufficient training examples. Especially, when the number of classes increases, the seperability of them decreases and therefore more training data are required for successful categorization.

In a multi-class environment, probably the number of training examples for different classes will be unequal. In such imbalanced situations, inevitably rare classes will suffer from the inadequacy of positive training examples for them. It will be difficult for a feature selection metric to find many reasonable keywords for rare classes. Therefore selecting too many features will cause overfitting and reduce the performance in such classes.

In this method, we applied the idea that different classes in a dataset may require different number of keywords for best accuracy in the classification task. This idea is rational and intuitive since a class that have hundreds of documents may have enough statistics to select a high number of keywords while a class containing only a few documents is not enough for extracting so many meaningful keywords. Therefore we thought that since SVM classification is a binary classification, we can select different number of keywords for different classes.

However, finding the best number of keywords for each class is not a trivial task. For finding the best number of keywords for each class, first we aggregated the classes into some groups according to their number of keywords: ones having many documents, ones having a fair amount of documents, etc.. After that, we carried out many experiments to find the division points and the best number of keywords for each group. Of course in these experiments we used some simple heuristics such as "Large classes have enough examples for selecting many keywords."

Finally we found out the number of keywords that can be used for different classes that have different number of documents. This may not be the optimal solution but it increased the results of almost all keyword selection metrics in different datasets. Another point to mention is that this method is not meaningful for datasets that have a homogenous class distribution since in such a case, all classes will use the same number of keywords. Therefore we carried out experiments for this method only in skew datasets.

Below is the keyword number selection procedure for classes with respect to the number of documents they contain where n represents the number of documents in a class in the training set:

$$\begin{aligned}
&\text{if } n > 0 \text{ and } n < 30 &&: \text{ use } 20 \text{ keywords} \\
&\text{if } n > 30 \text{ and } n < 100 &&: \text{ use } 100 \text{ keywords} \\
&\text{if } n > 100 \text{ and } n < 200 &&: \text{ use } 500 \text{ keywords} \\
&\text{if } n > 200 \text{ and } n < 500 &&: \text{ use } 1000 \text{ keywords} \\
&\text{if } n > 500 &&: \text{ use } 2000 \text{ keywords}
\end{aligned} \tag{3.11}$$

Basically, this framework selects more keywords as the document number in a class increases. However, we have seen in the experiments that selecting more keywords (e.g. 100) for classes that have less than 10 training instances improved the results slightly. The reason may be that for a class that has such a low number of training documents, a small number of reliable keywords describing the class cannot be determined. Therefore, we acquire a better classification when we use more keywords.

3.2.6. Other Heuristics and Methods Studied

In addition to the methods that were described in the previous sections, I have also tried some other heuristics that were not successful in the experiments:

First one is giving extra importance to the words that are found only in one of the categories. If a keyword is found only in a single class, then it is a perfect discriminator for that category, therefore it must have a very high weight so that other keywords should not be able to mislead the classifier. However, this method did not affect the results much. Possibly, the reason is that there are very few keywords that have this property. Moreover, these keywords are found only in one or two documents.

Another heuristic is using the scores of the words that are calculated in the keyword selection phase instead of the classical tf-idf weighting in the term weighting phase. In fact, this method was proposed by the study of Debole and Sebastiani [3] in which they call it Supervised Term Weighting (STW). However, their study does not cover a wide range of experimental settings. In addition, they propose only one way for their method: using the term selection score of a term instead of the idf part in the tf-idf function. In this study, I have tried some variations of STW and made experiments by using different datasets and feature selection metrics. However, we could not acquire good results by these varied supervised weighting methods or the results of them were not consistent.

The third heuristic is using the aggregation of the scores of different feature selection metrics instead of using the score of a single feature selection metric. We thought that if we aggregate metrics that do not have the same keyword selection logic, combination of them may give better results. However, in the experiments we have seen that aggregation generally acquires a success rate between the success rates of the aggregating metrics. For example, if the 1^{st} metric has 84.1% F1 measure and 2^{nd} metric has 81.5% F1 measure, and then the aggregation of the two metrics has an F1 measure about 83.0%. Due to this reason, we also gave up studying this heuristic.

3.3. Local vs. Global Policy

In text categorization there are two alternative policies in keyword selection in addition to the metrics used for keyword selection. First one is local policy, in which a seperate keyword set is used for each class. This policy helps us to find the most important terms for a class. This approach gives equal weight to each class in the keyword selection phase. So, less prevailing classes are not penalized.

The second policy is global policy, in which a common set of keywords is used which are the most important terms in the whole corpus. This approach favors the prevailing classes and gives penalty to classes with small number of training documents in document corpora where there is high skew.

In most feature selection methods it is possible to use either local or global policy. However, if the local policy is to be used the classifier must be suitable for binary classification since it seperates the documents as “belonging to category c” and “not belonging to the category c”. In this thesis, since binary SVM classifier is used, local policy can be used as well as global policy.

Generally global scores of terms is calculated from the local scores of terms by some globalization techniques. Most widely used globalization techniques are summing the scores of a term for all classes, using a weighted sum by multiplying the probability of a class with the score of the term in that class and then summing them, and finally using the maximum of the class-based scores of a term. In this thesis, I used the globalization technique called “maximum” since it was claimed that it outperforms the other globalization techniques in [3].

4. PROCESS DESCRIPTION

In this section, I will describe the steps of the text categorization process in detail. This section can also be used as a manual for using the project in later research. The overall process consists of independent steps that are connected to each other by using text files. The overall process can be divided into the following steps:

- *Preprocessing:* In this phase documents are preprocessed (stopwords removed, stemming done, document format is arranged, etc..) and then the tf-idf values are calculated for each words in the dataset and the resulting matrices are written to files for using in the next step. Although term weighting is done after feature selection in the literature, calculating the weights of terms in this phase does not change anything.

- *Feature Selection:* In this step, by using the data files created in the previous step, feature selection is applied. For each feature selection method, there is a different C++ class that is used to find the keywords according to that method. After the keywords are selected, the documents are represented in a lower-dimensional space by using only the selected keyword dimensions. These new document representations are written to files in a format understandable by the SVM-Light package.

- *SVM Learning:* In this step, by using the SVM-Light package, the prepared datasets are processed. In other words, training files are used so as to model the classification rules for the categorization of the test documents. Then by using these rules, category prediction is done for the test documents and recorded in text files.

- *Prediction Checking:* In this phase, the actual category information of the test documents are compared with the predicted categories of SVM and by counting the correct and false predictions, precision and recall is computed.

- *F Measure Computation:* In this phase, by using the precision and recall that are calculated in the previous step, the Micro and Macro-averaged F-measures are calculated.

Now we give the details of the steps that are shortly described above. The pseudo code that details the experimental procedure is also given in Appendix A.

4.1. Preprocessing Phase

This is the phase where the documents are read and the necessary files that will be used in the later phases of the project are prepared.

First of all, docTopic and docTerm matrices are created which holds the category information and the tfidf score information of the documents, respectively. After that, the stopwords are read into an array called stoplist from text file.

Next, the file containing train documents called “train-docs.txt” is opened and is processed line by line. In this process, each word is checked to see whether it is a keyword that indicates the starting point of a document, the category of the document or the content of the document. If it is the starting point of a document, numDocs is incremented, id of document is selected and then a new word is read from the file. If it is the keyword indicating the start of the category definition of the document, then the topic names are read one by one and each topic name is added to the topiclist if it does not exist in that list. Moreover, the docTopic matrix is changed and the point in the docTopic matrix corresponding to the current document and topic is set to “1”.

Then the task continues and a new word is read from the “train-docs.txt” file. When the body of the document starts, first the words are checked to see whether they are in the stopwords list. If so; they are skipped. If not; the word is first stemmed using the Porter’s Stemmer. Then if it is not in the “termlist” that keeps the list of words in the corpus, it is added to the termlist. Then docTerm matrix is changed and the point in the docTerm matrix corresponding to the current document and word is incremented by 1.

The process above continues until the end of the “train-docs.txt” file is reached. When it ends, the next step is copying the values in the docTerm matrix into the “tf-train-matrix.txt” that keeps the frequencies of all the terms in all documents. Note that, for now the docTerm matrix contains only the frequencies of the words (not the tfidf scores). We copy the docTerm matrix since in the next step, it will be used to hold the tfidf scores of the words; thus it is necessary to backup the word frequencies for later usage.

Next task is the computation of the tfidf scores of the words in the documents: In this part, the first thing done is the computation of the idf scores. It is simple since we can count the document frequencies of the words by looking at the docTerm matrix and increment the “df” of a word if for a document the term frequency of the word is not zero. Then “idf” score is calculated by the following formula where NUM_DOCS represents the total number of documents.

$$\mathrm{idf}(i) = \log 10(\mathrm{NUM_DOCS} / \mathrm{df}(i)) \qquad (4.1)$$

After the “idf” scores are calculated the tf-idf computation is also straightforward: multiply each point in the docTerm matrix with the “idf” score of that point. Now we have the un-normalized tf-idf scores. For normalization we divide the tfidf scores of each point in a row with the sum of the tfidf scores in that row. By this way we get the tfidf scores for all the words in all documents.

Finally, the calculated scores and information is written to text files for being used in later steps of the project. There are five such files:

- "Terms.txt": keeps the list of the words where each line has three columns; id of the word, idf score of the word and the word itself.
- "topics.txt": keeps the list of the topics where each line has two columns; id of the topic and the topic itself.
- "train-docIDs.txt" : keeps the id information of the training documents
- "train-topic-matrix.txt": keeps the topics of the documents. Since a document may have more than one topic, its dimension is "NUM_DOCS * NUM_TOPICS" and an entry in a row is "1" if that document has the topic and "0" otherwise
- "train-data-matrix.txt": keeps the "tfidf" scores of the terms in the documents. Its dimension is "NUM_DOCS * NUM_TERMS"

After the training data is processed, "docTerm" and "docTopic" matrices are deleted and the tasks explained above are done for test documents. The steps are almost same with some minor differences:

First of all, since the topiclist and termlist must be the same for both training and test documents, here no new terms or topics are added to the topiclist or termlist. If a new word is encountered, it is simply ignored since it does not affect the results much. Another difference is that, here idf scores are not calculated again. Instead the idf values of the words from the training phase are used and as in the training phase, they are multiplied with the tf scores gathered from the test data to acquire the tfidf scores of all words.

After the tfidf scores of the test documents are calculated, "test-topic-matrix.txt" and "test-data-matrix.txt" are written according to the docTerm and docTopic matrices that are formed in the processing of test data.

The processes that are done in this phase are also illustrated by Figure 4.1.

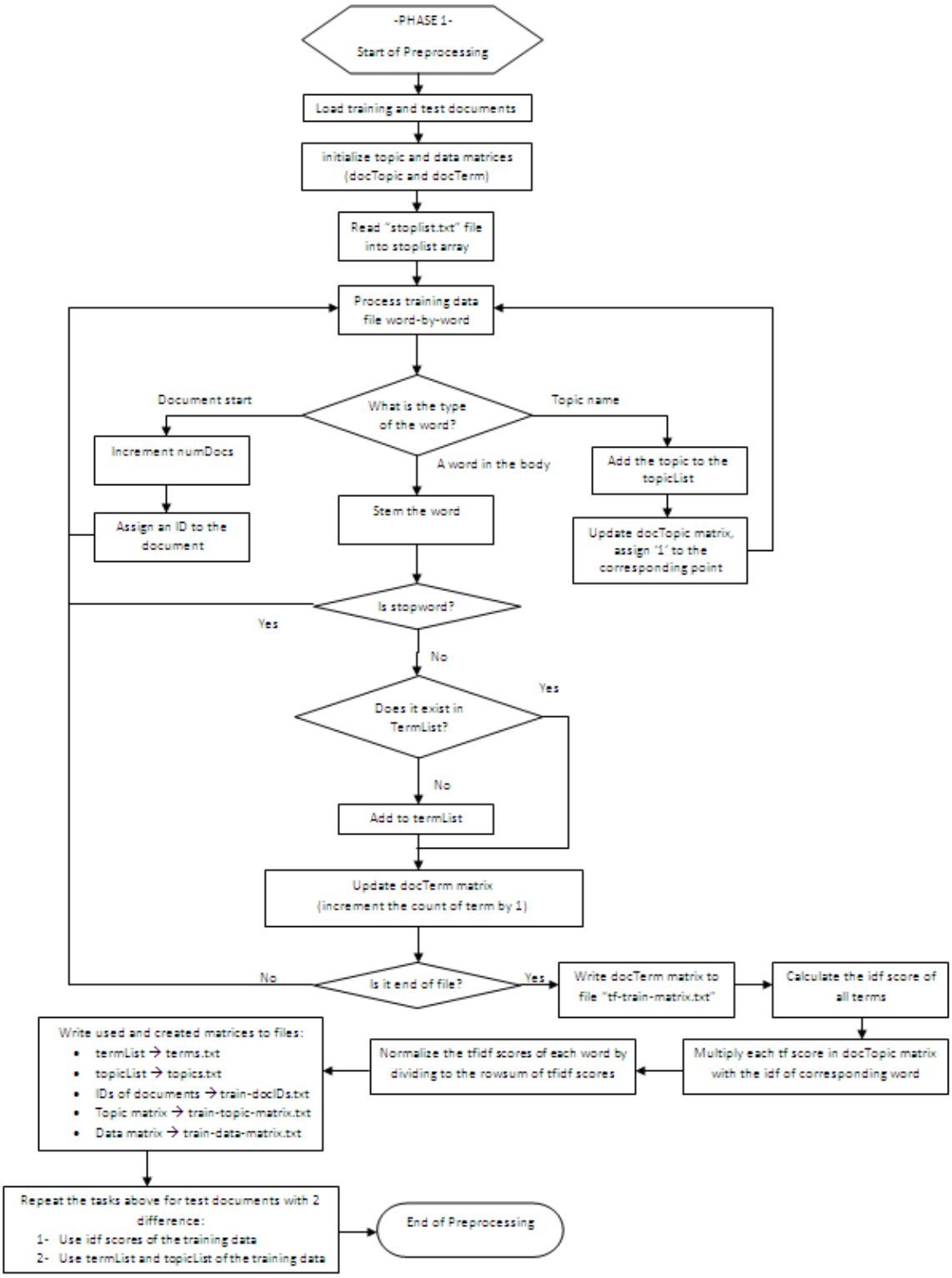


Figure 4.1. Steps of the preprocessing phase

4.2. Feature Selection Phase

In this phase, the aim is to select a given number of keywords from the term list of the data by using some feature selection metrics and then regenerate the training and test data matrices in this lower-dimensional space which will be given as inputs to the SVM-Light classifier. Firstly, the text files produced in the preprocessing stage are read into arrays and matrices. In Table 4.1 you can see the list of text files and the matrices produced by them.

Table 4.1. Text files that are used to store fundamental matrices

File Name	**Matrix Name**
terms.txt	termlist
train-data-matrix.txt	docTerm
test-data-matrix.txt	docTermTest
train-topic-matrix.txt	docTopic
test-topic-matrix.txt	docTopicTest
tf-train-matrix.txt	tfMatrix

After the files are read, the next step is the creation of some new arrays and matrices that will be used later. Below are a list of these structures and how they are filled:

- *TermTopicDoc*: It is of size NUM_TERMS*NUM_TOPICS and it says how many times term t_i exists in classc_j. It is calculated by counting the frequencies of words in tfMatrix by making use of docTopic matrix to understand which documents belong to that category.
- *TermFreq*: It is of size NUM_TERMS and it says how many times term t_i exists in the whole training set. It is calculated by summing up the elements in the rows of termTopicDoc.
- *TopicFreq*: It is of size NUM_TOPICS and it says how many of the documents belong to classc_j in the training set. It is calculated by summing up the elements in the columns of termTopicDoc.

After we prepare all the information above, we can easily make the calculations for feature selection methods such as Information Gain, Chi-square, DF Thresholding, etc... In the next step we compute the scores of all words for feature selection using one of the above methods and keep them in an array of size NUM_TERMS called “scores[]”.

Afterwards, we form another array called “keywords” and by making use of a ‘for’ loop we select the keywords from all words by taking the ones having the highest scores. Then, we use selection sort to sort the keywords based on their scores.

Finally, the sorted keywords are written in a file called “terms.txt”. Then the training and test data files are rewritten by using only the words in the keyword list. In other words, before keyword selection the number of columns in the data matrix of a document was equal to NUM_WORDS. But now, the columns except the ones corresponding to the keywords found are eliminated and now there are NUM_KEYWORDS columns in a document vector. The files calculated this way are then written to files called “train_svm.txt” and “test_svm.txt”.

The processes that are done in this phase are also illustrated by Figure 4.2.

4.3. SVM Classification

This phase is where SVM_light package is used. SVM_light takes the files “train_svm.txt” and “train-topic-matrix.txt” and by making use of the data in these files, it creates a model for category prediction of the text documents for each class (Note that SVM-light makes binary classification). Then it takes “test_svm.txt” and by making use of the models for each category, it predicts the categories of the test documents.

The predictions of SVM_light for each class are written in a separate file called “output_d1_fold_topic_X” where “X” is the topic number. In these output files a score is found for all documents. If the score is positive then it means the document is assigned to that topic. Otherwise the document is not assigned to that topic.

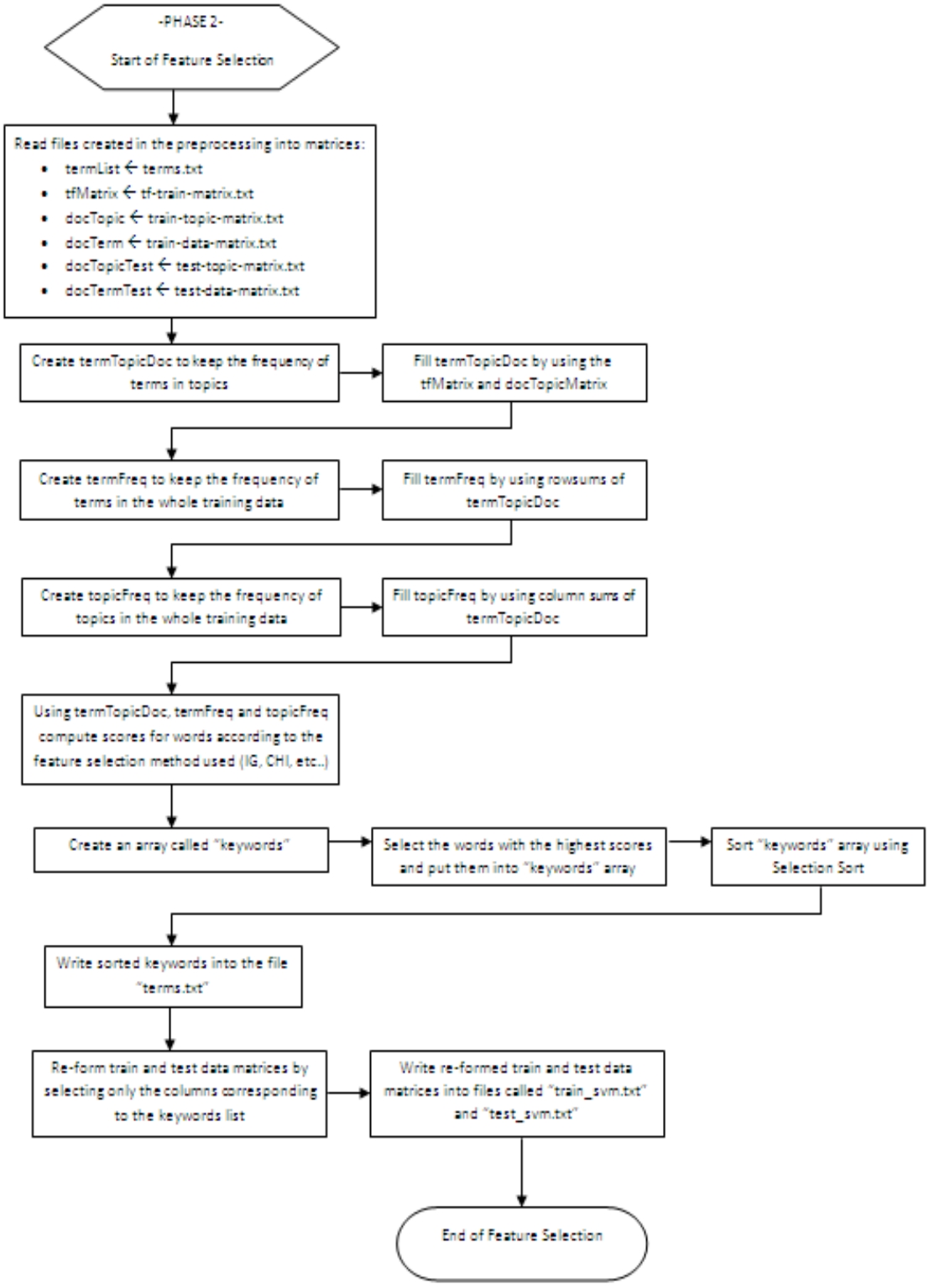


Figure 4.2. Steps of the feature selection phase

4.4. Transform SVM Output

In this phase, the input is the output files of the SVM-light ("output_d1_fold_topic_X") and the output is a single file called "test-topic-assign.txt". As it is mentioned earlier, in the output files of SVM-light, a score is found for all documents. If the score is positive then the document is assigned to that topic. Otherwise the document is not assigned to that topic.

In this phase, we form a topic prediction matrix for test documents called "docTestAssignment" by looking all of the output files and then write this matrix to the file called "test-topic-assign.txt".

4.5. Calculate Results

In this final phase, we read the files "test-topic-assign.txt" and "test-topic-matrix.txt" into matrices "docTestAssignment" and "docTopicTest" respectively and then we compare the matrices to find the number of True Positives, False Negatives and False Positives in the assignments of SVM classifier for text documents.

Then, precision and recall is calculated using the TP, FN and FP. After the calculation of precision and recall, the final step is the calculation of Micro and Macro-averaged F-Measures. They are also calculated and written to files called "microf.txt" and "macrof.txt" respectively.

The processes that are done in phases 4.3, 4.4 and 4.5 are also illustrated by Figure 4.3.

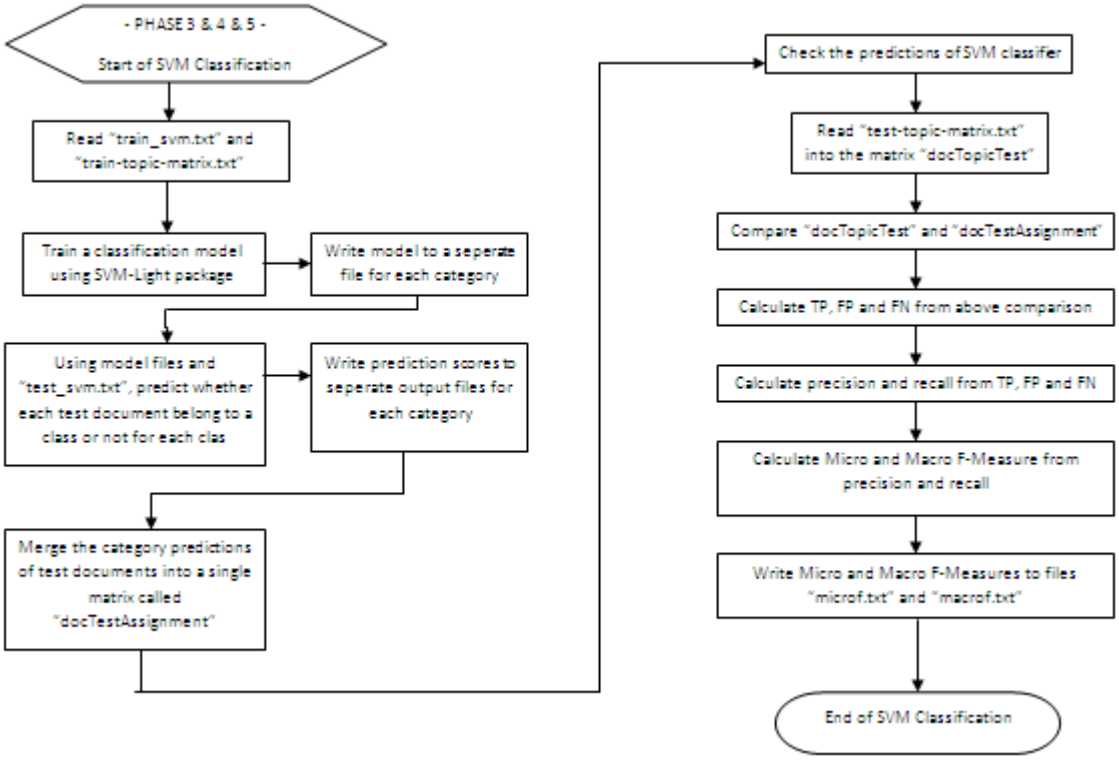


Figure 4.3. Steps of phases 4.3, 4.4 and 4.5

5. EXPERIMENTAL SETTINGS

5.1. Classifier

For this study, I used SVM classifier which has proven its superiority to other classification methods [2, 6, 9] despite it is a fairly new method that has started to be used in recent years in text categorization.

SVM was designed for solving binary classification problems by finding a hyper plane in n-dimensional space that separates positive and negative examples with the largest possible margin. By this way, the generalization error on unseen examples is minimized. SVM can be used with different kernels such as linear, polynomial, RBF, etc. which change the shape of the decision surface.

In this thesis, I used the SVM-Light package which is a popular and efficient implementation of support vector machines that was introduced by Joachims [6]. Since SVM is a classifier for 2-class problems, we divided the classification task into n binary classification problems where n is the number of categories. I used the default parameter settings of SVM-Light which means linear kernel is used.

5.2. Datasets

In this study, we have used seven document corpora that are used extensively in text categorization research. In each dataset, we apply a preprocessing as described in Section 2.1. Table 5.1 shows essential information about the datasets used in this study.

Classic3 dataset contains some medical journals, information retrieval documents and aeronautical system papers. It is a simple dataset for text categorization not only due to the small number of classes but also since it is highly homogenous with very similar class

distributions. However, it is not a very useful dataset for text categorization research since even without any feature selection; it can achieve very high Micro-averaged (99.4%) and Macro-averaged (99.4%) F-measures.

Table 5.1 Summary of the used datasets

Dataset	# of Training Docs	# of Test Docs	# of Classes	# of Terms	Skewness
Classic3	2699	1192	3	10930	homogenous
Wap	1047	513	20	8064	highly skewed
Hitech	1530	770	6	18867	medium
LA1	2134	1070	6	25024	medium
Reviews	2708	1361	5	31325	medium
Reuters	9603	3299	90	20308	highly skewed
RCV1	23149	781265	101	46487	highly skewed

On the other hand, Wap dataset is highly skewed with varying class distributions (see Table 5.2 and Table 5.3). It is a collection of web pages taken from Yahoo and classified as a part of the Web ACE project. Another difficulty of this dataset is that all classes are very similar to each other in content which make it a hard dataset for text classification.

Hitech, LA1 and Reviews datasets are parts of the TREC collection. Hitech consists of newspaper articles that have classes such as health, computers, research, technology, etc… LA1 dataset consists of documents from the Los Angeles Times newspaper and it has classes such as financial, sports, national, foreign, magazine, etc... Reviews dataset consists of articles from the San Jose Mercury newspaper. It also has classes such as food, music, movie, television, etc… All of these three datasets are similar in the sense that they all have similar number of classes and class distributions. The hardness of these datasets is that they have a large number of terms compared to the number of documents. This situation increases the sparseness of the term-document matrix representation and makes it harder and more time consuming for classifier to accomplish classification.

Table 5.2 Class distributions of the training sets of the datasets

Dataset	Class Distributions
Classic3	999 971 729
Wap	10 55 23 36 4 134 208 30 12 8 61 27 112 44 61 65 10 26 96 25
Hitech	322 76 286 402 320 124
LA1	370 226 182 628 492 236
Reviews	666 754 924 90 274
Reuters	55 433 212 181 37 8 24 87 1 14 5 78 124 11 2877 1650 47 16 140 111 197 126 369 55 30 13 1 39 50 75 389 75 69 101 538 347 75 35 37 2 30 2 9 35 94 5 16 18 18 1 5 131 6 23 41 21 40 16 3 14 37 46 12 45 21 16 20 13 19 3 15 5 9 2 8 10 4 5 2 10 4 3 1 2 2 2 4 1 1 1
RCV1	279 3449 1294 732 5882 922 10786 2366 4179 449 6970 471 674 190 2541 699 1596 65 381 1133 947 1058 343 679 943 1205 1462 1647 606 400 641 363 52 1004 346 1930 793 311 43 94 167 187 8 443 1172 166 196 407 1255 1115 246 853 106 399 160 286 312 41 202 166 1508 120 285 49 138 38 437 45 35 913 40 293 31 142 172 197 76 102 34 233 135 62 49 90 92 51 23 166 54 17 37 13 59 66 43 12 15 3 2 6 2

Reuters dataset is in fact the famous Reuters-21578 v1.0 document collection which is the standard benchmark for text classification research. It was collected from the Reuters newswire in 1987 and it also has a diverse and skewed class distribution as Wap dataset. This dataset is divided according to the modified Apte splitting method for convenience and compatibility to the results of other studies that have used this dataset. Indeed, it had 135 classes initially. But the classes that do not have at least one training and one test instance were removed.

Table 5.3 Class distributions of the test sets of the datasets

Dataset	Class Distributions
Classic3	461 427 304
Wap	5 21 10 18 1 62 133 14 6 3 30 13 56 21 30 32 3 11 34 10
Hitech	163 40 143 201 161 62
LA1	185 115 91 315 246 118
Reviews	333 379 464 47 138
Reuters	18 149 71 56 14 6 10 37 1 11 2 33 47 5 1087 719 18 4 34 28 89 36 117 18 19 13 1 20 18 24 189 30 28 35 179 131 30 24 12 1 10 1 4 23 30 7 11 12 9 1 3 44 2 2 12 8 14 6 3 5 17 21 3 14 13 11 12 10 10 3 14 4 7 1 1 6 1 1 1 4 3 1 2 4 2 1 2 1 2 1
RCV1	8289 116471 47402 25304 198938 31231 370541 79524 147606 16586 232297 16770 23651 5929 82899 26053 52038 2112 11563 31086 36463 39451 11535 26421 27242 42169 51355 55231 21351 12234 20639 20309 4248 36735 11186 71162 24610 11819 2041 2088 5492 6416 192 14889 40983 4133 6452 15361 41875 31500 5625 26552 3695 22812 7250 9986 11043 1074 7204 5102 46200 4715 11202 1871 8266 2086 17876 1991 2072 34404 2096 8364 1179 4529 6089 5833 2560 2831 1172 8609 3743 2563 3258 2712 2757 1818 657 5332 2236 922 2373 831 2301 2349 1658 364 376 108 38 307 258

Finally, RCV1 (Reuters Corpus, Volume 1) dataset is also collected from the Reuters newswire in 2000 for use in Natural Language Processing, Information Retrieval, and Machine Learning research. It has about 800000 documents which is much bigger than the well-known Reuters-21578 collection that is widely used in the text classification community. It is also highly skewed and has 101 different classes.

5.3. Performance Measures

There are a large number of performance measures that can be used on text categorization. A very popular and simple measure, "Accuracy", is used as a statistical

measure of how well a binary classification test correctly identifies or excludes a condition. However, especially when there is a high class skew, accuracy is not a good indicator. For example, assume a dataset that contains 2 topics; Topic 1 having 995 and Topic 2 having 5 examples. A classifier which wrongly classifies all documents as belonging to Topic 1 will have an accuracy of 99.5% despite it is erroneous.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

Therefore, text categorization research which generally has high class skew uses more reliable measures such as *precision* and *recall*. *Precision* is the fraction of the documents retrieved that are relevant to the user's information need while *recall* is the fraction of the documents that are relevant to the query that are successfully retrieved.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5.2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5.3}$$

Since both precision and recall measure the quality from a different perspective none should be omitted. Therefore another metric called *F1-measure* is used in Information retrieval community which is simply the weighted harmonic mean of precision and recall. F1-measure can be calculated for each category in a categorization task. Therefore, for avoiding the appearance of many F1-measures, two methods for combining the F1 scores of different classes have been proposed. *Micro-averaged F-measure* gives equal weight to all documents and therefore it is a simple average of all F1 scores whereas *Macro-averaged F-measure* gives equal weight to each category regardless of their frequency and therefore it is affected by the classifier's performance on rare categories.

$$F1 = \frac{2.precision.recall}{precision + recall} \tag{5.4}$$

Since in skew conditions rare categories are harder to classify correctly, Macro-averaged F-measure is generally lower than Micro-averaged F-measure. In other words, a classifier's performance on rare categories is measured more easily by using Macro-averaged F-measure. Because of the fact that each metric has its own benefit, we used both metrics to evaluate the results of our experiments.

5.4. External Resources

In this study, there are 2 main external resources which are in the form of either C or C++ codes that are used in the project implementation:

● *Porter's Stemmer:* It is used in the preprocessing for finding the root morphemes of the words for decreasing the dimensionality of the data to a reasonable number.

Ex: {computer, computing, computes} → root morpheme is "comput"

● *SVM-Light:* This is a popular package implementing SVM and it is used commonly for both binary and multiclass classification. In my project, I used binary classifier of SVM-Light since the task is multi-label, i.e. a document may have more than one categories assigned to it. In binary classification, for each category a document is checked whether it belongs to that category or not.

6. RESULTS AND DISCUSSION

In this section, we will give the results of the experiments in this study and discuss these results. These experiments have not always shown the superiority of a metric or a policy over the others: in different datasets different metrics achieved better results or a policy outperformed others at some metrics and keyword numbers while the other one did better in other cases. Even so, the results carry enough information to extract meaningful and valuable results. Table 7.1a-7.3b show the Micro and Macro-averaged F-measure results for the main six datasets (Classic3, Wap, Hitech, LA1, Reviews and Reuters) respectively. In these tables, we show the results of the popular feature selection metrics as well as the results of the new metrics which are proposed in this thesis. We carried out many experiments with local and global policy using different number of keywords ranging from 10 to 2000. For all experiments tfidf weighting with normalization is used.

6.1. General Analysis

When we look at Table 7.1a-7.4, first thing that we realize is that the highest variance in the F-measures in the table is among the datasets. In other words, for a given method and keyword number, F-measures can be very different for different datasets. This fact is primarily due to the difference in number of classes and skewness in different datasets. We see that in datasets such as Wap, Reuters and RCV1 where there are many classes and class distribution is skew, F-measures are relatively lower than other datasets since having many classes with a skew distribution is a principal hardness for text categorization. Another reason for the F-measure differences among different datasets is the relative hardness of the datasets. For example, despite having same number of classes, and having similar class-distributions Hitech dataset has lower F-measures compared to the LA1 dataset (about 65% vs. 80%).

In addition, when we compare Wap and Reuters dataset, we see that Wap has worse F-measures compared to Reuters dataset (about 70% vs. 80%) despite Wap dataset has less

number of classes (20 vs. 90). We think that this difference results from the difference in the number of documents in the two datasets. That is to say, since Reuters has a much larger training set (9603 documents) compared to Wap dataset (1047 documents), the SVM classifier can train better and thus gives better results in Reuters dataset.

Moreover, it should be noted that there is a higher difference between Micro and Macro-averaged F-measures in the skew datasets (Wap, Reuters and RCV1). This situation can be explained by the fact that Macro-averaged F-measure gives equal weights to all classes while Micro F-measure gives equal weight to each document. Therefore rare classes which are hard to classify correctly decrease the Macro-averaged F-measure.

Another observation about skew datasets is that Macro-averaged F-measure in skew datasets is highest about 100 keywords with local policy while in homogenous datasets it increases as the number of keywords increases. It seems like that this situation is also related with the number of training documents. Rare classes which have only a few documents in training corpus are best classified by using a small number of keywords that are selected locally. Since Macro-averaged F-measure is highly affected by the success of the classifier in rare classes, it increases when we achieve to classify rare classes more successfully.

We also see that using all words instead of keywords gives better results in some datasets such as Classic3 and LA1. We believe that since all classes have many training documents in these datasets, classifier can find enough statistics for almost all features. This situation may cause one to conclude that in these datasets feature selection is unnecessary. However, feature selection is not applied only for increasing accuracy but also for space limitations and a faster text categorization system which is important in practical applications.

In addition, we see that in some datasets such as Classic3, LA1, Reviews and RCV1, accuracies decrease gradually as number of keywords decrease from 2000 to 10. This shows that the keywords that are dropped as the number of keywords is decreased carry information that is useful in the classification of the documents. For deciding the number of keywords to use in these datasets, we may choose a point depending on the application area. If accuracy is

important, we can select more keywords. But if time and space limitations overweigh accuracy, then we should select a smaller number of keywords that give satisfactory results.

In other datasets (Wap, Hitech, Reuters) we see that either Micro-averaged or Macro-averaged F-measure reaches its peak value at a keyword number that is lower than 2000. In these datasets, it means that selecting too many keywords causes overfitting. The reason for overfitting may be the inadequacy of statistics to find so many reasonable keywords. In other words, training set may not be enough for extracting many meaningful features.

When we look at the comparison of existing metrics, we see that most of the time IG gives the best results independent of the dataset. However CHI method is also very successful and it has accuracies comparable to that of IG method. Tf-idf and DF Thresholding are also good when the number of keywords is high. However, they diminish more rapidly than IG and CHI method when the number of keywords decreases. This may be a result of the fact that when there are many keywords, most important ones, i.e. the ones that are best discriminators of classes, are almost always selected. But when we use a very small number of keywords such as 10 or 30, feature selection by DF or Tf-idf method may drop some of the principal keywords which in turn deteriorate the results highly. It is also remarkable that especially in Wap, Hitech and Reuters datasets Acc2 with local policy is more successful than other methods. In fact, this is the main observation that motivated us to study on finding a method that has a similar approach to the Acc2 method.

We also see that while global policy (denoted by ‘(g)’ in the tables) is better than local policy (denoted by ‘(l)’ in the tables) at large number of keywords, while it is generally beaten by local policy when the number of keywords is 500 or lower. This shows that when we select little number of keywords, global policy cannot find the ones that can represent all classes well. Or maybe it is not possible to find a small subset of keywords for representing all classes well. On the other hand, it also shows that local policy is better than global policy at finding a small number of keywords that are best for classification while it selects worse keywords when the number of keywords is high. So if one has time and space limitations local policy should be preferred.

Table 6.1a. Micro-averaged F-measure results for Classic3 and Wap datasets

MICRO_F	# words	10	30	50	100	200	500	1000	1500	2000	all
Classic3	tfidf(l)	0.653	0.895	0.939	0.951	0.959	0.960	0.964	0.965	0.971	**0.994**
	tfidf(g)	0.701	0.873	0.901	0.937	0.956	0.981	0.988	0.992	0.992	0.994
	IG(l)	0.735	0.896	0.918	0.958	0.973	0.986	0.989	0.992	0.991	0.994
	IG(g)	0.702	0.848	0.886	0.956	0.974	0.988	0.991	0.990	0.992	0.994
	CHI(l)	0.638	**0.915**	**0.947**	**0.963**	0.974	0.981	0.987	0.989	0.990	0.994
	CHI(g)	0.732	0.848	0.890	0.956	0.972	**0.989**	**0.991**	**0.992**	**0.992**	0.994
	Acc2(l)	0.787	0.880	0.926	0.958	0.972	0.985	0.991	0.991	0.991	0.994
	Acc2(g)	0.736	0.867	0.916	0.944	0.967	0.984	0.988	0.989	0.991	0.994
	DF(l)	0.745	0.865	0.883	0.917	0.949	0.964	0.973	0.973	0.978	0.994
	DF(g)	0.622	0.800	0.833	0.894	0.943	0.970	0.986	0.992	0.992	0.994
	Method M_4	**0.789**	0.892	0.934	0.956	**0.976**	0.983	0.990	0.990	**0.992**	0.994
	Method M_3	0.766	0.899	0.930	0.955	0.973	0.984	0.989	0.990	**0.992**	0.994
	Method M_2	0.743	0.881	0.920	0.955	0.972	0.984	**0.991**	0.991	0.991	0.994
	Method M_1	**0.789**	0.892	0.934	0.956	**0.976**	0.984	0.989	0.989	0.990	0.994
Wap	tfidf(l)	**0.671**	0.737	0.741	0.738	0.735	0.722	0.746	0.741	0.749	0.752
	tfidf(g)	0.134	0.496	0.587	0.655	0.691	0.721	0.740	0.749	0.743	0.752
	IG(l)	0.685	0.735	0.750	0.742	0.747	0.744	0.742	0.758	0.749	0.752
	IG(g)	0.399	0.526	0.577	0.644	0.746	0.753	0.755	0.756	0.755	0.752
	CHI(l)	0.440	0.714	0.732	0.732	0.720	0.736	0.742	0.756	**0.758**	0.752
	CHI(g)	0.242	0.523	0.540	0.607	0.631	0.712	0.730	0.741	0.749	0.752
	Acc2(l)	0.639	0.728	0.757	0.770	0.758	0.755	**0.752**	0.758	0.752	0.752
	Acc2(g)	0.221	0.476	0.529	0.629	0.697	0.730	0.743	0.753	0.758	0.752
	DF(l)	0.000	0.567	0.704	0.751	0.771	0.747	0.760	0.747	0.747	0.752
	DF(g)	0.000	0.341	0.395	0.543	0.657	0.723	0.756	0.757	**0.758**	0.752
	Method M_4	0.665	**0.739**	**0.769**	0.765	0.767	**0.761**	0.755	0.754	0.754	0.752
	Method M_3	0.610	0.701	0.735	**0.776**	**0.769**	**0.761**	**0.758**	0.748	0.750	0.752
	Method M_2	0.603	0.732	0.738	0.757	0.765	0.759	0.756	**0.760**	0.753	0.752
	Method M_1	0.667	0.738	0.762	0.767	0.758	0.750	0.747	0.749	0.748	0.752

Table 6.1b. Macro-averaged F-measure results for Classic3 and Wap datasets

MACRO F	# words	10	30	50	100	200	500	1000	1500	2000	all
Classic3	tfidf(l)	0.720	0.880	0.935	0.950	0.957	0.959	0.964	0.964	0.970	0.994
	tfidf(g)	0.665	0.871	0.898	0.936	0.953	0.980	0.989	0.992	0.992	0.994
	IG(l)	0.728	0.889	0.912	0.959	0.974	0.986	0.990	0.992	0.991	0.994
	IG(g)	0.665	0.811	0.863	0.955	0.975	0.988	0.991	0.990	0.992	0.994
	CHI(l)	0.706	**0.908**	**0.945**	**0.963**	0.974	0.981	0.987	0.989	0.990	0.994
	CHI(g)	0.709	0.821	0.870	0.956	0.972	**0.990**	**0.991**	**0.993**	**0.992**	0.994
	Acc2(l)	**0.761**	0.867	0.923	0.958	0.972	0.985	0.991	0.991	0.991	0.994
	Acc2(g)	0.690	0.865	0.914	0.944	0.967	0.985	0.988	0.990	0.991	0.994
	DF(l)	0.720	0.848	0.871	0.908	0.945	0.964	0.973	0.973	0.978	0.994
	DF(g)	0.623	0.798	0.831	0.893	0.941	0.970	0.987	0.992	0.992	0.994
	Method M_4	0.756	0.877	0.928	0.956	**0.977**	0.984	0.990	0.991	**0.992**	0.994
	Method M_3	0.737	0.896	0.928	0.955	0.974	0.984	0.990	0.990	**0.992**	0.994
	Method M_2	0.723	0.868	0.918	0.954	0.973	0.984	**0.991**	0.991	0.991	0.994
	Method M_1	0.756	0.877	0.928	0.956	0.976	0.985	0.990	0.990	0.991	0.994
Wap	tfidf(l)	**0.506**	**0.593**	0.565	0.532	0.507	0.495	0.509	0.477	0.483	0.450
	tfidf(g)	0.093	0.208	0.306	0.350	0.412	0.442	0.455	0.468	0.455	0.450
	IG(l)	0.492	0.531	0.548	0.517	0.508	0.508	0.460	0.490	0.482	0.450
	IG(g)	0.052	0.185	0.284	0.375	0.479	0.501	0.473	0.474	0.467	0.450
	CHI(l)	0.493	0.511	0.520	0.509	0.462	0.491	0.475	0.488	**0.491**	0.450
	CHI(g)	0.121	0.239	0.256	0.336	0.375	0.451	0.486	0.469	0.478	0.450
	Acc2(l)	0.435	0.554	0.564	0.551	0.509	0.516	0.488	0.491	0.485	0.450
	Acc2(g)	0.117	0.235	0.278	0.411	0.479	0.489	0.480	0.480	0.492	0.450
	DF(l)	0.000	0.353	0.513	0.550	0.538	0.483	0.524	0.483	0.481	0.450
	DF(g)	0.000	0.053	0.095	0.237	0.326	0.430	0.474	0.470	0.462	0.450
	Method M_4	0.480	0.554	**0.572**	0.558	0.524	0.497	0.471	0.468	0.467	0.450
	Method M_3	0.376	0.497	0.546	0.577	0.527	**0.522**	**0.495**	0.478	**0.481**	0.450
	Method M_2	0.386	0.562	0.539	0.558	0.524	0.519	0.492	**0.492**	0.486	0.450
	Method M_1	0.481	0.559	0.551	**0.594**	**0.534**	0.520	0.488	0.484	0.482	0.450

Table 6.2a. Micro-averaged F-measure results for Hitech and LA1 datasets

MICRO_F	# words	10	30	50	100	200	500	1000	1500	2000	all
Hitech	tfidf(l)	0.551	0.596	0.613	0.627	0.624	0.644	0.621	0.618	0.627	0.649
	tfidf(g)	0.372	0.518	0.538	0.603	0.606	0.623	0.643	0.647	0.659	0.649
	IG(l)	0.510	0.610	0.617	0.638	0.630	0.654	0.644	0.634	0.638	0.649
	IG(g)	0.430	0.523	0.559	0.621	0.641	0.645	0.649	0.658	0.666	0.649
	CHI(l)	0.557	0.590	0.620	0.631	0.636	0.636	0.619	0.630	0.632	0.649
	CHI(g)	0.485	0.559	0.597	0.621	0.637	0.633	0.651	**0.670**	0.667	0.649
	Acc2(l)	0.558	0.612	0.636	0.649	0.637	0.651	0.659	0.647	0.646	0.649
	Acc2(g)	0.521	0.581	0.575	0.606	0.607	0.657	0.642	0.637	0.661	0.649
	DF(l)	0.501	0.550	0.578	0.624	0.613	0.622	0.644	0.664	0.661	0.649
	DF(g)	0.214	0.546	0.538	0.583	0.616	0.609	0.624	0.624	0.629	0.649
	Method M_4	**0.573**	**0.625**	0.637	0.658	0.657	**0.656**	**0.666**	0.661	**0.673**	0.649
	Method M_3	0.555	0.610	0.637	0.638	0.650	0.652	0.652	0.659	0.653	0.649
	Method M_2	0.547	0.617	**0.638**	0.645	0.635	0.645	0.655	0.646	0.645	0.649
	Method M_1	0.571	0.623	0.630	**0.657**	**0.661**	0.648	0.655	0.633	0.629	0.649
LA1	tfidf(l)	0.631	0.731	0.761	0.785	0.789	0.807	0.814	0.812	0.815	0.841
	tfidf(g)	0.465	0.648	0.722	0.767	0.793	0.816	0.817	0.825	0.833	0.841
	IG(l)	0.660	0.739	0.765	0.793	0.807	0.830	0.831	0.831	0.833	0.841
	IG(g)	0.388	0.664	0.724	0.769	0.804	0.828	0.829	0.833	0.838	0.841
	CHI(l)	**0.671**	0.736	0.761	0.788	0.813	0.823	0.833	**0.840**	0.838	0.841
	CHI(g)	0.340	0.635	0.663	0.745	0.789	0.822	0.828	0.824	0.838	0.841
	Acc2(l)	0.659	0.742	0.764	0.802	**0.817**	0.829	0.835	**0.840**	0.840	0.841
	Acc2(g)	0.478	0.687	0.758	0.789	0.812	0.831	0.829	0.830	0.829	0.841
	DF(l)	0.318	0.688	0.740	0.766	0.782	0.814	0.815	0.827	0.826	0.841
	DF(g)	0.103	0.397	0.642	0.709	0.762	0.799	0.821	0.832	0.827	0.841
	Method M_4	0.669	0.735	**0.772**	0.800	0.811	0.827	**0.836**	0.833	0.836	0.841
	Method M_3	0.608	**0.743**	0.761	0.798	0.814	0.826	0.835	0.835	0.835	0.841
	Method M_2	0.554	0.730	0.760	**0.804**	0.813	0.831	0.833	0.837	**0.841**	0.841
	Method M_1	0.669	0.735	**0.772**	0.800	0.813	**0.832**	0.830	0.833	**0.841**	0.841

Table 6.2b. Macro-averaged F-measure results for Hitech and LA1 datasets

MACRO F	# words	10	30	50	100	200	500	1000	1500	2000	all
Hitech	tfidf(l)	**0.486**	**0.555**	**0.571**	0.571	0.564	0.589	0.567	0.549	0.561	0.558
	tfidf(g)	0.228	0.371	0.465	0.507	0.505	0.530	0.538	0.582	0.598	0.558
	IG(l)	0.456	0.529	0.539	0.577	0.571	0.591	0.573	0.555	0.557	0.558
	IG(g)	0.301	0.433	0.461	0.538	0.558	0.572	0.597	0.601	0.602	0.558
	CHI(l)	0.477	0.495	0.536	0.572	0.567	0.551	0.545	0.552	0.567	0.558
	CHI(g)	0.340	0.437	0.509	0.528	0.550	0.570	0.610	**0.611**	0.605	0.558
	Acc2(l)	0.459	0.522	0.550	0.571	0.564	**0.596**	0.600	0.583	0.593	0.558
	Acc2(g)	0.433	0.507	0.496	0.521	0.522	0.567	0.582	0.567	0.603	0.558
	DF(l)	0.397	0.485	0.507	0.549	0.540	0.549	0.592	**0.611**	0.603	0.558
	DF(g)	0.141	0.389	0.383	0.461	0.527	0.510	0.524	0.526	0.532	0.558
	Method M_4	0.482	0.553	0.578	0.582	0.572	0.590	**0.615**	0.609	**0.615**	0.558
	Method M_3	0.443	0.533	0.563	0.559	0.566	0.585	0.581	0.597	0.586	0.558
	Method M_2	0.449	0.527	0.546	0.568	0.555	0.594	0.597	0.578	0.588	0.558
	Method M_1	0.472	0.542	0.556	**0.594**	**0.596**	0.578	0.600	0.570	0.561	0.558
LA1	tfidf(l)	0.552	0.674	0.706	0.728	0.735	0.755	0.762	0.756	0.764	0.777
	tfidf(g)	0.284	0.528	0.628	0.692	0.715	0.752	0.748	0.753	0.765	0.777
	IG(l)	0.578	**0.688**	0.714	0.743	0.756	**0.781**	**0.779**	0.775	0.777	0.777
	IG(g)	0.301	0.510	0.603	0.658	0.745	0.771	0.764	0.762	0.772	0.777
	CHI(l)	**0.607**	0.686	**0.715**	0.741	0.766	0.772	0.778	**0.788**	**0.785**	0.777
	CHI(g)	0.318	0.523	0.549	0.651	0.722	0.762	0.765	0.757	0.775	0.777
	Acc2(l)	0.546	0.682	0.712	**0.759**	**0.773**	0.770	0.774	0.776	0.781	0.777
	Acc2(g)	0.387	0.584	0.677	0.732	0.754	0.769	0.758	0.768	0.759	0.777
	DF(l)	0.376	0.582	0.665	0.702	0.715	0.755	0.758	0.767	0.768	0.777
	DF(g)	0.117	0.227	0.515	0.588	0.688	0.724	0.756	0.766	0.760	0.777
	Method M_4	0.590	0.680	**0.715**	0.749	0.749	0.762	0.771	0.764	0.770	0.777
	Method M_3	0.472	0.675	0.698	0.745	0.763	0.762	0.772	0.770	0.767	0.777
	Method M_2	0.455	0.662	0.707	0.750	0.770	0.774	0.771	0.775	0.776	0.777
	Method M_1	0.590	0.680	**0.715**	0.749	0.760	0.773	0.774	0.773	0.782	0.777

Table 6.3a. Micro-averaged F-measure results for Reviews and Reuters datasets

MICRO_F	# words	10	30	50	100	200	500	1000	1500	2000	all
Reviews	tfidf(l)	0.842	0.865	0.889	0.900	0.906	0.918	0.926	0.924	0.920	0.941
	tfidf(g)	0.790	0.869	0.869	0.894	0.935	**0.944**	**0.943**	0.937	0.936	0.941
	IG(l)	0.850	0.884	0.900	0.909	0.926	0.930	0.936	0.940	**0.942**	0.941
	IG(g)	0.816	**0.897**	0.904	0.909	**0.937**	0.943	0.940	0.940	0.941	0.941
	CHI(l)	0.828	0.877	0.901	0.907	0.919	0.921	0.928	0.933	0.933	0.941
	CHI(g)	0.736	0.896	**0.912**	**0.923**	0.930	0.940	0.941	**0.944**	0.940	0.941
	Acc2(l)	0.842	0.900	0.905	0.917	0.927	0.938	0.942	0.940	0.938	0.941
	Acc2(g)	0.829	0.899	0.921	0.919	0.930	0.940	0.941	0.944	0.944	0.941
	DF(l)	0.805	0.852	0.868	0.895	0.906	0.918	0.928	0.930	0.933	0.941
	DF(g)	0.468	0.791	0.813	0.852	0.897	0.930	0.935	0.933	0.939	0.941
	Method M_4	0.844	0.888	0.902	0.921	0.922	0.936	0.940	0.942	0.940	0.941
	Method M_3	**0.869**	0.895	0.903	0.914	0.924	0.940	0.941	**0.944**	0.940	0.941
	Method M_2	0.850	0.902	0.903	0.916	0.923	0.940	0.942	0.940	0.937	0.941
	Method M_1	0.844	0.888	0.902	0.921	0.922	0.934	0.938	0.939	0.936	0.941
Reuters	tfidf(l)	0.776	0.812	0.831	0.835	0.838	0.845	0.853	0.850	0.855	0.855
	tfidf(g)	0.367	0.565	0.625	0.694	0.760	0.811	0.843	0.858	0.860	0.855
	IG(l)	**0.777**	0.820	0.838	0.842	0.845	0.850	0.856	0.858	0.856	0.855
	IG(g)	0.485	0.661	0.705	0.765	0.815	0.849	0.857	**0.862**	0.861	0.855
	CHI(l)	0.520	**0.823**	**0.840**	0.842	0.839	0.845	0.852	0.855	0.854	0.855
	CHI(g)	0.231	0.367	0.531	0.626	0.742	0.798	0.844	0.856	**0.862**	0.855
	Acc2(l)	0.773	0.811	0.835	0.846	0.855	0.860	**0.862**	0.859	0.859	0.855
	Acc2(g)	0.352	0.388	0.513	0.622	0.725	0.814	0.832	0.848	0.860	0.855
	DF(l)	0.725	0.802	0.820	0.841	0.847	0.854	0.859	0.859	0.859	0.855
	DF(g)	0.412	0.542	0.624	0.679	0.753	0.802	0.839	0.854	0.857	0.855
	Method M_4	0.773	0.815	0.823	0.852	**0.860**	0.861	0.857	0.859	0.861	0.855
	Method M_3	0.690	0.803	0.819	0.846	0.856	**0.863**	0.861	0.861	0.860	0.855
	Method M_2	0.762	0.815	0.828	0.847	0.858	0.861	0.861	0.859	0.860	0.855
	Method M_1	0.773	0.817	0.835	**0.854**	0.857	0.858	0.861	**0.862**	**0.862**	0.855

Table 6.3b. Macro-averaged F-measure results for Reviews and Reuters datasets

MACRO F	# words	10	30	50	100	200	500	1000	1500	2000	all
Reviews	tfidf(l)	0.847	0.863	0.890	0.904	0.904	0.916	0.916	0.912	0.906	0.928
	tfidf(g)	0.567	0.697	0.693	0.720	**0.931**	**0.939**	0.939	0.935	0.932	0.928
	IG(l)	0.860	0.892	0.886	0.908	0.928	0.928	0.930	0.934	0.936	0.928
	IG(g)	0.655	0.871	0.867	0.899	0.933	0.938	0.937	0.935	**0.939**	0.928
	CHI(l)	0.841	0.881	0.886	0.905	0.916	0.916	0.919	0.923	0.926	0.928
	CHI(g)	0.664	**0.905**	**0.915**	**0.923**	0.928	0.937	0.933	0.937	0.935	0.928
	Acc2(l)	0.847	0.901	0.905	0.919	0.930	0.935	**0.940**	0.939	0.935	0.928
	Acc2(g)	0.840	0.908	0.923	0.922	0.931	0.939	0.939	0.942	0.941	0.928
	DF(l)	0.819	0.869	0.880	0.905	0.909	0.920	0.929	0.931	0.933	0.928
	DF(g)	0.295	0.567	0.605	0.678	0.731	0.927	0.935	0.934	0.936	0.928
	Method M_4	0.847	0.888	0.901	0.920	0.921	0.932	**0.940**	0.939	0.936	0.928
	Method M_3	**0.866**	0.892	0.907	0.918	0.925	0.935	0.938	**0.940**	0.935	0.928
	Method M_2	0.854	0.904	0.903	0.915	0.924	0.937	**0.940**	0.936	0.934	0.928
	Method M_1	0.847	0.888	0.901	0.922	0.925	0.933	0.935	0.935	0.929	0.928
Reuters	tfidf(l)	0.494	0.512	0.519	0.508	0.514	0.493	0.495	0.491	0.492	0.438
	tfidf(g)	0.014	0.031	0.044	0.090	0.163	0.262	0.370	0.417	0.432	0.438
	IG(l)	0.494	0.530	0.512	0.517	0.496	0.495	0.493	**0.496**	0.490	0.438
	IG(g)	0.034	0.099	0.140	0.195	0.321	0.392	0.457	0.490	0.476	0.438
	CHI(l)	0.466	0.491	0.493	0.500	0.488	0.493	0.493	0.494	0.491	0.438
	CHI(g)	0.051	0.107	0.163	0.242	0.377	0.439	0.476	0.475	0.482	0.438
	Acc2(l)	0.492	0.525	0.524	0.527	0.515	**0.513**	**0.500**	0.492	0.489	0.438
	Acc2(g)	0.039	0.113	0.145	0.215	0.193	0.484	0.488	0.492	0.490	0.438
	DF(l)	0.463	0.497	0.515	**0.539**	**0.532**	0.511	**0.500**	0.491	0.493	0.438
	DF(g)	0.010	0.034	0.058	0.090	0.147	0.243	0.364	0.411	0.438	0.438
	Method M_4	0.484	0.485	0.477	0.491	0.499	0.491	0.472	0.486	0.478	0.438
	Method M_3	0.302	0.459	0.495	0.506	0.507	0.506	0.498	0.492	0.489	0.438
	Method M_2	0.470	**0.531**	0.519	0.529	0.518	**0.513**	0.499	0.493	0.489	0.438
	Method M_1	**0.507**	0.512	**0.531**	0.529	0.513	0.505	0.496	**0.495**	**0.494**	0.438

Table 6.4. Micro and Macro-averaged F-measure results for RCV1

MICRO F	# words	10	30	50	100	200	500	1000	1500	2000	all
RCV1	tfidf(l)	0.329	0.463	0.525	0.597	0.659	0.730	0.762	0.776	0.783	**0.797**
	tfidf(g)	0.191	0.402	0.465	0.551	0.660	0.736	0.775	0.783	0.788	0.797
	IG(l)	0.338	0.416	0.493	0.573	0.649	0.728	0.756	0.765	0.770	0.797
	IG(g)	0.274	0.411	0.476	0.578	0.669	0.745	0.778	0.785	0.784	0.797
	CHI(l)	0.364	0.452	0.531	0.599	0.676	0.735	0.762	0.772	0.776	0.797
	CHI(g)	0.192	0.405	0.457	0.550	0.661	0.726	0.768	0.782	0.787	0.797
	DF(l)	**0.444**	**0.568**	**0.629**	**0.693**	**0.738**	**0.775**	**0.788**	**0.792**	**0.792**	0.797
	DF(g)	0.281	0.423	0.466	0.557	0.658	0.737	0.772	0.784	0.788	0.797

MACRO F	# words	10	30	50	100	200	500	1000	1500	2000	all
RCV1	tfidf(l)	0.045	0.082	0.125	0.186	0.262	0.363	0.424	0.460	0.470	0.479
	tfidf(g)	0.013	0.054	0.077	0.145	0.284	0.389	0.460	0.471	0.484	0.479
	IG(l)	0.096	0.197	0.267	0.345	0.406	0.465	0.478	0.483	0.490	0.479
	IG(g)	0.018	0.056	0.079	0.170	0.293	0.411	0.466	0.475	0.486	0.479
	CHI(l)	0.139	0.247	0.308	0.365	0.434	0.469	0.486	0.490	0.492	0.479
	CHI(g)	0.013	0.054	0.071	0.150	0.280	0.387	0.457	0.479	0.490	0.479
	DF(l)	**0.225**	**0.347**	**0.400**	**0.464**	**0.506**	**0.533**	**0.533**	**0.531**	**0.530**	0.479
	DF(g)	0.019	0.059	0.075	0.151	0.278	0.391	0.458	0.477	0.486	0.479

6.2. Analysis of the Proposed Methods

The success of local policy at low number of keywords motivated us to concentrate on local feature selection methods. Therefore all of the methods that are proposed in this thesis are local methods. In fact, other methods are modified versions of M_1 Method. But we include all of them since each is better in different cases. For example, M_3 Method is best in Wap dataset while M_4 Method is the best method for Hitech dataset.

When we look at Table 7.1a-7.3b, we see that newly proposed methods are as good as existing methods in homogenous datasets such as Classic3, LA1 and Reviews while they are better than existing ones when the dataset is skew or has low success rates with existing

methods. This situation makes them suitable for using in place of previous methods because we will know that the results will be as good as the previous methods regardless of the dataset. Especially, they are very good at Wap and Hitech datasets. In these datasets, they are better than existing methods in terms of both Micro and Macro-averaged F-measures.

In Wap dataset (see Figure 7.1 and 7.2), M_1 Method reaches 76.7% Micro and 59.4% Macro-averaged F-measures while IG can reach at most 75.8% Micro and 54.8% Macro-averaged F-measures with local policy. In addition, IG method reaches its highest values at different number of keywords while 100 keywords is an optimal choice for M_1 Method for both Micro and Macro-averaged F-measures. The exceptional success of the proposed methods in Wap dataset shows that they are very good at finding the best features even when a class does not have too much training documents.

In Hitech dataset (see Figure 7.3 and 7.4), M_4 Method reaches 67.3% Micro and 61.5% Macro-averaged F-measures while CHI, the best method in this dataset, can achieve at most 67.0% Micro and 61.1% Macro-averaged F-measures. Again, the gap enlarges when the number of keywords is decreased.

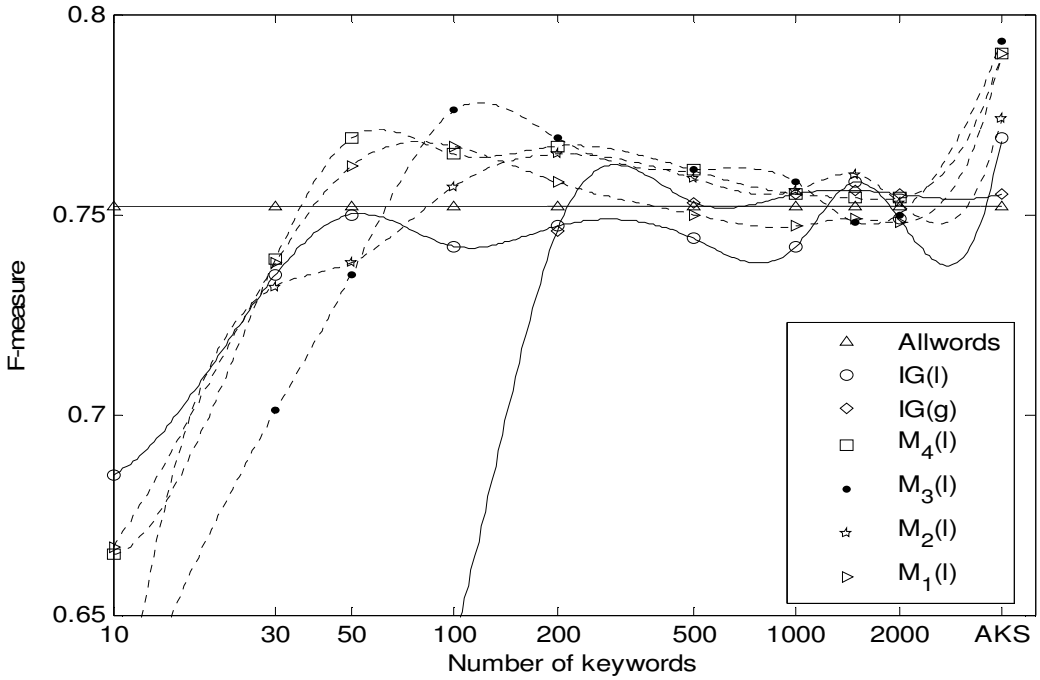


Figure 6.1. Micro-averaged F-measure results for Wap dataset for new methods

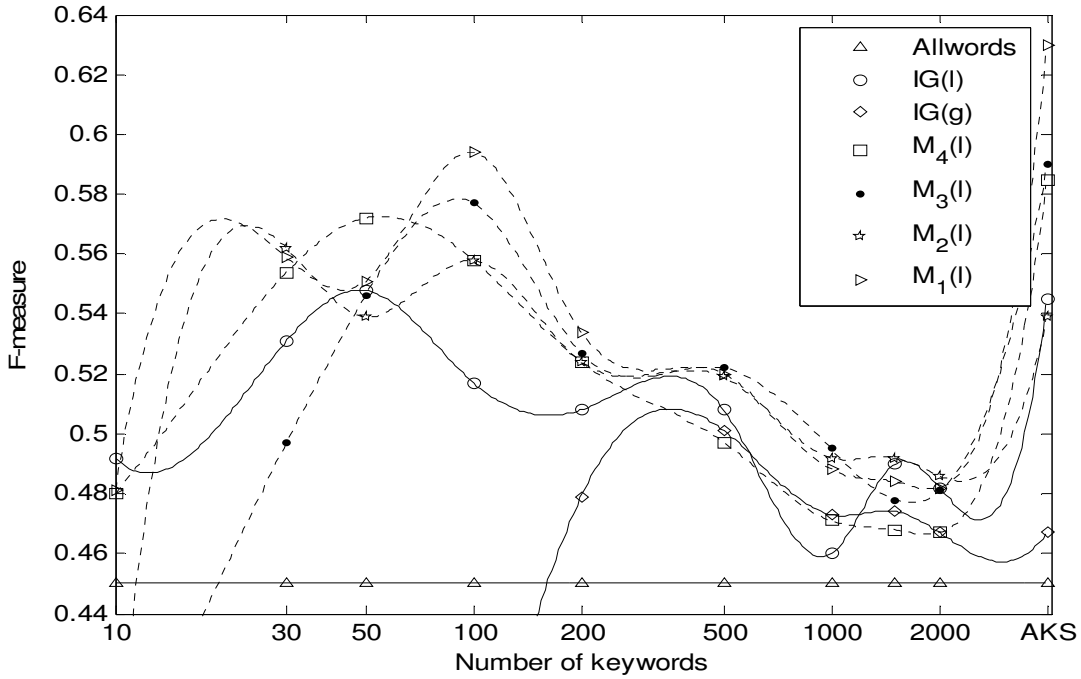


Figure 6.2. Macro-averaged F-measure results for Wap dataset for new methods

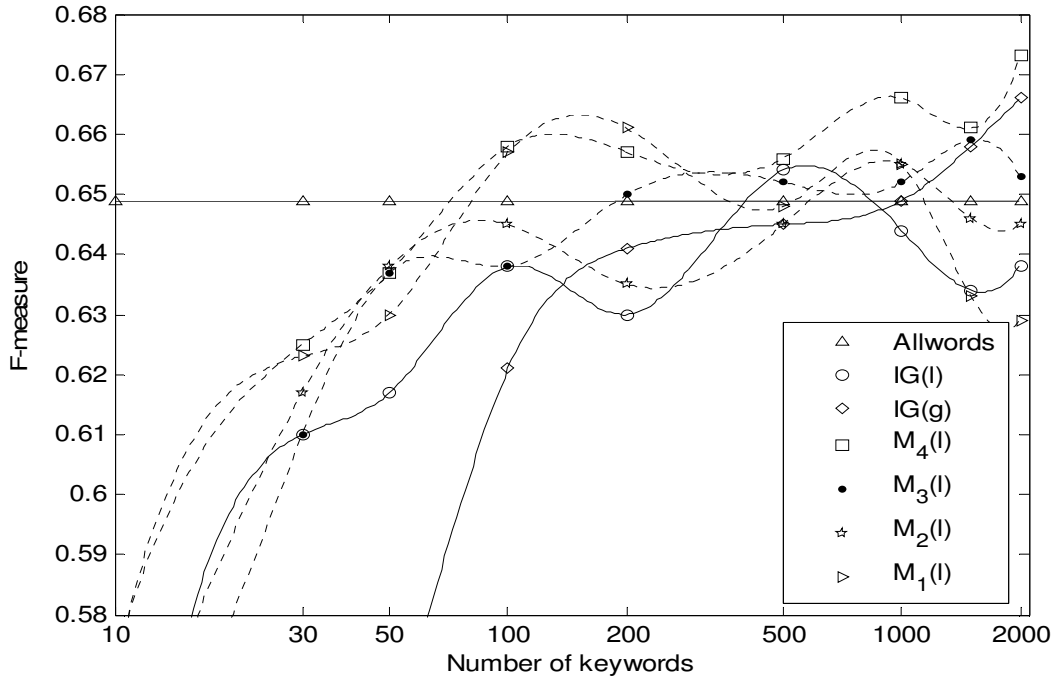


Figure 6.3. Micro-averaged F-measure results for Hitech dataset for new methods

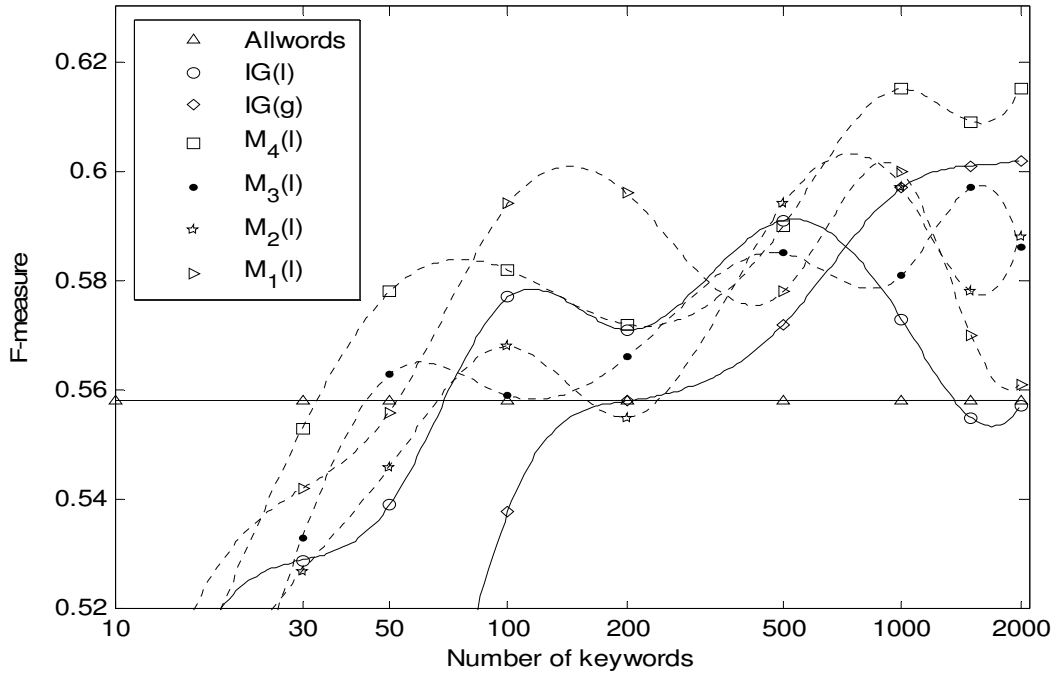


Figure 6.4. Macro-averaged F-measure results for Hitech dataset for new methods

Another remarkable property of the new methods is that they reach their maximum values or at least give satisfactory results about 100 keywords. This may be explained by the fact that all of them are based on local policy. Nevertheless, in Hitech dataset we see that M_4 Method preserves its high success when the number of keywords is increased from 100 to 2000 keywords despite the other 3 methods are not as good as it when the keyword number is high. This situation is expected since M_4 Method uses some of the keywords found by IG (g) method. Therefore it is affected by the success of IG (g) at high number of keywords.

In Table 7.5 and 7.6, we see the results of Adaptive Keyword Selection (AKS) on Wap and Reuters datasets. We have not carried out experiments on other datasets since it is reasonable only if the dataset is skew. This method was inspired from the observation that classes which have only a small number of documents are better represented by a small number of keywords while classes which have a high number of documents are better represented by a large number of keywords. This method uses different number of keywords in different classes which are proportional to the number of documents in the class.

Table 6.5: Micro-averaged F-measure results for Adaptive Keyword Selection (AKS)

MICRO_F	# of words	10	30	50	100	200	500	1000	1500	2000	all	AKS
Wap	tfidf(l)	0.671	0.737	0.741	0.738	0.735	0.722	0.746	0.741	0.749	0.752	0.734
	tfidf(g)	0.134	0.496	0.587	0.655	0.691	0.721	0.740	0.749	0.743	0.752	-
	IG(l)	0.685	0.735	0.750	0.742	0.747	0.744	0.742	0.758	0.749	0.752	0.769
	IG(g)	0.399	0.526	0.577	0.644	0.746	0.753	0.755	0.756	0.755	0.752	-
	CHI(l)	0.440	0.714	0.732	0.732	0.720	0.736	0.742	0.756	0.758	0.752	0.751
	CHI(g)	0.242	0.523	0.540	0.607	0.631	0.712	0.730	0.741	0.749	0.752	-
	Acc2(l)	0.639	0.728	0.757	0.770	0.758	0.755	0.752	0.758	0.752	0.752	0.795
	Acc2(g)	0.221	0.476	0.529	0.629	0.697	0.730	0.743	0.753	0.758	0.752	-
	DF(l)	0.000	0.567	0.704	0.751	0.771	0.747	0.760	0.747	0.747	0.752	0.777
	DF(g)	0.000	0.341	0.395	0.543	0.657	0.723	0.756	0.757	0.758	0.752	-
	Method M_4	0.665	0.739	0.769	0.765	0.767	0.761	0.755	0.754	0.754	0.752	0.790
	Method M_3	0.610	0.701	0.735	0.776	0.769	0.761	0.758	0.748	0.750	0.752	0.793
	Method M_2	0.603	0.732	0.738	0.757	0.765	0.759	0.756	0.760	0.753	0.752	0.774
	Method M_1	0.667	0.738	0.762	0.767	0.758	0.750	0.747	0.749	0.748	0.752	0.790
Reuters	tfidf(l)	0.776	0.812	0.831	0.835	0.838	0.845	0.853	0.850	0.855	0.855	0.850
	tfidf(g)	0.367	0.565	0.625	0.694	0.760	0.811	0.843	0.858	0.860	0.855	-
	IG(l)	0.777	0.820	0.838	0.842	0.845	0.850	0.856	0.858	0.856	0.855	0.858
	IG(g)	0.485	0.661	0.705	0.765	0.815	0.849	0.857	0.862	0.861	0.855	-
	CHI(l)	0.520	0.823	0.840	0.842	0.839	0.845	0.852	0.855	0.854	0.855	0.853
	CHI(g)	0.231	0.367	0.531	0.626	0.742	0.798	0.844	0.856	0.862	0.855	-
	Acc2(l)	0.773	0.811	0.835	0.846	0.855	0.860	0.862	0.859	0.859	0.855	0.863
	Acc2(g)	0.352	0.388	0.513	0.622	0.196	0.814	0.832	0.848	0.860	0.855	-
	DF(l)	0.725	0.802	0.820	0.841	0.847	0.854	0.859	0.859	0.859	0.855	0.862
	DF(g)	0.412	0.542	0.624	0.679	0.753	0.802	0.839	0.854	0.857	0.855	-
	Method M_4	0.773	0.815	0.823	0.852	0.860	0.861	0.857	0.859	0.861	0.855	0.861
	Method M_3	0.690	0.803	0.819	0.846	0.856	0.863	0.861	0.861	0.860	0.855	0.862
	Method M_2	0.762	0.815	0.828	0.847	0.858	0.861	0.861	0.859	0.860	0.855	0.864
	Method M_1	0.773	0.817	0.835	0.854	0.857	0.858	0.861	0.862	0.862	0.855	0.866

Table 6.6: Macro-averaged F-measure results for Adaptive Keyword Selection (AKS)

MACRO_F	# of words	10	30	50	100	200	500	1000	1500	2000	all	AKS
Wap	tfidf(l)	0.506	0.593	0.565	0.532	0.507	0.495	0.509	0.477	0.483	0.450	0.543
	tfidf(g)	0.093	0.208	0.306	0.350	0.412	0.442	0.455	0.468	0.455	0.450	-
	IG(l)	0.492	0.531	0.548	0.517	0.508	0.508	0.460	0.490	0.482	0.450	0.545
	IG(g)	0.052	0.185	0.284	0.375	0.479	0.501	0.473	0.474	0.467	0.450	-
	CHI(l)	0.493	0.511	0.520	0.509	0.462	0.491	0.475	0.488	0.491	0.450	0.538
	CHI(g)	0.121	0.239	0.256	0.336	0.375	0.451	0.486	0.469	0.478	0.450	-
	Acc2(l)	0.435	0.554	0.564	0.551	0.509	0.516	0.488	0.491	0.485	0.450	0.574
	Acc2(g)	0.117	0.235	0.278	0.411	0.479	0.489	0.480	0.480	0.492	0.450	-
	DF(l)	0.000	0.353	0.513	0.550	0.538	0.483	0.524	0.483	0.481	0.450	0.587
	DF(g)	0.000	0.053	0.095	0.237	0.326	0.430	0.474	0.470	0.462	0.450	-
	Method M_4	0.480	0.554	0.572	0.558	0.524	0.497	0.471	0.468	0.467	0.450	0.585
	Method M_3	0.376	0.497	0.546	0.577	0.527	0.522	0.495	0.478	0.481	0.450	0.590
	Method M_2	0.386	0.562	0.539	0.558	0.524	0.519	0.492	0.492	0.486	0.450	0.539
	Method M_1	0.481	0.559	0.551	0.594	0.534	0.520	0.488	0.484	0.482	0.450	0.630
Reuters	tfidf(l)	0.494	0.512	0.519	0.508	0.514	0.493	0.495	0.491	0.492	0.438	0.516
	tfidf(g)	0.014	0.031	0.044	0.090	0.163	0.262	0.370	0.417	0.432	0.438	-
	IG(l)	0.494	0.530	0.512	0.517	0.496	0.495	0.493	0.496	0.490	0.438	0.527
	IG(g)	0.034	0.099	0.140	0.195	0.321	0.392	0.457	0.490	0.476	0.438	-
	CHI(l)	0.466	0.491	0.493	0.500	0.488	0.493	0.493	0.494	0.491	0.438	0.497
	CHI(g)	0.051	0.107	0.163	0.242	0.377	0.439	0.476	0.475	0.482	0.438	-
	Acc2(l)	0.492	0.525	0.524	0.527	0.515	0.513	0.500	0.492	0.489	0.438	0.531
	Acc2(g)	0.039	0.113	0.145	0.215	0.193	0.484	0.488	0.492	0.490	0.438	-
	DF(l)	0.463	0.497	0.515	0.539	0.532	0.511	0.500	0.491	0.493	0.438	0.538
	DF(g)	0.010	0.034	0.058	0.090	0.147	0.243	0.364	0.411	0.438	0.438	-
	Method M_4	0.484	0.485	0.477	0.491	0.499	0.491	0.472	0.486	0.478	0.438	0.499
	Method M_3	0.302	0.459	0.495	0.506	0.507	0.506	0.498	0.492	0.489	0.438	0.499
	Method M_2	0.470	0.531	0.519	0.529	0.518	0.513	0.499	0.493	0.489	0.438	0.531
	Method M_1	0.507	0.512	0.531	0.529	0.513	0.505	0.496	0.495	0.494	0.438	0.535

When we look at the tables we see that AKS improves the results of almost all keyword selection metrics in Wap dataset while it improves slightly in Reuters dataset. In Wap dataset, M_1 Method with AKS improves the Micro and Macro-averaged F-measures up to 79.0% and 63.0% respectively which was under 76.0% and 55.0% respectively with IG or CHI. It means that if we manage to find optimal number of keywords for each class, AKS can be very valuable for skew datasets that have a small number of training instances.

In addition to this, it has a high value both in Macro and Micro-averaged F-measures. This is particularly important since no other method has both F-measures very high at the same time. For example if we select IG (l) at 2000 keywords for Reuters dataset, Micro-averaged F-measure is very high (85.6%) but Macro-averaged F-measure is only 49.0%. On the other hand, if we select 100 keywords, Macro-averaged F-measure increases to 51.7% but Micro decreases to 84.2%. When we use Adaptive IG (l) Micro and Macro F-measures are both at their highest (85.8% and 52.7% respectively). This situation is a consequence of its success in classifying both rare and common classes correctly.

7. CONCLUSIONS AND FUTURE WORK

In this study, we have made an extensive study of feature selection metrics in text categorization with Support Vector Machine as the classifier. We have compared some of the well-known feature selection metrics such as IG, CHI, DF-Thresholding and Acc2 by varying the number of selected features from 10 to 2000 and also compared the local and global policy on each metric. For our experiments, we have used many datasets with different skewness, size and hardness.

We have also introduced some new feature selection metrics that are at least as good as the well-known metrics in all datasets. In some datasets such as Wap and Hitech we have seen that they are better than existing metrics. In addition, these new metrics have shown great performances especially at small number of keywords such as 100 keywords. This makes them invaluable especially when the practitioner is constrained to use a small number of keywords.

Another contribution of this study is a new feature selection policy called Adaptive Keyword Selection which selects different number of keywords for classes that have different sizes. It has shown great improvements especially with datasets that have a limited number of training instances.

Future work includes the experiments of the new feature selection metrics with other term weighting approaches such as Supervised Term Weighting and with learning algorithms apart from Support Vector Machines. In addition Adaptive Keyword Selection can be extended to make it capable of adjusting the number of features automatically according to the properties of the dataset used.

APPENDIX A: EXPERIMENTAL PROCEDURE

```
Start of Preprocessing

    Load Training and Test Documents (train-docs.txt, test-docs.txt)
      Initialize topic and data matrices (docTopic, docTerm)

      While(!Is_empty(stoplist.txt))      //Step 1
            Add a word to stoplist[] array
      End_while

      While(!EOF(train-docs.txt))    //Step 2
            if(word is document start)
                  increment document count
                  add document id to document list
               else if(word is topic start)
                  while(word is not topic end)
                        add word to topic list
                        insert '1' to the corresponding point in docTopic matrix
                  endwhile
               else if(word is body start)
                  while(word is not body end)
                        stem the word
                        if(word is not stop word)
                              add word to termlist if it is not in termlist
                              insert '1' to the corresponding point in docTerm matrix
                        endif
                  endwhile
               endif
    endwhile

    write docTerm to "tfmatrix.txt"

    foreach (term in termlist)       //Step 3
            idf(term) = log10(numDocs / count(term))
    foreach(entry in docTerm matrix)    //Step 4
            tfidf_score(entry) = docTerm[entry]*idf[term]
    foreach(entry in docTerm matrix)    //Step 5
               Normalize tfidf(entry) by dividing to the rowsum of tfidf scores

      Write used and created matrices to files     //Step 6
               termList → terms.txt
               topicList → topics.txt
               IDs of documents → train-docIDs.txt
               Topic matrix → train-topic-matrix.txt
```

```
                Data matrix → train-data-matrix.txt
        End of file writing

        Repeat Step(2) through Step(6) for test documents with two exceptions
            Use idf(term) from training data
            Use termList and topicList of the training data
        End of test documents
End of Preprocessing



//********** FEATURE SELECTION **********//

Start of Feature Selection

        Read required data matrices from files     //Step 6
                termList ← terms.txt
                tfMatrix ← tf-train-matrix.txt
                docTopic ← train-topic-matrix.txt
                docTerm ← train-data-matrix.txt
                docTopicTest ← test-topic-matrix.txt
                docTermTest ← test-data-matrix.txt
        End of file reading

        Create termTopicDoc  // termTopicDoc[i][j] = frequency of term i in topic j
        Fill termTopicDoc by using tfMatrix & docTopicMatrix

        Create termFreq // termFreq [i] = frequency of term i in whole training data
        Fill termFreq by using rowSums of termTopicDoc

        Create topicFreq // topicFreq [j] = frequency of topic j in whole training data
        Fill topicFreq by using columnSums of termTopicDoc

        For k=0 to NUM_TERMS
            Scores[k]=formula(term[k])          //formula() is the specific formula
        endfor                                  //used for keyword selection. i.e.IG,CHI,etc..

      For j=0 to NUM_TERMS
            If scores[j] > min(score_of(keywords[]))
                  Add termlist[j] to keywords[]
            Endif
      Endfor

      SelectionSort(keywords[])
      Write “keywords[]” to “terms.txt”

      For i=0 to NUM_DOCS_TRAIN
            For r=0 to NUM_KEYWORDS
```

```
                Write docTerm[i][keywords[r]] to "train_svm.txt"
            Endfor
        Endfor

        For i=0 to NUM_DOCS_TEST
            For r=0 to NUM_KEYWORDS
                Write docTermTest[i][keywords[r]] to "test_svm.txt"
            Endfor
        Endfor

End of Feature Selection



//********** SVM CLASSIFICATION **********//

Start of SVM Classification

        Read training data files "train_svm.txt" and "train-topic-matrix.txt"
        For each class X in dataset
            Train a model using the SVM-Light package
            Write the model to file "model_d1_fold_topic_X.txt"
        Endfor

        Read "test_svm.txt"
        For each class X in dataset
            For each document i in test data
                Check whether i belongs to class X
            Endfor
            Write the output to file "output_d1_fold_topic_X.txt"
        Endfor

      Initialize docTestAssignment[][] to 0
        For each class X in dataset
            For each document i in test data
                Read score[i] from "output_d1_fold_topic_X.txt"
            Endfor
            For each document i in test data
                if(score[i]>0)
                    docTestAssignment[i][X]=1
                endif
            Endfor
        Endfor

        Write the docTestAssignment[][] to file "test-topic-assign.txt"
        Read "test-topic-matrix.txt" into the matrix "docTopicTest"

        For i=0 to NUM_DOCS_TEST
```

```
            For j=0 to NUM_TOPICS
                if(docTopicTest[i][j]=1 && docTestAssignment[i][j]=1)
                    TP[j]++
                Else if(docTopicTest[i][j]=0 && docTestAssignment[i][j]=1)
                    FP[j]++
                Else if(docTopicTest[i][j]=1 && docTestAssignment[i][j]=0)
                    FN[j]++
                endif
            Endfor
        Endfor

        Compute precision and recall from TP, FP and FN
        Compute Micro and Macro-averaged F-Measure from precision and recall
        Write Micro and Macro-averaged F-Measures to files “microf.txt” and “macrof.txt”

End of SVM Classification
```

APPENDIX B: FIGURES OF THE EXISTING FEATURE SELECTION METRICS

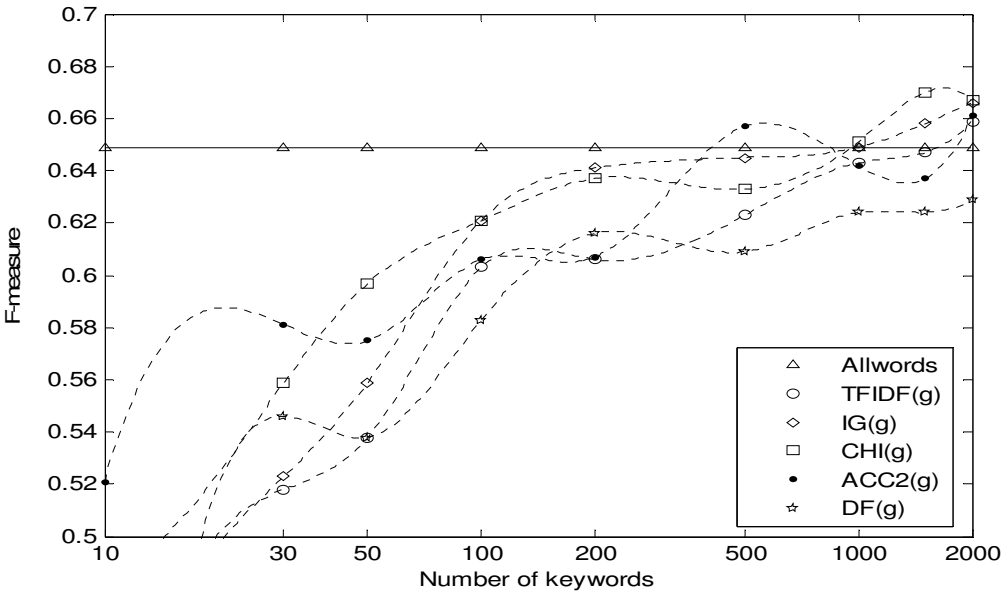


Figure B.1. Micro-averaged F-measure results for Hitech dataset for global policy

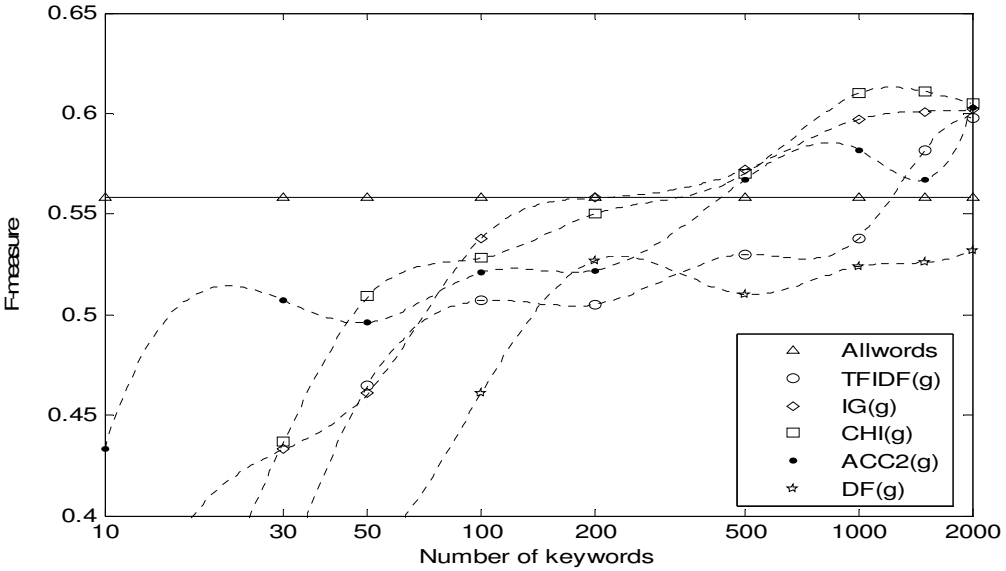


Figure B.2. Macro-averaged F-measure results for Hitech dataset for global policy

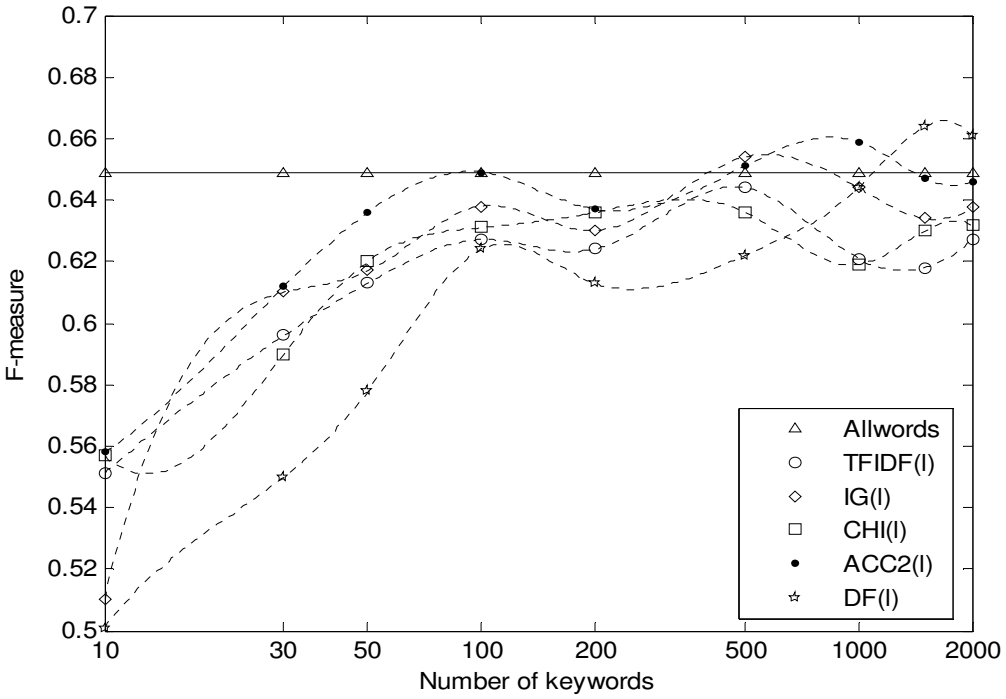


Figure B.3. Micro-averaged F-measure results for Hitech dataset for local policy

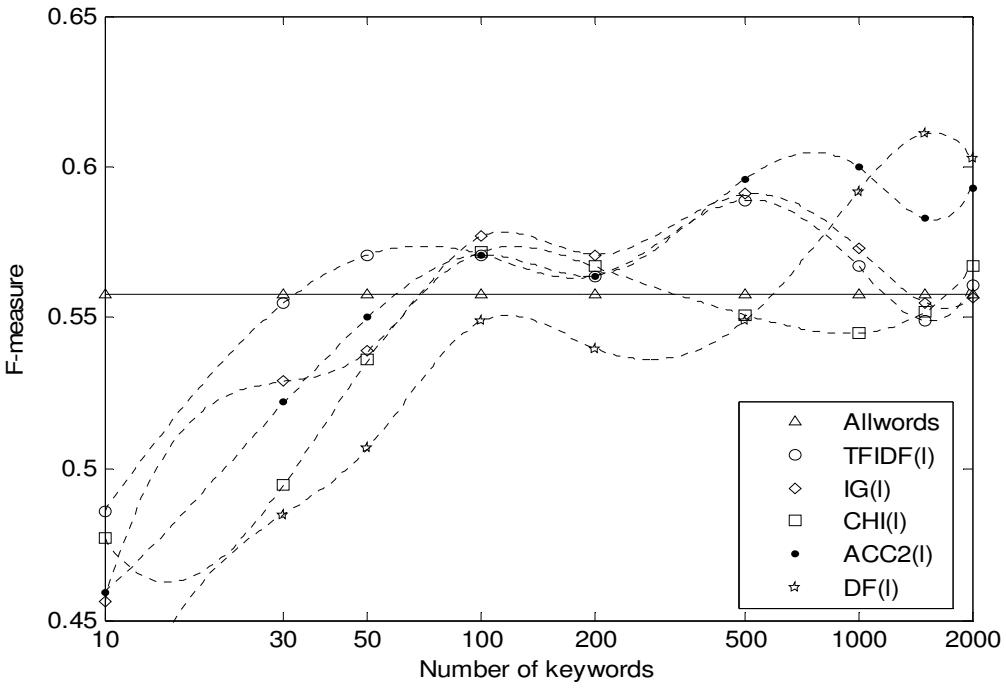


Figure B.4. Macro-averaged F-measure results for Hitech dataset for local policy

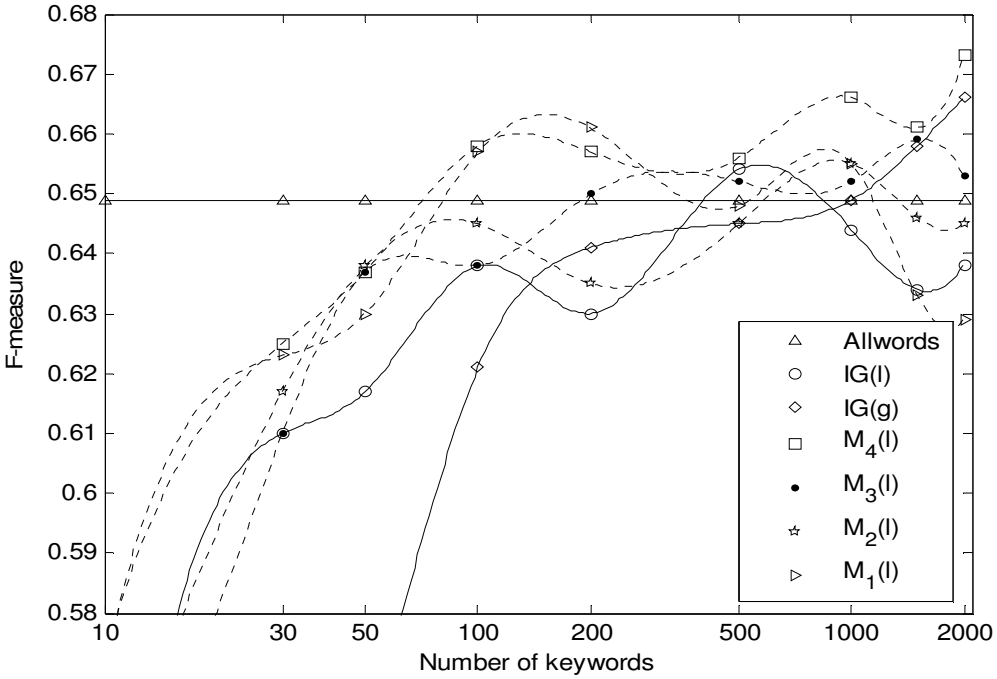


Figure B.5. Micro-averaged F-measure results for Hitech dataset for new methods

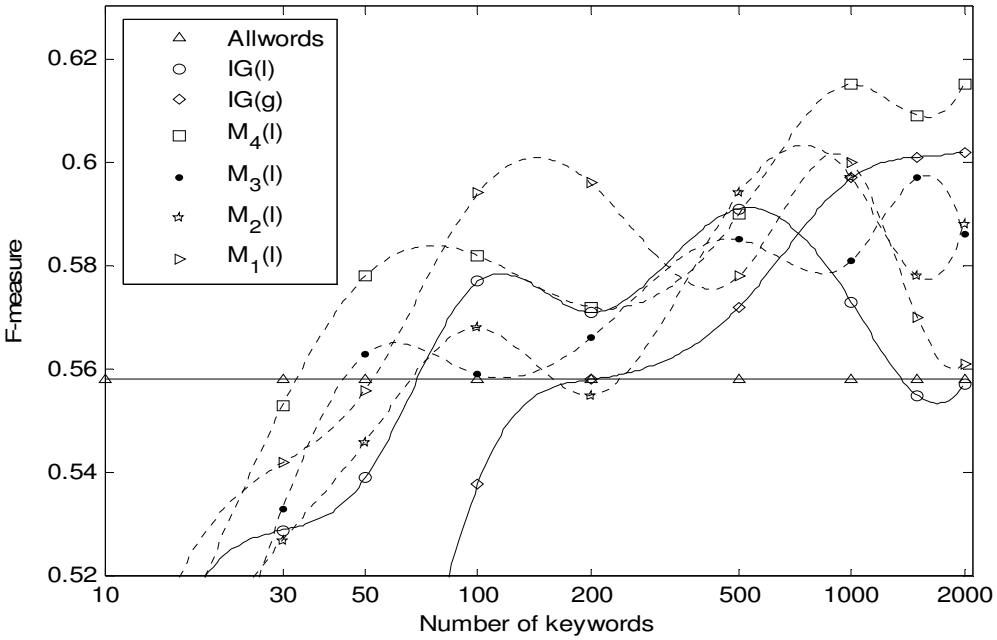


Figure B.6. Macro-averaged F-measure results for Hitech dataset for new methods

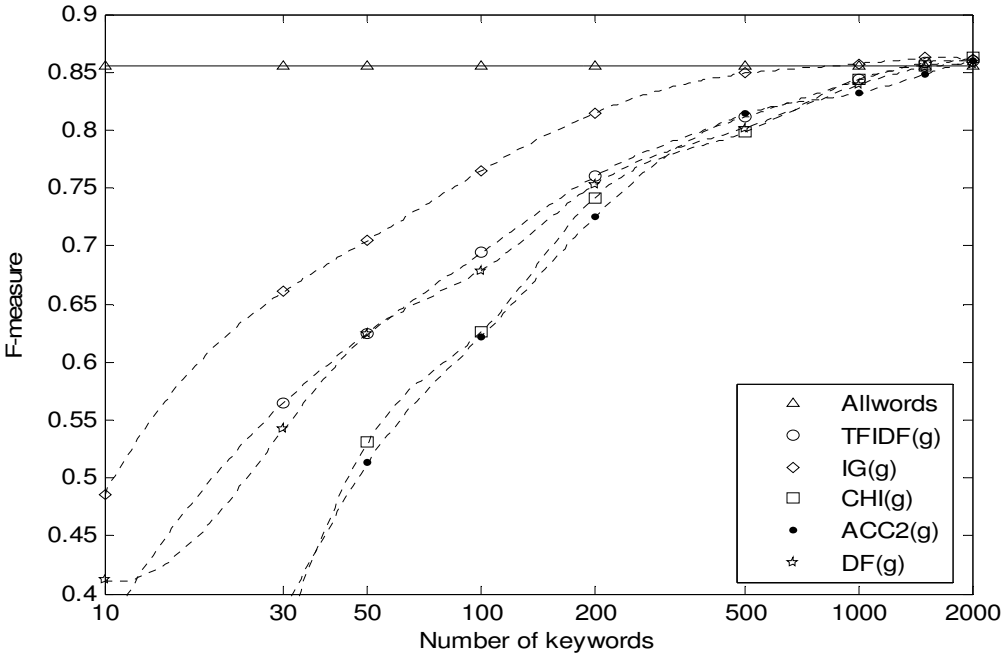


Figure B.7. Micro-averaged F-measure results for Reuters dataset for global policy

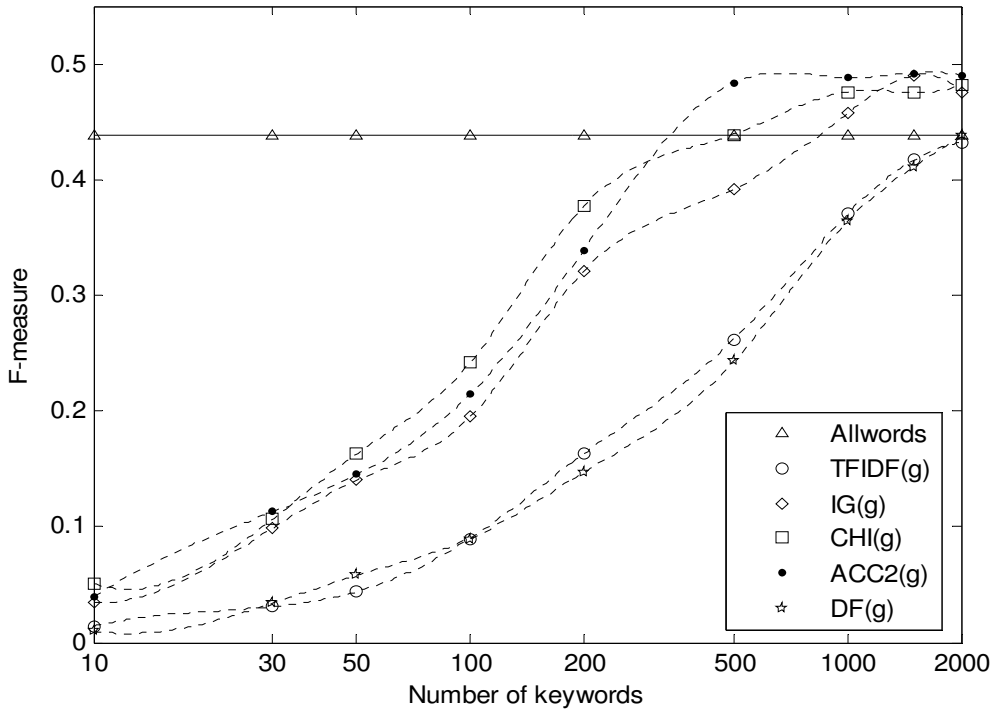


Figure B.8. Macro-averaged F-measure results for Reuters dataset for global policy

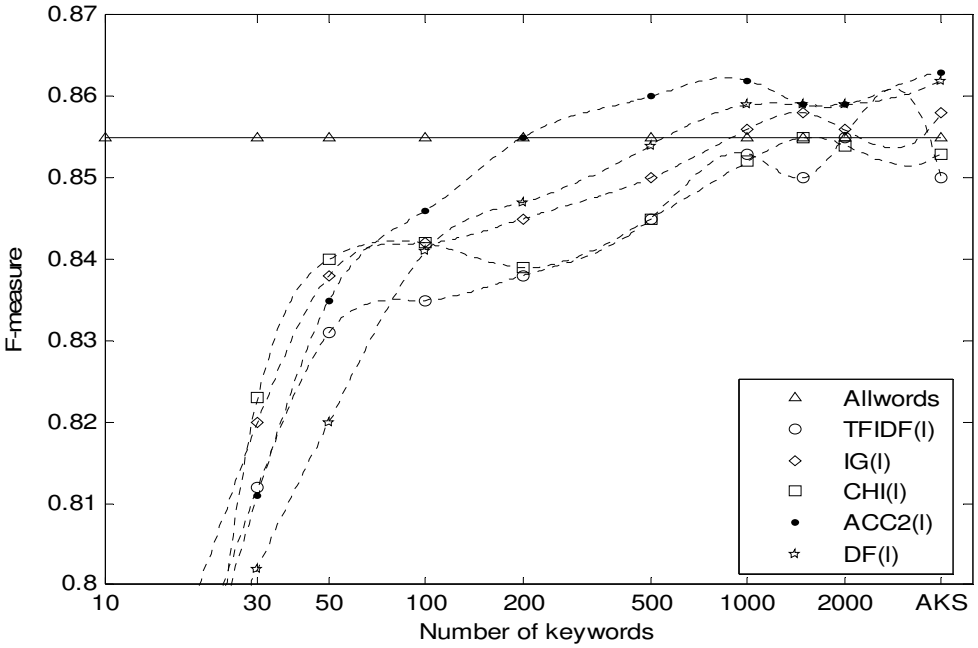


Figure B.9. Micro-averaged F-measure results for Reuters dataset for local policy

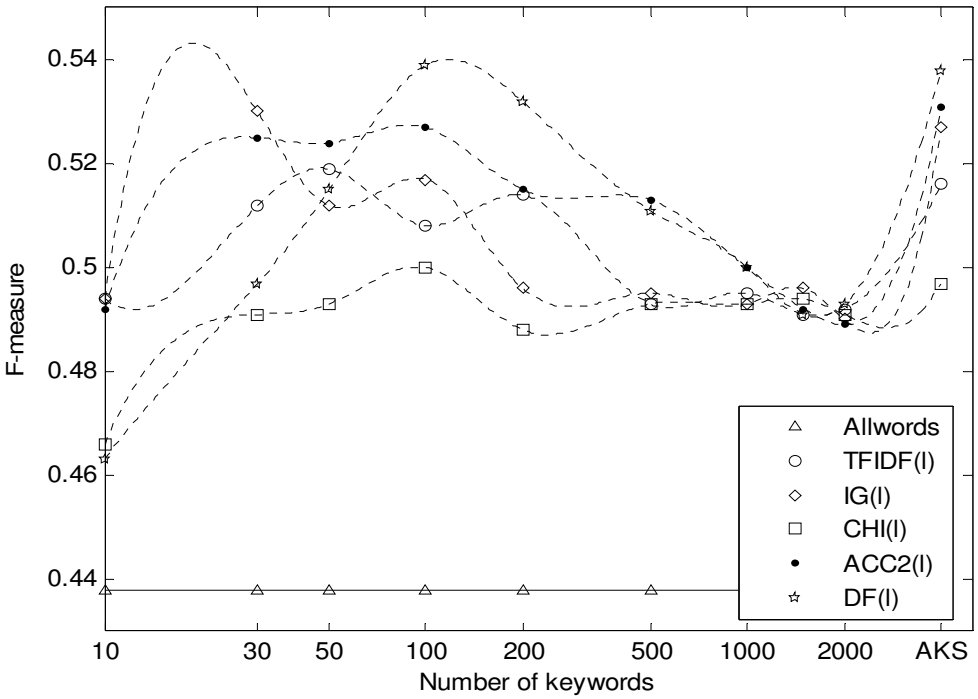


Figure B.10. Macro-averaged F-measure results for Reuters dataset for local policy

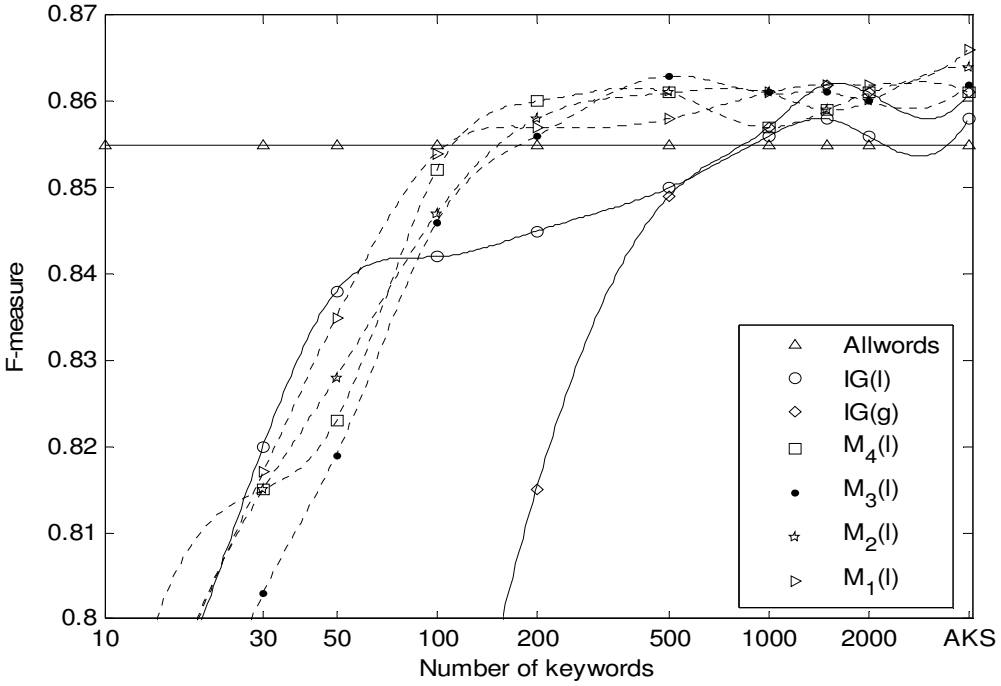


Figure B.11. Micro-averaged F-measure results for Reuters dataset for new methods

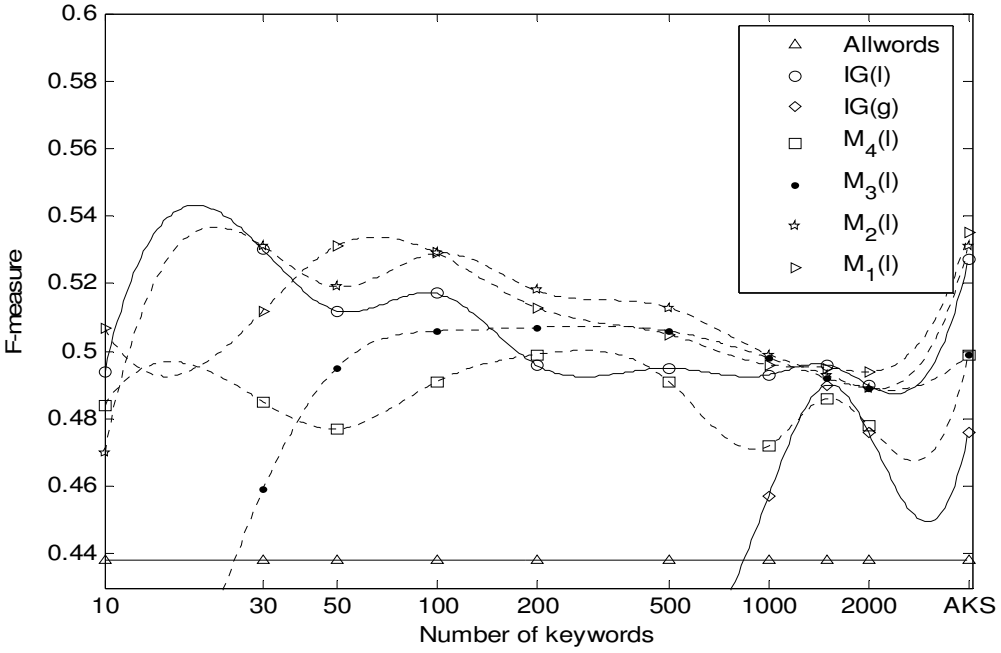


Figure B.12. Macro-averaged F-measure results for Reuters dataset for new methods

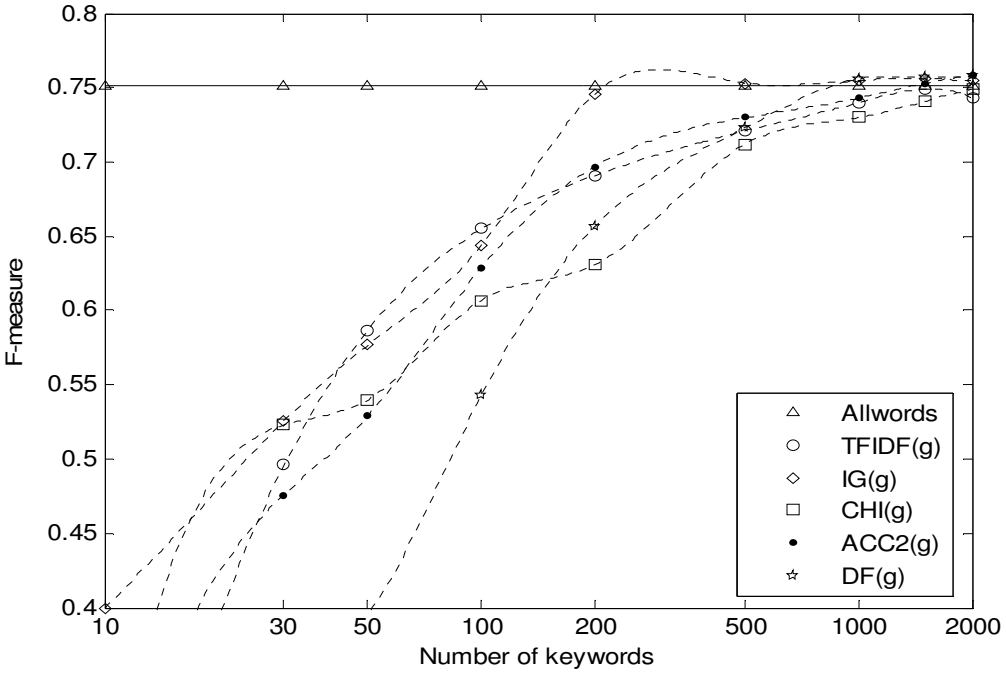


Figure B.13. Micro-averaged F-measure results for Wap Dataset for global policy

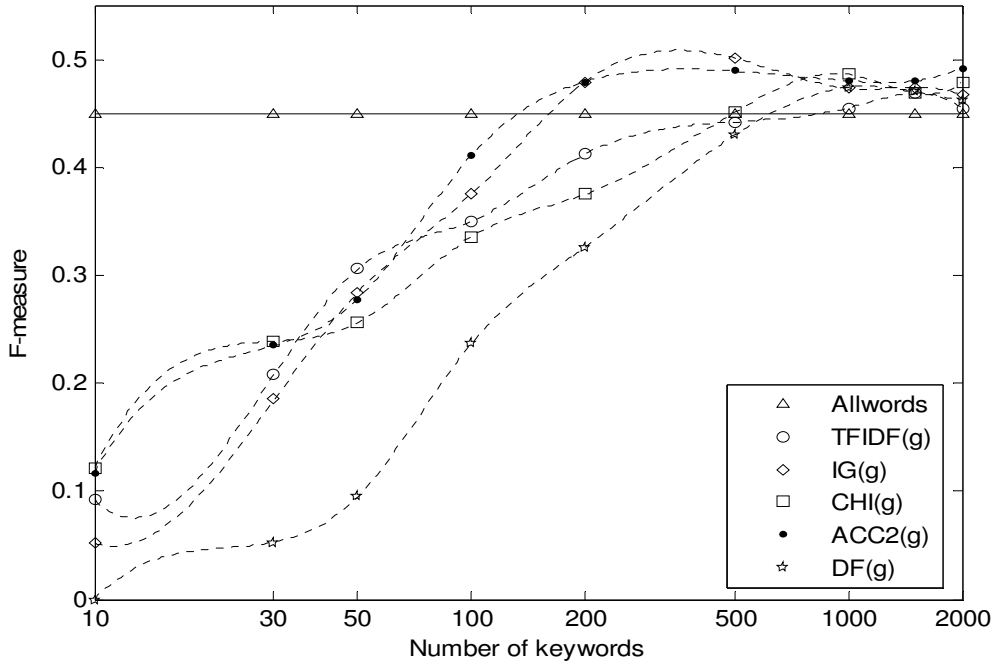


Figure B.14. Macro-averaged F-measure results for Wap dataset for global policy

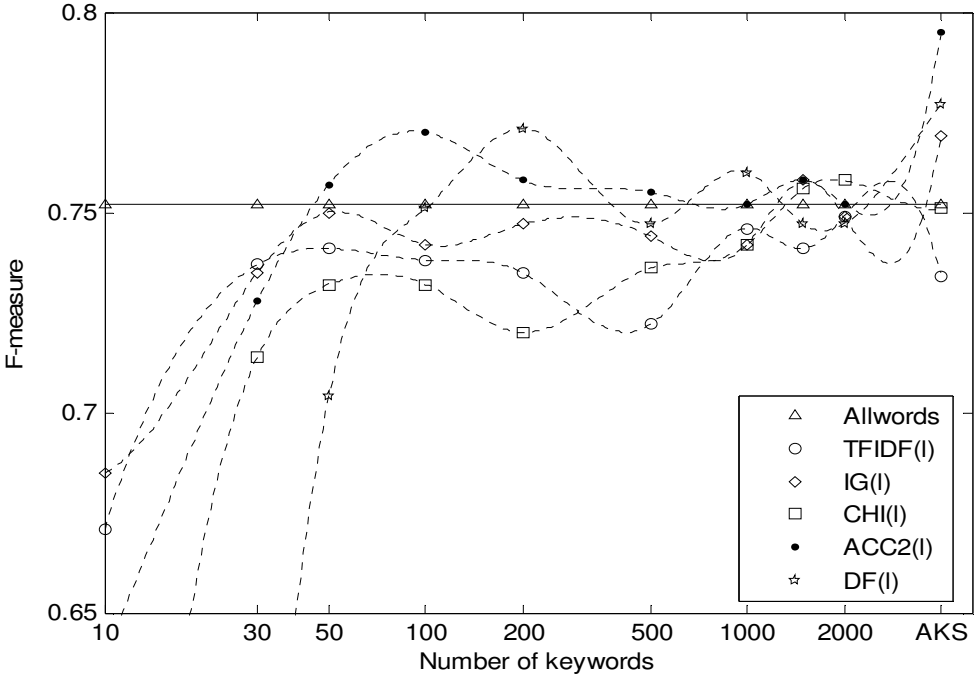


Figure B.15. Micro-averaged F-measure results for Wap dataset for local policy

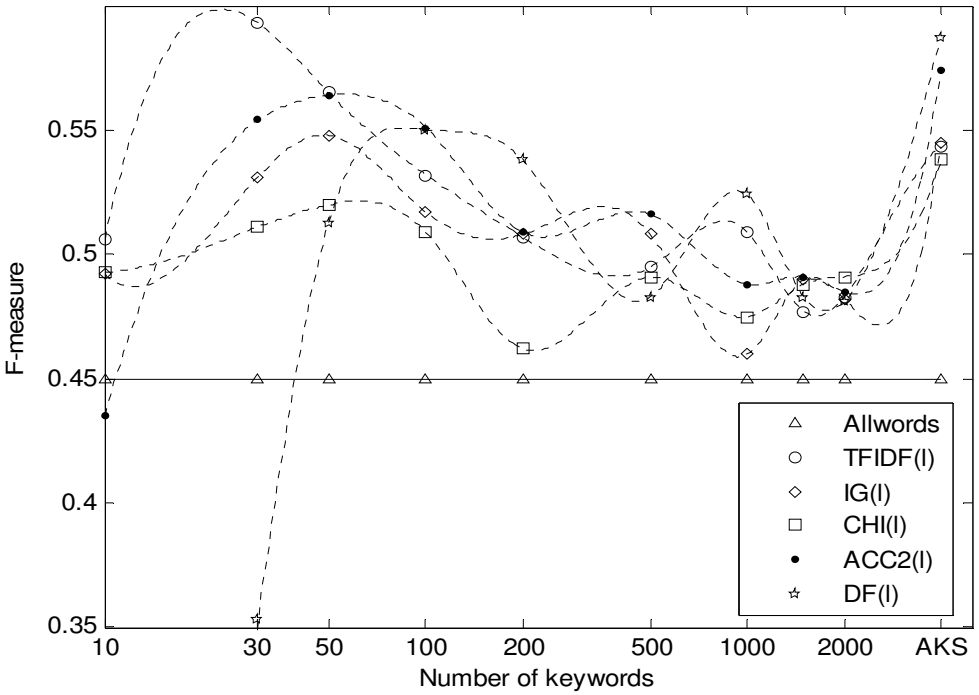


Figure B.16. Macro-averaged F-measure results for Wap dataset for local policy

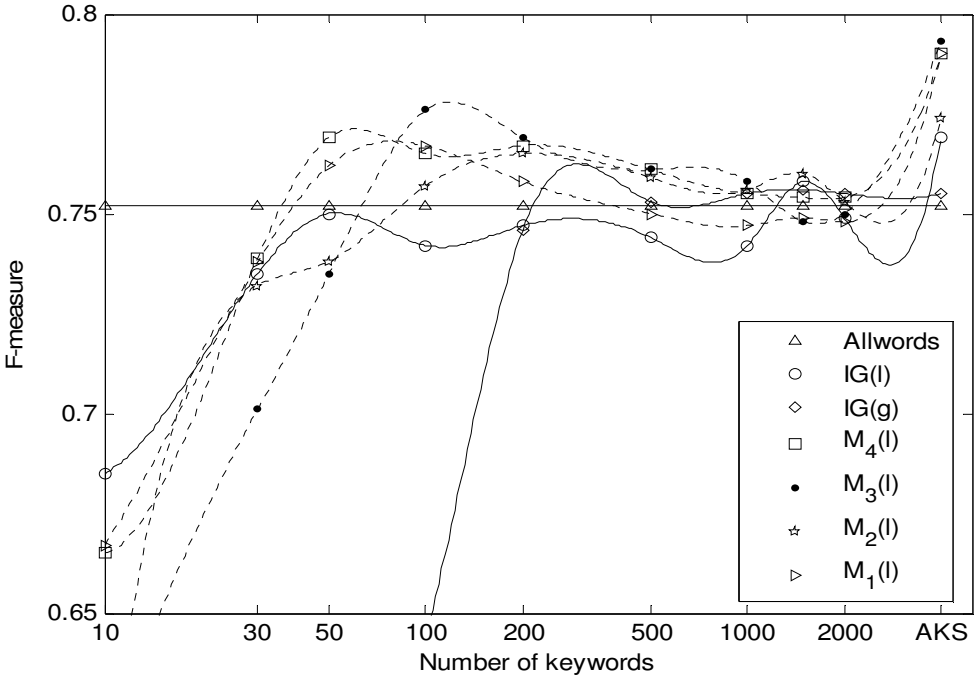


Figure B.17 Micro-averaged F-measure results for Wap dataset for new methods

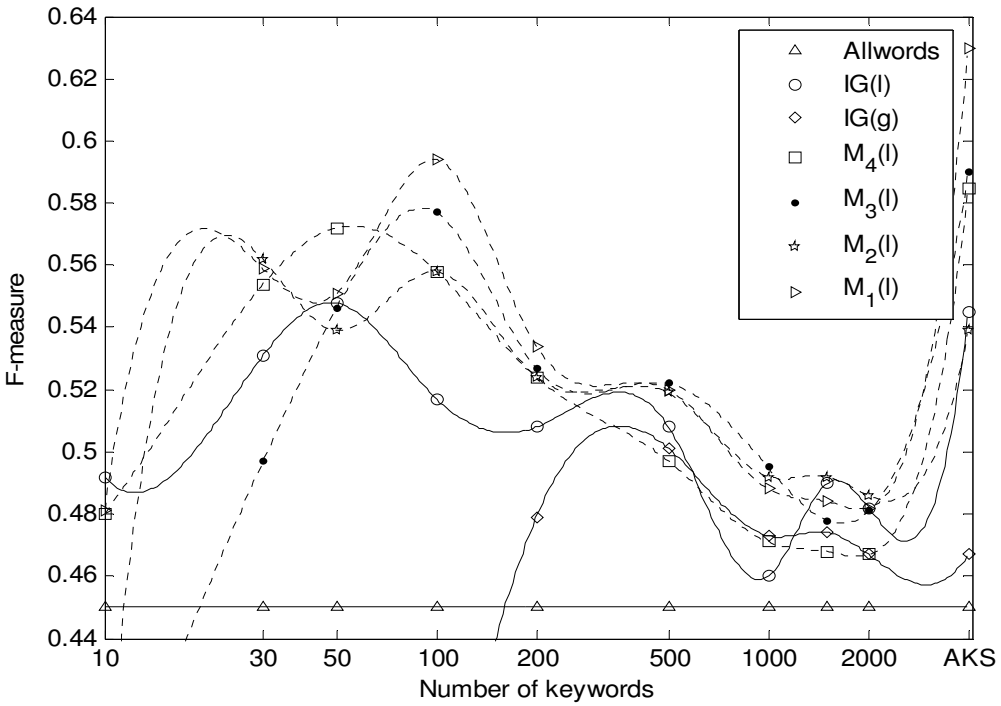


Figure B.18 Macro-averaged F-measure results for Wap dataset for new methods

REFERENCES

1. Yang, Y. and J. O. Pedersen, "A Comparative Study on Feature Selection in Text Categorization", *Proceedings of the 14th International Conference on Machine Learning*, pp. 412–420, 1997.

2. Forman, G., "An Extensive Empirical Study of Feature Selection Metrics for Text Classification", *Journal of Machine Learning Research* 3, pp. 1289–1305, 2003.

3. Debole, F. and F. Sebastiani, "Supervised Term Weighting for Automated Text Categorization", *Proceedings of SAC-03, 18th ACM Symposium on Applied Computing*, ACM Press, pp. 784–788, 2003.

4. Salton, G. and C. Buckley, "Term Weighting Approaches in Automatic Text Retrieval", *Information Processing and Management* 24 no. 5, pp. 513–523, 1988.

5. Özgür, A., L. Özgür and T. Güngör, "Text Categorization with Class-Based and Corpus-Based Keyword Selection", *Proceedings of ISCIS'05, Lecture Notes in Computer Science* 3733, Springer Verlag, pp. 607–616, 2005.

6. Joachims, T., "Text Categorization with Support Vector Machines: Learning with Many Relevant Features", *European Conference on Machine Learning (ECML)*, 1998.

7. Kohavi, R. and G. H. John, "Wrappers for Feature Subset Selection", *Artificial Intelligence*, 97(1-2), pp. 273-324, 1997.

8. Özgür, A. and T. Güngör, "Classification of Skewed and Homogeneous Document Corpora with Class-Based and Corpus-Based Keywords", *29th German Conference on Artificial Intelligence* (KI 2006), Eds. C.Freksa, M.Kohlhase and K.Schill, June 2006,

Bremen - *LNAI (Lecture Notes in Artificial Intelligence)*, Vol.4314, pp. 91-101, 2007, Springer-Verlag, Berlin Heidelberg.

9. Yang, Y. and Xin Liu, "A Re-examination of Text Categorization Methods", *Proceedings of the Twenty-Second International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 42-49, 1999.

10. Mladenic, D. and M. Grobelnik, "Feature Selection for Unbalanced Class Distribution and Naive Bayes", *Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*, pp. 258-267, 1999.

11. Dumais, S., J. Platt, D. Heckerman and M. Sahami, "Inductive Learning Algorithms and Representations for Text Categorization", *Proceedings of the 17th International Conference on Information and Knowledge Management*, pp. 148-155, Maryland, 1998.

12. Ozgur, A., "Supervised and Unsupervised Machine Learning Techniques for Text Document Categorization", Master's Thesis, Bogazici University, Turkey, 2004.

13. Soucy, P. and G. W. Mineau, "Beyond TFIDF Weighting for Text Categorization in the Vector Space Model", *IJCAI* 2005, pp. 1130-1135.

14. Salton, G., A. Wong, and C. S. Yang, "A Vector Space Model for Automatic Indexing", *Communications of the ACM*, vol. 18, nr. 11, pp. 613–620, 1975.

15. Zobel, J. and A. Moffat, "Exploring the similarity space", *SIGIR Forum*, 32(1), pp. 18–34, 1998.

16. http://www.tartarus.org/ martin/PorterStemmer/, 2004.

17. Lewis, David D., "Reuters-21578 Document Corpus V1.0", http://daviddlewis.com/resources/testcollections/reuters21578/, 2004.

18. http://svmlight.joachims.org/

19. ftp://ftp.cs.cornell.edu/pub/smart/, 2004.

20. Bakus, J. and M. S. Kamel, "Higher order feature selection for text classification", *Knowledge Information Systems* 9(4): pp. 468-491, 2006.

21. Blum, A. L. and P. Langley, "Selection of relevant features and examples in machine learning", *Artificial Intelligence* (97), pp. 245-271, 1997.

22. Chawla, Nitesh V., N. Japkowicz , A. Kotcz, "Editorial: special issue on learning from imbalanced data sets", *ACM SIGKDD Explorations Newsletter*, v.6 n.1, June 2004.

23. Forman, G., "A pitfall and solution in multi-class feature selection for text classification", *Proceedings of the twenty-first international conference on machine learning*, p.38, July 04-08, 2004, Banff, Alberta, Canada.

24. Galavotti, L., F. Sebastiani and M. Simi, "Experiments on the use of feature selection and negative evidence in automated text categorization", *Proceedings of ECDL-00, 4th European Conference on Research and Advanced Technology for Digital Libraries* (Lisbon, Portugal, 2000), pp. 59-68, 2000.

25. Guyon, I. and A. Elisseeff, "An introduction to variable and feature selection", *Journal of Machine Learning Research*, 3, pp. 1157–1182, 2003.

26. How, B. C. and K. Narayanan, "An Empirical Study of Feature Selection for Text Categorization based on Term Weightage", *Web Intelligence* 2004, pp. 599-602.

27. Lewis, David D., Y. Yang, Tony G. Rose and F. Li, "RCV1: A New Benchmark Collection for Text Categorization Research", *The Journal of Machine Learning Research*, 5, pp.361-397, 12/1/2004.

28. Rakotomamonjy, A., "Variable selection using SVM-based criteria", *The Journal of Machine Learning Research*, 3, 3/1/2003.

29. Sebastiani, F., "Machine learning in automated text categorization", *ACM Computing Surveys*, 34(1), pp. 1-47, 2002.

30. Yu, L. and H. Liu, "Efficient Feature Selection via Analysis of Relevance and Redundancy", *The Journal of Machine Learning Research*, 5, pp.1205-1224, 12/1/2004.

31. Zheng, Z. and R. Srihari, "Optimally Combining Positive and Negative Features for Text Categorization", *Proceedings of the ICML*, Workshop on Learning from Imbalanced Datasets II, Washington DC, 2003.

32. Zheng, Z., X. Wu and R. Srihari, "Feature selection for text categorization on imbalanced data", *ACM SIGKDD Explorations Newsletter*, v.6 n.1, June 2004.