Arab J Sci Eng (2016) 41:951–964
DOI 10.1007/s13369-015-1945-x

RESEARCH ARTICLE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

Improving Text Classification Performance with Random Forests-Based Feature Selection

Sameen Maruf[1] · **Kashif Javed**[2] · **Haroon A. Babri**[2]

Received: 16 April 2015 / Accepted: 19 October 2015 / Published online: 5 November 2015

Abstract Feature selection (FS) is employed to make text classification (TC) more effective. Well-known FS metrics like information gain (IG) and odds ratio (OR) rank terms without considering term interactions. Building classifiers with FS algorithms considering term interactions can yield better performance. But their computational complexity is a concern. This has resulted in two-stage algorithms such as information gain-principal component analysis (IG–PCA). Random forests-based feature selection (RFFS), proposed by Breiman, has demonstrated outstanding performance while capturing gene–gene relations in bioinformatics, but its usefulness for TC is less explored. RFFS has fewer control parameters and is found to be resistant to overfitting and thus generalizes well to new data. It does not require use of a test dataset to report accuracy and does not use conventional cross-validation. This paper investigates the working of RFFS for TC and compares its performance against IG, OR and IG–PCA. We carry out experiments on four widely used text data sets using naive Bayes' and support vector machines as classifiers. RFFS achieves macro-F_1 values higher than other FS algorithms in 73 % of the experimental instances. We also analyze the performance of RFFS for TC in terms of its parameters and class skews of the data sets and yield interesting results.

Keywords Text classification · Feature selection · Random forests · Term interactions

✉ Sameen Maruf
sameen.maruf@gmail.com

[1] Faculty of Engineering, University of Central Punjab, Lahore, Pakistan

[2] Departmental of Electrical Engineering, University of Engineering and Technology, Lahore, Pakistan

1 Introduction

Text classification (TC) is a popular application of machine learning which deals with the automatic classification of text documents into one or more predefined classes or categories [1,2]. Because of the rapidly growing digital text documents, TC finds its use in a broad spectrum of applications [3]. To apply machine learning algorithms to text documents a lot of preprocessing techniques are employed such as tokenization, stemming, stop-word removal, and pruning [4,5]. For further details, interested readers should refer to [6,7]. In TC, the data are most widely represented by the bag-of-words model [8]. The ordering and structure of the text are ignored, and a document is taken to be a sequence of terms in this model. The outcome is a data set $\mathcal{D} = \{\mathbf{d}^t, C^t\}_{t=1}^{N}$ with N documents and M terms such that the tth document is $\mathbf{d}^t \in R^M$ and its category is $C^t \in \{\pm 1\}$. The set $\mathcal{T} = \{T_1, T_2, \ldots, T_M\}$ denotes the M distinctive terms. The value of a term can be its term frequency (tf) or its term frequency–inverse document frequency (tf–idf) [9]. Various other weighting schemes are also in practice [10]. In the quest for improving TC's performance, joint occurrence of term pairs has also been proposed [11].

The feature space for a text classifier is made up of a set of words[1] with quite a large M, thus making training of the classifier computationally infeasible [12,13]. High dimensionality also compromises the classifier's accuracy. This is attributed to the presence of irrelevant and redundant features in data sets [14,15]. Irrelevant terms provide no useful information about the class or category. Similarly, redundant features are those which provide no additional information about the class beyond what has already been provided by relevant features. These irrelevant and redun-

[1] We use the words feature, variable, term, and word interchangeably.

dant features, which are detrimental for a classifier should be avoided [16,17]. To reduce the size of the feature space while enhancing the performance of text classifiers significantly, dimensionality reduction is employed [18]. This can be achieved through feature transformation or feature selection methods [19,20]. Feature transformation (FT) also known as feature extraction maps the original M features to a new space and selects the most useful m dimensions of the newly created space. On the other hand, feature selection (FS) algorithms select a feature subset of size m from the original feature space with M features, thus preserving the original meaning of the features [21,22]. In this study, our focus is on feature selection algorithms.

Feature selection algorithms can be broadly categorized into feature ranking (FR) methods and feature subset selection (FSS) methods [23]. An FR algorithm evaluates a feature using a metric and then ranks the features in decreasing order of their importance according to the metric's value [24]. The features are then chosen for classification on the basis of some threshold value or the number of top ranked features. These methods are the preferred choice for high-dimensional data such as the ones of text classification since they run in linear time [25]. Some popular FR metrics for text classification include information gain (IG) [1], Chi-square [26], and odds ratio (OR) [27]. On the other hand, FSS algorithms, which are broadly classified into filters, wrappers, and embedded methods [28,29], employ a search algorithm to find the most useful subset of features. They take the feature interactions into consideration and thus can be considered to be better than the feature ranking algorithms while building classifiers. However, FSS typically run in quadratic time and are computationally inefficient to work for text data sets [23].

Random forests-based feature selection (RFFS) [30] has been used in ecology [31], in bioinformatics for identifying the most useful genes and disease classification [32,33] and various other domains. However, its use for building text data classifiers has been less explored. In this paper, we investigate its performance for TC and compare it against two popular FR metrics (IG and OR) and one multivariate method (IG–PCA). To find out whether or not RFFS performs better than IG, OR, and IG–PCA algorithms, we experiment on four text data sets (Reuters-21578, WebKB, WAP, and TREC) using two classifiers (naive Bayes' [34] and support vector machines [35]).

The remainder of this paper is organized into four sections. Section 2 provides the related literature to this study while Sect. 3 describes the working of RFFS algorithm and compares our study to previous works. In Sect. 4, we present the experimental setup that we followed for our research. Results are provided and discussed in Sects. 5 and 6. The conclusions are drawn in Sect. 7.

2 Related Work

This section describes the work that has been done regarding feature selection in the text classification domain.

Most of the work that has been done in text classification to improve the feature selection stage involves feature ranking methods. Many comparative studies in this regard have been carried out including that by Yang and Pederson [1], Forman [9], and Mladenic and Grobelnik [27] to name a few. The former found Chi-square and IG to be the most efficient, while using k-nearest neighbour (kNN) and linear least-square fitting model (LLSF) as classifiers. Forman [9] proposed a new feature ranking metric namely bi-normal separation (BNS) and compared it to 11 other feature ranking metrics [IG, OR, document frequency (DF), etc.] using support vector machines (SVM) as classifier. His results showed that BNS outperforms the others in majority of the cases especially in high-skew situations which are quite common in TC. Mladenic and Grobelnik [27] found odds ratio to be the best and information gain to be the worst in comparison with nine other FR metrics using naive Bayes' as classifier. The quest for enhancing TC's performance is going on. More recently, Uysal and Gunal [12] proposed a new FR metric namely distinguishing feature selector (DFS) and compared its performance against Gini index, information gain, and Chi-square. Let us now describe a few popular feature ranking metrics for text classification that have been previously mentioned.

2.1 Information Gain (IG)

Information gain (IG) measures the information obtained for class prediction by knowing the presence or absence of a term in a document [1,26]. In other words, it measures the decrease in entropy when the term is present to when it is absent. It is an information theory-based metric which ranks individual terms based on their association with the class variable and neglects possible interactions between the terms themselves. It is defined as:

$$\mathrm{IG} = e(\mathrm{pos}, \mathrm{neg}) - \left[P_{\mathrm{word}} \times e(\mathrm{tp}, \mathrm{fp}) + (1 - P_{\mathrm{word}}) \times e(\mathrm{fn}, \mathrm{tn})\right] \quad (1)$$

where $e(x, y) = -\frac{x}{x+y}\log_2\frac{x}{x+y} - \frac{y}{x+y}\log_2\frac{y}{x+y}$. The pos and neg are the number of positive and negative cases respectively, P_{word} is $(\mathrm{tp}+\mathrm{fp})/(\mathrm{pos}+\mathrm{neg})$, tp denotes true positives, i.e., number of actual positive cases containing word and fn denotes false negatives, i.e., number of actual positive cases not containing word, while fp denotes false positives, i.e., number of actual negative cases containing word and tn denotes true negatives, i.e., number of actual negative cases not containing word.

2.2 Odds Ratio (OR)

Odds ratio measures the odds of the word occurring in the positive class normalized by that of the negative class [26,27]. The notion behind this feature ranking metric is that the distribution of features on the relevant documents is different from the distribution of features on the non-relevant documents. It is normally used in information retrieval, where the problem is to rank the features according to their relevance for the positive class. It is defined as:

$$\mathrm{OR} = \frac{\mathrm{tp}}{\mathrm{fp}}\frac{\mathrm{tn}}{\mathrm{fn}} \tag{2}$$

The definitions of tp, tn, fp and fn have been explained in previous subsection.

2.3 Chi-Square (CHI)

Chi-square is a statistical feature selection metric which measures the divergence from the expected distribution, assuming that the occurrence of features is independent of the class value [9]. It is defined as:

$$\begin{aligned}\mathrm{CHI} = {} & t(\mathrm{tp}, (\mathrm{tp}+\mathrm{fp}) \times P_{\mathrm{pos}}) + t(\mathrm{fn}, (\mathrm{fn}+\mathrm{tn}) \times P_{\mathrm{pos}}) \\ & + t(\mathrm{fp}, (\mathrm{tp}+\mathrm{fp}) \times P_{\mathrm{neg}}) + t(\mathrm{tn}, (\mathrm{fn}+\mathrm{tn}) \times P_{\mathrm{neg}})\end{aligned} \tag{3}$$

where $P_{\mathrm{pos}} = \frac{\mathrm{tp}+\mathrm{fn}}{\mathrm{tp}+\mathrm{fn}+\mathrm{fp}+\mathrm{tn}}$, $P_{\mathrm{neg}} = \frac{\mathrm{fp}+\mathrm{tn}}{\mathrm{tp}+\mathrm{fn}+\mathrm{fp}+\mathrm{tn}}$ and $t(\mathrm{count}, \mathrm{expect}) = \frac{(\mathrm{count}-\mathrm{expect})^2}{\mathrm{expect}}$. This test is known to not work for very small expected counts, which are quite common in text classification applications. This can be both because of having rarely occurring terms, and sometimes because of having few positive training examples for a feature.

2.4 Two-Stage Feature Selection Algorithms

As pointed out by Forman [9], the presence of redundant terms in text documents can deteriorate the classification performance of TC if feature selection algorithms used do not take care of term interactions. This has led researchers to propose two-stage algorithms [13,20]. In the first stage, an FR metric selects a subset of the most relevant terms. From this reduced feature subset, redundancy is eliminated in the second stage. The second stage is a feature transformation algorithm such as principal component analysis (PCA) [4] or latent semantic indexing (LSI) [13]. Apart from evaluating IG and OR, we also evaluated the performance of IG–PCA against that of RFFS in this study.

In the IG–PCA algorithm [4], two-stage feature selection and feature extraction is used to improve the performance of text categorization. In the first stage, each term within the document is first ranked in decreasing order of importance for classification using the information gain (IG) method. In the second stage, principal component analysis (PCA) feature extraction method is applied to the terms and a dimension reduction is carried out. Thereby, during feature ranking irrelevant terms are ignored and feature extraction is applied to the terms of highest importance.

We next describe the working of random forests, which selects important features in the forward direction taking their interactions into account.

3 Random Forests-Based Feature Selection (RFFS)

Random forest (RF) algorithm, developed by Breiman [30], is a parallel ensemble method used for classification and is an extension of his bagging predictor [36] and classification and regression tree (CART) [37] algorithms. It combines numerous binary decision trees built using several bootstrap samples of the data. In addition to this, random forests change the method of construction of classification or regression trees, that is, instead of splitting each node using the best split among all features as in standard trees, in a random forest each node is split using the best among a subset of features randomly chosen at that node [38]. It is also user-friendly in the sense that it has only two parameters: the number of variables in the random subset at each node m_{try} and the number of trees in the forest n_{tree}. The RF tree evolution algorithm can be summarized as follows:

1. The forest consists of n_{tree} trees. Each of these trees is constructed using bootstrap samples from the original data, that is, with a new training set that is drawn at random with replacement.
2. At each node of a tree:
   - a new set of eligible m_{try} features from the set of all features are selected at random
   - the feature among the m_{try} features which provides the best split according to Gini impurity function [39] is selected
3. Each tree is grown with maximum depth (i.e., no pruning is performed), such that important variables finally make it into the tree.
4. Each tree then casts a vote on its terminal nodes. For a binary target the vote will be YES or NO.
5. The forest then takes the majority votes for classification.

If N_{tr} denotes the number of instances in the training set, then its running time complexity is $n_{\mathrm{tree}} \cdot m_{\mathrm{try}} \cdot N_{\mathrm{tr}} \log N_{\mathrm{tr}}$ [23].

3.1 Variable Importance Using Random Forests

Random forests additionally give a measure of importance of the predictor variables. A variable's importance can be estimated by four different measures [40]. The most widely adopted is the permutation accuracy importance measure [41]. The prediction capability of each tree is estimated on the data not used in its construction. Those instances are called 'out-of-bag' instances (approximately 1/3 of the total instances) and are used to estimate the importance of a variable. In doing so for an lth variable, its values in the left out instances for the kth tree are first randomly permuted so that its original association with the class variable is lost. Then the permuted variable l, together with the remaining unpermuted predictor variables, is used to predict the response. The change in the classification performance for the entire forest before and after the permutation determines the importance of the lth feature. The greater the increase in the performance, more is the importance of the variable.

3.2 Comparison of Our Study with Other Similar Works

The studies based on random forests like [38] have focused on the working of the general algorithm and the usage of its package in R.[2] Chen [42] has focused on using random forests as a classifier for learning imbalanced data. Hapfelmeier [43] has used Random forests as a variable selection method by developing a new approach based on permutation tests. Amaratunga [44] has used weighted random sampling for choosing the eligible subset of features at each node. Random forest is thus popular as a classifier and has also been employed as a feature selection method namely RFFS in various domains.

There are many other parallel ensembles that have been studied for classification, for instance, the ensembles based on k-NN classifier [45] and for feature selection, for instance, Group Method of Data Handling (GMDH) [46]. Neumayer, however, failed to prove that ensembles of k-NN classifiers outperform a single classifier that is trained on the complete dataset. GMDH, on the other hand, is based on abductive network training algorithm and has yielded good results in various domains [46]. It is based on model synthesis, which although is a more adequate approach and is free from human biases, but is not a focus of this study. This can be part of a subsequent separate study where the merits of GMDH, RFFS, and other such ensembles can be compared for text classification.

When it comes to our goal of study, RFFS has most widely been used in the field of bioinformatics for gene ranking and selection [32,33,43,47] and has demonstrated promising results. The working of RFFS has been less explored for text classification. Gene expression data and text data have two attributes in common: both comprise of sparse data sets and have large number of features. The number of observations for gene expression data is much low (in mostly order of tens or sometimes hundreds) than that available for text data. They are also different in the number of classes, i.e., gene expression data are two-class problems, while text data are multi-class. Also, the skew is much higher than that in gene expression data sets. With these challenging characteristics, we are interested in determining how good RFFS is at in improving text classification performance.

When we compare the merits of RFFS over other popular feature ranking methods that have been described in this study, the first thing we realize is that FR methods are univariate methods, meaning that they consider that the words/features in a document or text corpora are independent of each other. However, it has been researched that considering term interactions can yield better performance [48]. RFFS also has a few control parameters and is found to be resistant to overfitting and thus generalizes well to new data. It does not require use of a test dataset to report accuracy (uses 'out-of-bag' instances) and does not use conventional cross-validation. It can even work with missing values in the data which is not possible with FR metrics.

We next describe our methodology for evaluating different FS algorithms.

4 Experiments

This section describes our experiments, summary of the data sets and the mechanism for evaluating feature selection algorithms.

We carried out our experiments using the challenge learning object package (CLOP) [49], which is implemented in MATLAB [50]. For RFFS and OR, the CLOP implementation was used while we implemented IG and IG–PCA in MATLAB. RFFS uses the R package 'randomForest.'[3] The two main parameters for RFFS are m_{try}, the number of input features randomly chosen at each split and n_{tree}, the number of trees in the forest. These are user-defined. We have used the default values, i.e., $m_{\text{try}} = \sqrt{M}$ and $n_{\text{tree}} = 100$. Brieman suggests trying $1/2 \cdot \sqrt{M}$, or $2 \cdot \sqrt{M}$ for m_{try} if the default does not produce good results [30]. For n_{tree}, greater is its value, greater is the stability of the feature score estimates generated by the algorithm.

4.1 Data Description

We investigated the performance of IG, OR, IG–PCA and RFFS on four widely used text classification data sets, namely, WebKB, WebACE, TREC, and Reuters-21578. The

[2] http://www.r-project.org.

[3] http://www.r-project.org.

Table 1 Summary of the data sets R8 and WebKB

R8 ($M = 4095$, $N = 7674$)					**WebKB** ($M = 522$, $N = 4168$)				
Class	N_{C_T}	$N_{C_{\mathrm{Tr}}}$	$N_{C_{Ts}}$	Skew	Class	N_{C_T}	$N_{C_{\mathrm{Tr}}}$	$N_{C_{Ts}}$	Skew
Acq	2292	1596	696	1:3	Student	1625	1085	540	1:3
Crude	374	253	121	1:22	Faculty	1116	745	371	1:4
Earn	3923	2840	1083	1:2	Course	926	620	306	1:5
Grain	51	41	10	1:134	Project	501	335	166	1:8
Interest	271	190	81	1:29	–	–	–	–	–
Money-fx	293	206	87	1:27	–	–	–	–	–
Ship	144	108	36	1:51	–	–	–	–	–
Trade	326	251	75	1:22	–	–	–	–	–
Total	7674	5485	2189	–	–	4168	2785	1383	–

M and N denote the number of terms and documents, respectively. $N_{\mathrm{C_T}}$ denotes the total number of documents in a class while $N_{\mathrm{C_{Tr}}}$ and $N_{\mathrm{C_{Ts}}}$ denote the number of documents in training set and test set of a class, respectively. Skew is the ratio of $N_{\mathrm{C_{Tr}}}$ to the total number of training documents

WebKB data set, also made available by Ana Cardoso-Cachopo, consists of Web pages collected by the World Wide Knowledge Base (WebKb) project of the CMU (Carnegie Mellon University) text learning group.[4] The WebACE Projects data set WAP consists of Web pages listed in the subject hierarchy of Yahoo! For the TREC data sets, FBIS version was used. The WAP and FBIS data sets were provided by Karypis.[5] For the Reuters-21578 data set, we used the version R8 made available by Ana Cardoso-Cachopo [51]. This data set contains documents which came out on the Reuters newswire in 1987. These four data sets use the term counts in a document as the term representation.

R8 and WebKb data sets were already split into training and test sets [51]. For the WAP and FBIS data sets, this was not the case. In these, the common words were already removed using stop-word list and words stemmed using Porters suffix stripping algorithm [52]. We employed a rare word cutoff and a too frequent word cutoff, i.e., the terms which appeared in less than three documents and the ones which appeared in more than 25 % of the documents were removed. The data sets were then randomly split into training and test sets. The summary of the data sets are given in Tables 1 and 2.

4.2 Performance Evaluation

A feature selection (FS) algorithm (IG, OR, IG–PCA, or RFFS) when applied on a class of a data set outputs a list of terms in the decreasing order of relevance to the class with the term having the highest relevance as the first term. From this ranked list, nested subsets containing different number of top ranked features are generated which are then used to train a classifier so that the feature selection capability of a feature selection algorithm can be evaluated. For the WebKB data set, size of the nested subsets are 20, 50, 100, 150, 200, 300, 450 while for the other data sets, we have measured the performance of the feature selection algorithms at subsets consisting of top 20, 50, 100, 150, 200, 300, 450, 550, 700, 900, 1000, 1300, 1500 and 2000 terms.

Two popular classifiers in the text classification community namely: naive Bayes' [34,53] and support vector machines (SVM) with a linear kernel [2,35] have been used. The CLOP implementation of these classifiers was used. There are no tuning parameters for naive Bayes', while for SVM, we used the default values of the tuning parameters. The procedure above is repeated for each class of the four data sets.

Since our data sets were originally multi-class, so to measure the performance of the feature selection methods, we used the popular macro-F_1 measure [9]. The F-measure, in general, measures the classifier's effectiveness at both precision and recall. In many studies, it is taken to be is the ultimate measure of performance of the classifier. The F_1 measure, however, is simply the harmonic mean of precision and recall and gives equal weight to both of them. The macro-F_1 measure, that we used in this study, is the average of F_1 measure over the various classes of a data set. This measure gives equal weight to all classes of a data set whether they have high or low skew [9]. It is given by:

$$\text{Macro } F_1 = \frac{\sum_{i=1}^{|C|}(F_1)_i}{|C|} \tag{4}$$

where i is the index variable of the classes of a data set, $F_1 = \frac{2\times\text{recall}\times\text{precision}}{\text{recall}+\text{precision}}$, $\text{recall} = \frac{\text{tp}}{\text{tp}+\text{fn}}$ and $\text{precision} = \frac{\text{tp}}{\text{tp}+\text{fp}}$.

In the discussion to follow, $\mathbf{G}$ is a set that contains the top ranked terms of an FS algorithm and $|\ \mathbf{G}\ |$ represents its cardinality. Let $\%\Delta_{F_1} = F_{1_{\mathrm{RFFS}}} - F_{1_x}$, where $x \in \{\text{IG, OR, IG–PCA}\}$ denotes the %age improvement in the macro F_1.

[4] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/.

[5] http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download.

Table 2 Summary of the data set WAP and FBIS

WAP ($M = 8423$, $N = 1560$)					**FBIS** ($M = 1907$, $N = 2463$)				
Class	N_{C_T}	$N_{C_{Tr}}$	$N_{C_{Ts}}$	Skew	Class	N_{C_T}	$N_{C_{Tr}}$	$N_{C_{Ts}}$	Skew
People	168	113	55	1:10	3	48	33	15	1:52
Television	130	96	34	1:11	4	43	25	18	1:69
Health	341	246	95	1:4	11	46	27	19	1:64
Media	18	11	7	1:98	12	121	84	37	1:21
Art	15	9	6	1:121	95	38	30	8	1:57
Film	196	127	69	1:9	100	92	58	34	1:30
Business	76	49	27	1:22	108	94	73	21	1:24
Cable	33	24	9	1:46	111	387	275	112	1:7
Culture	54	38	16	1:29	118	139	98	41	1:18
Music	91	66	25	1:17	119	46	32	14	1:54
Politics	65	48	17	1:23	142	506	361	145	1:5
Sports	97	72	25	1:15	161	65	42	23	1:41
Review	91	63	28	1:17	187	125	89	36	1:19
Technology	37	25	12	1:44	189	358	250	108	1:7
Stage	13	12	1	1:91	202	190	120	70	1:14
Entertainment	5	3	2	1:364	221	119	95	24	1:18
Online	40	26	14	1:42	240	46	32	14	1:54
Industry	44	31	13	1:35	–	–	–	–	–
Variety	35	27	8	1:40	–	–	–	–	–
Multimedia	11	6	5	1:182	–	–	–	–	–
Total	1560	1092	468	–	–	2463	1724	739	–

M and N denote the number of terms and documents, respectively. N_{C_T} denotes the total number of documents in a class while $N_{C_{Tr}}$ and $N_{C_{Ts}}$ denote the number of documents in training set and test set of a class, respectively. Skew is the ratio of $N_{C_{Tr}}$ to the total number of training documents

If $\%\Delta_{F_1} \geq 1$, its a win for RFFS. If $-1 < \%\Delta_{F_1} < 1$, then we consider it to be a tie. Otherwise, a loss. In the following tables, RFFS's win over IG, OR, or IG–PCA is represented by a $\bullet$, its loss by a $\circ$, and the absence of a symbol indicates a tie.

5 Results

Section 5.1 compares and analyzes the results that we obtained in terms of macro-F_1 measure for the feature selection methods RFFS, IG, OR, and IG–PCA on the four data sets. Then, we analyze the changes in performance of RFFS, also in terms of macro-F_1 measure, upon varying the values of 'm_{try}' and 'n_{tree}' parameters for the four data sets in Sect. 5.2. And finally in the same section, we will be analyzing the performance of RFFS in terms of class skews.

5.1 RFFS Versus IG, OR and IG–PCA

5.1.1 The WebKB Data Set

Let us first look at the performance of the four FS algorithms on the WebKB data set. The m_{try} and n_{tree} of RFFS are set to 23 and 100 respectively. Table 3 provides a detailed look at the macro F_1 performance of RFFS versus IG, OR and IG–PCA. With naive Bayes, IG and OR have been outperformed by RFFS in four out of seven nested subsets, while performance of IG–PCA was found to be poorer than that of RFFS in all the cases. Almost similar results were attained when the nested subsets were evaluated by SVM. Out of the 14 experimental instances of WebKB, the performance for RFFS surpasses that of IG, OR and IG–PCA in majority cases. Hence, we can conclude that RFFS is a better choice for WebKB.

5.1.2 The WAP Data Set

Now, we are going to compare the performance of RFFS, IG, OR, and IG–PCA for the WAP data set. Here, m_{try} is taken to be 92. The macro-F_1 values obtained by the four FS algorithms using naive Bayes' and SVM are shown in Table 4. For the naive Bayes classifier, the RFFS algorithm demonstrates performance better or comparable to IG. The other two algorithms OR and IG–PCA are outperformed by RFFS. When the rankings are evaluated by SVM, we find that RFFS performs much better than IG, OR and IG–PCA in 9, 14, and 5 nested subsets, respectively. Hence, it can be said that the WAP classification task is better performed by RFFS.

Table 3 Macro-F_1 performance of RFFS versus IG, OR, and IG–PCA algorithms on WebKB

| **G** |	Naive Bayes'				Support vector machines			
	RFFS	IG	OR	IG–PCA	RFFS	IG	OR	IG–PCA
20	0.273	0.202 •	0.218 •	0.106 •	0.580	0.543 •	0.479 •	0.458 •
50	0.330	0.316 •	0.297 •	0.199 •	0.516	0.458 •	0.445 •	0.477 •
100	0.455	0.412 •	0.381 •	0.277 •	0.695	0.649 •	0.614 •	0.429 •
150	0.482	0.464 •	0.443 •	0.295 •	0.782	0.752 •	0.713 •	0.508 •
200	0.492	0.496	0.488	0.308 •	0.784	0.774 •	0.765 •	0.605 •
300	0.505	0.519 ○	0.516 ○	0.356 •	0.786	0.788	0.788	0.672 •
450	0.538	0.541	0.542	0.374 •	0.806	0.811	0.810	0.736 •

G contains the top ranked terms and | **G** | represents its cardinality. ○ and • denote the improvement/degradation of IG, OR, or IG–PCA w.r.t. RFFS in terms of macro F_1

Table 4 Macro F_1 performance of RFFS versus IG, OR and IG–PCA algorithms on WAP

| **G** |	Naive Bayes'				Support vector machines			
	RFFS	IG	OR	IG–PCA	RFFS	IG	OR	IG–PCA
20	0.225	0.244 ○	0.219	0.148 •	0.280	0.253 •	0.226 •	0.210 •
50	0.290	0.258 •	0.225 •	0.181 •	0.303	0.303	0.266 •	0.308
100	0.286	0.264 •	0.239 •	0.163 •	0.356	0.385 ○	0.310 •	0.418 ○
150	0.293	0.295	0.230 •	0.154 •	0.402	0.414 ○	0.304 •	0.439 ○
200	0.289	0.310 ○	0.240 •	0.138 •	0.425	0.410 •	0.309 •	0.440 ○
300	0.337	0.342	0.266 •	0.133 •	0.514	0.504 •	0.342 •	0.482 •
450	0.369	0.370	0.310 •	0.195 •	0.579	0.560 •	0.445 •	0.568 •
550	0.346	0.345	0.284 •	0.131 •	0.599	0.595	0.496 •	0.595
700	0.371	0.358 •	0.328 •	0.124 •	0.627	0.568 •	0.544 •	0.609 •
900	0.385	0.355 •	0.334 •	0.123 •	0.625	0.585 •	0.551 •	0.623
1000	0.382	0.347 •	0.348 •	0.123 •	0.649	0.583 •	0.580 •	0.631 •
1300	0.346	0.326 •	0.346	0.084 •	0.638	0.586 •	0.592 •	0.644
1500	0.331	0.323	0.339	0.081 •	0.653	0.576 •	0.594 •	0.646
2000	0.336	0.337	0.352 ○	0.080 •	0.611	0.590 •	0.596 •	0.660 ○

G contains the top ranked terms and | **G** | represents its cardinality. ○ and • denote the improvement/degradation of IG, OR, or IG–PCA w.r.t. RFFS in terms of macro F_1

5.1.3 The FBIS Data Set

The m_{try} of RFFS is initialized to 44. Table 5 provides the performance comparison of RFFS, IG, OR and IG–PCA for the FBIS data set. We can see that with naive Bayes' classifier RFFS outperforms IG, OR and IG–PCA in 9, 10 and 12 nested subsets respectively. A similar performance is exhibited by RFFS when the nested subsets are evaluated by SVM. There are 6, 9 and 12 cases where RFFS surpasses IG, OR and IG–PCA respectively. Hence, we can conclude that RFFS is the winner on the FBIS data set.

5.1.4 The R8 Data Set

Finally, we compare the four FS algorithms on the R8 data set. The m_{try} parameter of RFFS is 64. Table 6 shows their macro-F_1 values. We can observe that using naive Bayes, the highest macro F_1 value is attained by the RFFS algorithm, which comes out to be 0.601 with its top 300 ranked terms. Out of the 14 experimental instances, RFFS has outperformed IG in 12 cases, OR and IG–PCA in all cases. Using SVM, the highest macro F_1 value is exhibited by the RFFS and IG–PCA algorithms. RFFS has achieved this value with its top 1300 ranked terms while IG–PCA's top 1000 ranked terms yielded this value. We can see also that RFFS performs much better than IG and OR but its performance, though good, is quite poor in comparison with IG–PCA. Also, if we look closely at the results obtained by RFFS and IG–PCA (with SVM) we see that as the size of the nested subset increases, the difference between the values of macro-F_1 measure obtained using these algorithms decreases.

Table 5 Macro-F_1 performance of RFFS versus IG and OR algorithms on FBIS

| **G** |	Naive Bayes'				Support vector machines			
	RFFS	IG	OR	IG–PCA	RFFS	IG	OR	IG–PCA
20	0.251	0.241 •	0.238 •	0.055 •	0.239	0.268 ○	0.258 ○	0.178 •
50	0.298	0.335 ○	0.280 •	0.108 •	0.423	0.420	0.330 •	0.311 •
100	0.413	0.392 •	0.370 •	0.149 •	0.577	0.617 ○	0.511 •	0.352 •
150	0.459	0.434 •	0.421 •	0.163 •	0.615	0.641 ○	0.576 •	0.379 •
200	0.513	0.462 •	0.463 •	0.164 •	0.664	0.657	0.622 •	0.410 •
300	0.549	0.493 •	0.495 •	0.192 •	0.700	0.688 •	0.677 •	0.444 •
450	0.552	0.496 •	0.514 •	0.244 •	0.732	0.703 •	0.712 •	0.502 •
550	0.556	0.504 •	0.535 •	0.241 •	0.742	0.719 •	0.715 •	0.522 •
700	0.547	0.509 •	0.535 •	0.232 •	0.742	0.726 •	0.737	0.568 •
900	0.536	0.525 •	0.523 •	0.230 •	0.753	0.732 •	0.736 •	0.597 •
1000	0.516	0.513	0.525	0.242 •	0.752	0.732 •	0.742 •	0.602 •
1300	0.519	0.526	0.522	0.219 •	0.747	0.742	0.744	0.633 •
1500	0.517	0.498	0.486	0.205	0.740	0.747	0.754	0.650

G contains the top ranked terms and | **G** | represents its cardinality. ○ and • denote the improvement/degradation of IG, OR, or IG–PCA w.r.t. RFFS in terms of macro F_1

Table 6 Macro-F_1 performance of RFFS versus IG, OR and IG–PCA algorithms on R8

| **G** |	Naive Bayes'				Support vector machines			
	RFFS	IG	OR	IG–PCA	RFFS	IG	OR	IG–PCA
20	0.490	0.467 •	0.466 •	0.284 •	0.476	0.360 •	0.444 •	0.586 ○
50	0.482	0.499 ○	0.438 •	0.352 •	0.493	0.473 •	0.438 •	0.634 ○
100	0.566	0.493 •	0.480 •	0.395 •	0.689	0.619 •	0.508 •	0.748 ○
150	0.569	0.498 •	0.487 •	0.398 •	0.759	0.714 •	0.531 •	0.787 ○
200	0.597	0.545 •	0.496 •	0.406 •	0.772	0.726 •	0.614 •	0.818 ○
300	0.601	0.555 •	0.509 •	0.423 •	0.771	0.764 ○	0.655 •	0.829 ○
450	0.585	0.546 •	0.518 •	0.425 •	0.802	0.781 •	0.697 •	0.858 ○
550	0.571	0.542 •	0.537 •	0.417 •	0.802	0.803	0.720 •	0.859 ○
700	0.566	0.542 •	0.534 •	0.425 •	0.798	0.812 ○	0.782 •	0.858 ○
900	0.559	0.542 •	0.531 •	0.412 •	0.823	0.845 ○	0.811 •	0.858 ○
1000	0.555	0.536 •	0.518 •	0.416 •	0.836	0.852 ○	0.822 •	0.863 ○
1300	0.529	0.516 •	0.509 •	0.415 •	0.863	0.838 •	0.843 •	0.858
1500	0.527	0.521	0.506 •	0.411 •	0.860	0.852	0.840 •	0.859
2000	0.518	0.506 •	0.506 •	0.429 •	0.858	0.864	0.846 •	0.858

G contains the top ranked terms and | **G** | represents its cardinality. ○ and • denote the improvement/degradation of IG, OR, or IG–PCA w.r.t. RFFS in terms of macro F_1

5.2 Performance of RFFS with Respect to m_{try}, n_{tree} and Class Skew

5.2.1 Effect of m_{try}

The performance of RFFS was analyzed in the previous section by taking the default values of m_{try} and n_{tree}, i.e., $m_{\text{try}} = \sqrt{M}$ and $n_{\text{tree}} = 100$ into account. In this section, we investigate the performance of RFFS by varying the m_{try} values to the other two values suggested by Breiman, i.e., $1/2 \cdot \sqrt{M}$, or $2 \cdot \sqrt{M}$ and compare those results with the ones we already obtained for $m_{\text{try}} = \sqrt{M}$. The value of n_{tree} is fixed at 100, the default.

Results of RFFS obtained for m_{try} values of 11, 23 and 46, as per Breiman's suggestion, are listed in Table 7 for the WebKB case. It is revealed that the performance of RFFS for WebKB (a data set with small number of terms) turns out to be the best with $m_{\text{try}} = \sqrt{M} = 23$ in majority of the cases. For WAP, RFFS was run using m_{try} values of 46, 92, and 184. Table 8 provides a detailed comparison. The value of m_{try} equal to $\sqrt{M}$ gives the best performance for RFFS in majority cases. Similarly, upon taking m_{try} as 22, 44 and 87 for the

Table 7 Macro F_1 performance of RFFS for WebKB data set upon varying m_{try}

$\mid \mathbf{G} \mid$	Naive Bayes'				Support vector machines			
	m_{try}			$m_{try_{best}}$	m_{try}			$m_{try_{best}}$
	11	23	46		11	23	46	
20	0.241	0.273	0.251	23	0.524	0.580	0.545	23
50	0.338	0.330	0.352	46	0.508	0.516	0.563	46
100	0.444	0.455	0.464	23	0.680	0.695	0.725	46
150	0.471	0.482	0.490	23	0.760	0.782	0.775	23
200	0.484	0.492	0.490	23	0.783	0.784	0.784	23
300	0.503	0.505	0.503	23	0.797	0.786	0.798	46
450	0.538	0.538	0.541	23	0.807	0.806	0.811	23

G contains the top ranked terms and $\mid \mathbf{G} \mid$ represents its cardinality. $m_{try_{best}}$ is chosen by taking the maximum of the macro F_1 values obtained using each m_{try}. If however the percentage difference in the macro F_1 obtained using $m_{try} = \sqrt{M}$ and the other m_{try} values is in the range $(-1, 1)$, then we consider it to be a tie and $m_{try_{best}}$ is chosen as $\sqrt{M}$

Table 8 Macro-F_1 performance of RFFS for WAP data set upon varying m_{try}

$\mid \mathbf{G} \mid$	Naive Bayes'				Support vector machines			
	m_{try}			$m_{try_{best}}$	m_{try}			$m_{try_{best}}$
	46	92	184		46	92	184	
20	0.222	0.225	0.222	92	0.256	0.280	0.285	92
50	0.268	0.290	0.286	92	0.346	0.303	0.307	92
100	0.278	0.286	0.281	92	0.356	0.356	0.363	92
150	0.272	0.293	0.291	92	0.380	0.402	0.399	92
200	0.283	0.289	0.283	92	0.391	0.425	0.397	92
300	0.298	0.337	0.316	92	0.461	0.514	0.545	184
450	0.342	0.369	0.344	92	0.549	0.579	0.624	184
550	0.328	0.346	0.387	184	0.576	0.599	0.615	184
700	0.339	0.371	0.418	184	0.584	0.627	0.628	92
900	0.335	0.385	0.412	184	0.576	0.625	0.639	184
1000	0.336	0.382	0.402	184	0.581	0.649	0.637	92
1300	0.338	0.346	0.396	184	0.611	0.638	0.647	92
1500	0.327	0.331	0.349	184	0.599	0.653	0.670	184
2000	0.311	0.336	0.318	92	0.581	0.611	0.648	184

G contains the top ranked terms and $\mid \mathbf{G} \mid$ represents its cardinality. $m_{try_{best}}$ is chosen by taking the maximum of the macro F_1 values obtained using each m_{try}. If however the percentage difference in the macro-F_1 obtained using $m_{try} = \sqrt{M}$ and the other m_{try} values is in the range $(-1, 1)$, then we consider it to be a tie and $m_{try_{best}}$ is chosen as $\sqrt{M}$

FBIS data set, RFFS's performance is tabulated in Table 9. We can see that for both the classifiers, $m_{try} = 44$ yields better performance than the other m_{try} values in majority of the cases.

For the R8 data set, m_{try} values come out to be 32, 64 and 128 for $1/2 \cdot \sqrt{M}$, $\sqrt{M}$ and $2 \cdot \sqrt{M}$, respectively. The variation in the performance of RFFS is given in Table 10. For both, naive Bayes' and SVM classifiers, the best value of m_{try} for majority of feature subsets comes out to be 128. As R8 has a large number of features, selecting features from a larger set of features for each node instead of just $\sqrt{M}$ has resulted in better performance for RFFS.

Hence, we can conclude that for data sets with large number of terms, one should not choose $m_{try} = 1/2 \cdot \sqrt{M}$ as a parameter value for RFFS.

Table 9 Macro-F_1 performance of RFFS for FBIS data set upon varying m_{try}

| **G** |	Naive Bayes'				Support vector machines			
	m_{try}			$m_{\text{try}_{\text{best}}}$	m_{try}			$m_{\text{try}_{\text{best}}}$
	22	44	87		22	44	87	
20	0.277	0.251	0.268	22	0.245	0.239	0.245	44
50	0.288	0.298	0.310	87	0.394	0.423	0.474	87
100	0.443	0.413	0.441	22	0.575	0.577	0.604	87
150	0.463	0.459	0.475	87	0.611	0.615	0.630	87
200	0.497	0.513	0.511	44	0.653	0.664	0.668	44
300	0.533	0.549	0.533	44	0.696	0.700	0.702	44
450	0.552	0.552	0.542	44	0.723	0.732	0.728	44
550	0.550	0.556	0.547	44	0.734	0.742	0.740	44
700	0.535	0.547	0.531	44	0.738	0.742	0.752	44
900	0.545	0.536	0.531	22	0.750	0.753	0.743	44
1000	0.534	0.516	0.538	87	0.755	0.752	0.747	44
1300	0.524	0.519	0.521	44	0.750	0.747	0.749	44
1500	0.517	0.517	0.522	44	0.754	0.740	0.747	22

G contains the top ranked terms and | **G** | represents its cardinality. $m_{\text{try}_{\text{best}}}$ is chosen by taking the maximum of the macro F_1 values obtained using each m_{try}. If however the percentage difference in the macro F_1 obtained using $m_{\text{try}} = \sqrt{M}$ and the other m_{try} values is in the range $(-1, 1)$, then we consider it to be a tie and $m_{\text{try}_{\text{best}}}$ is chosen as $\sqrt{M}$

Table 10 Macro-F_1 performance of RFFS for R8 data set upon varying m_{try}

| **G** |	Naive Bayes'				Support vector machines			
	m_{try}			$m_{\text{try}_{\text{best}}}$	m_{try}			$m_{\text{try}_{\text{best}}}$
	32	64	128		32	64	128	
20	0.478	0.490	0.469	64	0.433	0.476	0.464	64
50	0.448	0.482	0.501	128	0.503	0.493	0.589	128
100	0.533	0.566	0.593	128	0.641	0.689	0.730	128
150	0.554	0.569	0.623	128	0.734	0.759	0.788	128
200	0.560	0.597	0.628	128	0.747	0.772	0.796	128
300	0.570	0.601	0.622	128	0.768	0.771	0.811	128
450	0.590	0.585	0.593	64	0.775	0.802	0.800	64
550	0.599	0.571	0.593	32	0.784	0.802	0.806	64
700	0.573	0.566	0.589	128	0.794	0.798	0.828	128
900	0.558	0.559	0.555	64	0.800	0.823	0.849	128
1000	0.550	0.555	0.553	64	0.806	0.836	0.857	128
1300	0.540	0.529	0.541	128	0.846	0.863	0.857	64
1500	0.536	0.527	0.539	128	0.848	0.860	0.857	64
2000	0.516	0.518	0.542	128	0.862	0.858	0.860	64

G contains the top ranked terms and | **G** | represents its cardinality. $m_{\text{try}_{\text{best}}}$ is chosen by taking the maximum of the macro-F_1 values obtained using each m_{try}. If, however, the percentage difference in the macro F_1 obtained using $m_{\text{try}} = \sqrt{M}$ and the other m_{try} values is in the range $(-1, 1)$, then we consider it to be a tie and $m_{\text{try}_{\text{best}}}$ is chosen as $\sqrt{M}$

5.2.2 *Effect of* n_{tree}

In this section, we analyze the results obtained by running the RFFS algorithm for the four data sets using $n_{\text{tree}} = 500$ and compare those to the results obtained previously for $n_{\text{tree}} = 100$. The value of m_{try} is fixed at $\sqrt{M}$, the default. Keeping the limitation of page numbers in view, we present the results for the data sets with the largest and smallest number of instances.

Among the four data sets, R8 has the largest number of instances. Lets refer to Table 11 to see the variation in the performance of RFFS. Although $n_{\text{tree}} = 100$ wins, the dif-

Table 11 Macro F_1 performance of RFFS for R8 data set upon varying n_{tree}

| **G** |	Naive Bayes'			Support vector machines		
	n_{tree}		$n_{\text{tree}_{\text{best}}}$	n_{tree}		$n_{\text{tree}_{\text{best}}}$
	100	500		100	500	
20	0.490	0.486	100	0.476	0.458	100
50	0.482	0.481	100	0.493	0.482	100
100	0.566	0.579	500	0.689	0.685	100
150	0.569	0.595	500	0.759	0.774	500
200	0.597	0.599	100	0.772	0.799	500
300	0.601	0.622	500	0.771	0.792	500
450	0.585	0.598	500	0.802	0.793	100
550	0.571	0.574	100	0.802	0.790	100
700	0.566	0.562	100	0.798	0.804	100
900	0.559	0.553	100	0.823	0.828	100
1000	0.555	0.542	100	0.836	0.836	100
1300	0.529	0.540	500	0.863	0.856	100
1500	0.527	0.536	100	0.860	0.862	100
2000	0.518	0.529	500	0.858	0.862	100

G contains the top ranked terms and | **G** | represents its cardinality. $n_{\text{tree}_{\text{best}}}$ is chosen by taking the maximum of the Macro F_1 values obtained using each n_{tree}. If however the percentage difference in the macro F_1 obtained using $n_{\text{tree}} = 100$ and $n_{\text{tree}} = 500$ values is in the range $(-1, 1)$, then we consider it to be a tie and $n_{\text{tree}_{\text{best}}}$ is chosen as 100

Table 12 Macro F_1 performance of RFFS for WAP data set upon varying n_{tree}

| **G** |	Naive Bayes'			Support vector machines		
	n_{tree}		$n_{\text{tree}_{\text{best}}}$	n_{tree}		$n_{\text{tree}_{\text{best}}}$
	100	500		100	500	
20	0.225	0.232	100	0.280	0.269	100
50	0.290	0.258	100	0.303	0.304	100
100	0.286	0.303	500	0.356	0.358	100
150	0.293	0.282	100	0.402	0.368	100
200	0.289	0.286	100	0.425	0.392	100
300	0.337	0.312	100	0.514	0.451	100
450	0.369	0.349	100	0.579	0.534	100
550	0.346	0.368	500	0.599	0.567	100
700	0.371	0.380	100	0.627	0.604	100
900	0.385	0.348	100	0.625	0.629	100
1000	0.382	0.338	100	0.649	0.591	100
1300	0.346	0.345	100	0.638	0.605	100
1500	0.331	0.340	100	0.653	0.619	100
2000	0.336	0.327	100	0.611	0.595	100

G contains the top ranked terms and | **G** | represents its cardinality. $n_{\text{tree}_{\text{best}}}$ is chosen by taking the maximum of the Macro F_1 values obtained using each n_{tree}. If however the percentage difference in the macro F_1 obtained using $n_{\text{tree}} = 100$ and $n_{\text{tree}} = 500$ values is in the range $(-1, 1)$, then we consider it to be a tie and $n_{\text{tree}_{\text{best}}}$ is chosen as 100

ference between the macro-F_1 values obtained using both n_{tree} values for majority of cases, especially for larger feature subsets is trivial. This is due to the fact that since R8 has quite a large number of instances, so increase in number of trees in the forest can make the performance better.

Next, we refer to Table 12 to see the performance on WAP. It can be observed that for 26 out of 28 cases $n_{\text{tree}} = 100$ yields better performance than $n_{\text{tree}} = 500$. We obtained similar results for the other two data sets. We can conclude that the performance of RFFS does not vary significantly by changing n_{tree}, the number of trees in the forest.

5.2.3 Class Skew Analysis

The one thing that affects the performance of any feature selection algorithm when it comes to text data is the class skew. So, to analyze the performance of RFFS for low-to-moderate and high-skew classes of a data set and compare it to IG, OR and IG–PCA, we first set a threshold, i.e., we took the classes having skews greater than 1:50 as high skew and the rest as having low-to-moderate skew. Employing this threshold, there were two classes of R8 ("grain" and "ship"), no class of WebKb, five classes of WAP ("Media", "Art", "Stage", "Entertainment" and "Multimedia") and six classes of FBIS (3, 4, 11, 95, 119 and 240) in the high-skew category.

We obtained the average F_1-measure of the low-to-moderate-skew and high-skew classes of each data set for all the FS algorithms and tabulated the results as given in Table 13.

For the R8 data set, using naive Bayes' as classifier, it can be seen that in both low-skew and high-skew situations RFFS wins over IG, OR and IG–PCA in 97.6, and 57 % cases, respectively. Using SVM classifier, it can be seen that in both low-skew and high-skew situations RFFS wins over IG, OR and IG–PCA in 61.9 and 40.5 % cases, respectively. The deterioration in the results for high-skew situations is because the performance of RFFS with respect to IG–PCA is very lossy in high-skew situations.

For the WAP data set, using naive Bayes' as classifier, it can be seen that in both low-skew and high-skew situations RFFS wins over IG, OR and IG–PCA in 57.1 and 95.2 % cases, respectively. Using SVM classifier, the performance of RFFS with respect to IG and IG–PCA is very lossy in low-skew situations. Overall for SVM, it can be seen that in both low-skew and high-skew situations RFFS wins over IG, OR, and IG–PCA in 42.8 and 78.6 % cases, respectively.

For the FBIS data set, using naive Bayes' as classifier, it can be seen that in both low-skew and high-skew situations RFFS wins over IG, OR and IG–PCA in 61.5 and 89.7 % cases, respectively. While for SVM classifier, it can be seen that in both low-skew and high-skew situations RFFS

Table 13 RFFS versus IG, OR and IG–PCA in terms of class skew

	Low-to-moderate skew			High skew			Row-wise total
R8 and naive Bayes'	IG	OR	IG–PCA	IG	OR	IG–PCA	
Wins of RFFS	13	14	14	5	5	14	65
Loss of RFFS	0	0	0	3	3	0	6
Tie with RFFS	1	0	0	6	6	0	13
R8 and SVM							
Wins of RFFS	7	14	5	8	8	1	43
Loss of RFFS	1	0	7	4	2	13	27
Tie with RFFS	6	0	2	2	4	0	14
WAP and naive Bayes'							
Wins of RFFS	1	9	14	13	13	14	64
Loss of RFFS	9	1	0	1	1	0	12
Tie with RFFS	4	4	0	0	0	0	8
WAP and SVM							
Wins of RFFS	3	13	2	13	11	9	51
Loss of RFFS	6	0	8	1	0	4	19
Tie with RFFS	5	1	4	0	3	1	14
FBIS and naive Bayes'							
Wins of RFFS	5	6	13	9	13	13	59
Loss of RFFS	2	4	0	2	0	0	8
Tie with RFFS	6	3	0	2	0	0	11
FBIS and SVM							
Wins of RFFS	10	10	13	4	1	13	51
Loss of RFFS	0	0	0	6	7	0	13
Tie with RFFS	3	3	0	3	5	0	14

Win/loss denote the number of cases in which RFFS shows significant improvement/degradation w.r.t. IG, OR, or IG–PCA in terms of average F_1

wins over IG, OR and IG–PCA in 84.6 and 46.1 % cases, respectively. The deterioration in the results for high-skew situations is because the performance of RFFS with respect to IG and OR is very lossy in high-skew situations.

6 Discussion

The main objective of this study was to find out whether or not RFFS shows improved performance over IG, OR and IG–PCA methods for text data sets. Table 14 summarizes our results by tabulating collectively the number of experimental instances in which RFFS shows improved performance in comparison with IG, OR, and IG–PCA in terms of macro-F_1 measure. The effectiveness of RFFS algorithm for text classification is quite clear.

The following points are worth noting:

- In terms of macro F_1, RFFS shows improved performance in comparison with IG, OR and IG–PCA in 210 out of the total 287 instances.
- In terms of $\%\Delta_{F_1}$, the average improvement of RFFS over IG, OR and IG–PCA using Naive Bayes as classifier is 1.96, 3.45 and 20.06 % respectively while for SVM classifier it is 1.84, 5.64, and 6.96 %, respectively.
- The improvement in the FR metric's performance is due to the fact that RFFS takes term interactions into consideration.

Table 14 RFFS versus IG, OR and IG–PCA in terms of macro F_1 measure

	Naive Bayes'			SVM		
Data set	Wins	Losses	Ties	Wins	Losses	Ties
R8	40	1	1	21	15	6
WebKB	15	2	4	17	0	4
WAP	30	3	8	29	6	7
FBIS	31	1	7	27	4	8
Total	116	7	20	94	25	25

The number of cases in which RFFS shows significant improvement/degradation (denoted by • and ○ respectively) w.r.t. IG, OR, and IG–PCA in terms of macro F_1, as marked in Tables 3, 4, 5, 6, have been collected as wins and losses of RFFS w.r.t. other FS methods. The unmarked cases in the said tables are the ties of RFFS with IG, OR, and IG–PCA

- For text data sets with fewer classes, RFFS gives better performance for smaller feature subsets.
- For text data sets with greater number of classes having extremely high skews, RFFS gives better performance for medium to large feature subsets.
- If the number of features is quite large then only one may need to try $2 \cdot \sqrt{M}$ as a value for mtry to achieve better performance than that for $m_{\text{try}} = \sqrt{M}$.
- If the number of instances is quite large than only one may need to try values of n_{tree} variable greater than 100.

7 Conclusions

Traditionally, the text classification community employs feature ranking (FR) metrics for feature selection. These methods, however, do not take term interactions into account. So we propose using random forests-based feature selection to find relevant and informative features. We evaluated the performance of four FS methods, information gain (IG), odds ratio (OR), IG–PCA, and RFFS, in terms of macro-F_1 measure using naive Bayes' and support vector machines as classifiers on four text data sets: Reuters-21578, WebKB, WAP, and TREC.

We found RFFS to surpass IG, OR and IG–PCA in 73 % of all the experimental instances while matching their performance in 15.6 % of the instances. We also analyzed the performance of RFFS by varying its parameters of m_{try} (the number of features selected at each node) and n_{tree} (the number of trees in the forest) and found that greater values than the default of m_{try} and n_{tree} should only be tried when the number of features and instances is quite large. We also found that RFFS overall performs better than conventional methods with respect to both low- and high-skew classes of text data sets. Thus we can say that using RFFS for feature selection gives promising results for text classification in comparison with IG, OR and IG–PCA.

References

1. Yang, Y.; Pedersen, J.O.: A comparative study on feature selection in text categorization. In: Proceedings of the 14th international conference on machine learning, pp. 412–420 (1997)
2. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In: Proceedings of the 10th European conference on machine learning, pp. 137–142 (1998)
3. Sebastiani, F.: Machine learning in automated text categorization. ACM Comput. Surv. **34**, 1–47 (2002)
4. Uguz, H.: A two-stage feature selection method for text categorization by using information gain, principal component analysis and genetic algorithm. Knowl. Based Syst. **24**, 1024–1032 (2011)
5. Montanes, E.; Diaz, I.; Ranilla, J.; Combarro, E.F.; Fernandez, J.: Scoring and selecting terms for text categorization. IEEE Intell. Syst. **20**, 40–47 (2005)
6. Manning, C.D.; Raghavan, P.; Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, New York (2008)
7. Joachims, T.: Learning to Classify Text Using Support Vector Machines. Kluwer Academic Publishers, Dordrecht (2002)
8. Aggarwal, C.C.; Zhai, C.: A survey of text classification algorithms. In: Aggarwal, C.C.; Zhai, C. Mining Text Data, pp. 163–222. Springer, Berlin (2012)
9. Forman, G.; Guyon, I.; Elisseeff, A.: An extensive empirical study of feature selection metrics for text classification. J. Mach. Learn. Res. **3**, 1289–1305 (2003)
10. Zhang, W.; Yoshida, T.; Tang, X.: A comparative study of tf*idf, lsi and multi-words for text classification. Expert Syst. Appl. **38**, 2758–2765 (2011)
11. Badawi, D.; Altincay, H.: A novel framework for termset selection and weighting in binary text classification. Eng. Appl. Artif. Intell. **35**, 38–53 (2014)
12. Uysal, A.K.; Gunal, S.: A novel probabilistic feature selection method for text classification. Knowl. Based Syst. **36**, 226–235 (2012)
13. Meng, J.; Lin, H.; Yu, Y.: A two-stage feature selection method for text categorization. Comput. Math. Appl. **62**, 2793–2800 (2011)
14. Yu, L.; Liu, H.: Feature selection for high-dimensional data: a fast correlation based filter solution. In: Proceedings of 20th international conference on machine learning, pp. 856–863 (2003)
15. Javed, K.; Babri, H.A.; Saeed, M.: Impact of a metric of association between two variables on performance of filters for binary data. Neurocomputing **143**, 248–260 (2014)
16. Koller, D.; Sahami, M.: Toward optimal feature selection. Technical report 1996–77. Stanford InfoLab (1996)
17. Hall, M.; Holmes, G.: Benchmarking attribute selection techniques for discrete class data mining. IEEE Trans. Knowl. Data Eng. **15**, 1437–1447 (2003)
18. Makrehchi, M.: Feature ranking for text classifiers. Ph.D. thesis Department of Electrical and Computer Engineering, University of Waterloo Waterloo, Ontario, Canada (2007)
19. Javed, K.; Babri, H.A.; Saeed, M.: Feature selection based on class-dependent densities for high-dimensional binary data. IEEE Trans. Knowl. Data Eng. **24**, 465–477 (2012)
20. Uysal, A.K.; Gunal, S.: Text classification using genetic algorithm oriented latent semantic features. Exp. Syst. Appl. **41**, 5938–5947 (2014)
21. Alpaydin, E.: Introduction to Machine Learning, 2nd edition. The MIT Press, Cambridge (2010)
22. Saeed, M.; Javed, K.; Babri, H.A.: Machine learning using Bernoulli mixture models: clustering, rule extraction and dimensionality reduction. Neurocomputing **119**, 366–374 (2013)
23. Guyon, I.; Gunn, S.; Nikravesh, M.; Zadeh, L.A.: Feature Extraction: Foundations and Applications (Studies in Fuzziness and Soft Computing). Springer, New York (2006)
24. Duch, W.: Filter methods. In: Guyon, I.; Gunn, S.; Nikravesh, M.; Zadeh, L.A. Feature Extraction: Foundations and Applications, pp. 89–117. Springer, New York (2006)
25. Guyon, I.; Bitter, H.M.; Ahmed, Z.; Brown, M.; Heller, J.: Multivariate non-linear feature selection with kernel methods. In: Nikravesh, M.; Zadeh, L.; Kacprzyk, J. Soft Computing for Information Processing and Analysis, Studies in Fuzziness and Soft Computing, vol. 164, pp. 313–326. Springer, Berlin (2005)
26. Zheng, Z.; Wu, X.; Srihari, R.: Feature selection for text categorization on imbalanced data. SIGKDD Explor. Newsl. **6**, 80–89 (2004)
27. Mladenic, D.; Grobelnik, M.: Feature selection for unbalanced class distribution and naive Bayes. In: Proceedings of the 6th international conference on machine learning, pp. 258–267 (1999)
28. Kohavi, R.; John, G.H.: Wrappers for feature subset selection. Artif. Intell. **97**, 273–324 (1997)

29. Das, S.: Filters, Wrappers, and a boosting based hybrid for feature selection. In: Proceedings of the 18th international conference on machine learning, pp. 74–81 (2001)
30. Breiman, L.: Random forests. Mach. Learn. **45**, 5–32 (2001)
31. Cutler, D.; Edwards, T.C.; Beard, K.; Cutler, A.; Hess, K.; Gibson, J.; Lawler, J.: Random forests for classification in ecology. Ecology **88**, 2783–2792 (2007)
32. Dìaz-Uriarte, R.; Alvarez de Andrès, S.: Gene selection and classification of microarray data using random forest. BMC Bioinform. **7**,3 (2006). doi:10.1186/1471-2105-7-3
33. Rodenburg, W.; Heidema, A.; Boer, J.; Bovee-Oudenhoven, I.; Feskens, E.; Mariman, E.; Keijer, J.: A framework to identify physiological responses in microarray-based gene expression studies: selection and interpretation of biologically relevant genes. Physiol. Genom. **33**, 78–90 (2008).
34. Mitchell, T.M.: Machine Learning. McGraw-Hill, Inc., New York (1997)
35. Scholkopf, B.; Smola, A.: Learning with Kernels. MIT Press, Cambridge (2002)
36. Breiman, L.: Bagging predictors. Mach. Learn. **26**, 123–140 (1996)
37. Breiman, L.; Friedman, J.H.; Olshen, R.A.; Stone, C.J: Classification and Regression Trees. Chapman & Hall, New York (1984)
38. Liaw, A.; Wiener, M.: Classification and regression by randomforest. R News **2**, 18–22 (2002)
39. Strobl, C.; Boulesteix, A.L.; Zeileis, A.; Hothorn, T.: Bias in random forest variable importance measures: illustrations, sources and a solution. BMC Bioinform. **8**, 25 (2007). doi:10.1186/1471-2105-8-25
40. Breiman, L.: Manual on setting up, using, and understanding random forests v3.1. Technical report (2002)
41. Genuer, R.; Poggi, J.M.; Tuleau-Malot, C.: Variable selection using random forests. Pattern Recognit. Lett. **31**, 2225–2236 (2010)
42. Chen, C.; Liaw, A.; Breiman, L.: Using random forest to learn imbalanced data. www.stat.berkeley.edu/tech-reports/666.pdf (2004)
43. Hapfelmeier, A.; Ulm, K.: A new variable selection approach using random forests. Comput. Stat. Data Anal. **60**, 50–69 (2013)
44. Amaratunga, D.; Cabrera, J.; Yung-Seop, L.: Enriched random forests. Bioinformatics **24**(18), 2010–2014 (2008)
45. Neumayer, R.: Clustering based ensemble classification for spam filtering. In: Proceedings of the 7th workshop on data analysis (2006)
46. Abdel-Aal, R.E.: GMDH-based feature ranking and selection for improved classification of medical data. J. Biomed. Inf. **38**, 456–468 (2005)
47. Tang, R.; Sinnwell, J.P.; Li, J.; Rider, D.N.; De Andrade, M.; Biernacka, J.M.: Identification of genes and haplotypes that predict rheumatoid arthritis using random forests. BMC Proc. Genet. Anal. Workshop **16**(Suppl 7), S68 (2009)
48. Javed, K.; Maruf, S.; Babri, H.A.: A two-stage Markov blanket based feature selection algorithm for text classification. Neurocomputing **157**, 91–104 (2015)
49. Saffari, A.; Guyon, I.: Quick start guide for challenge learning object package (CLOP). Technical report. Graz University of Technology and Clopinet (2006)
50. MathWorks. MATLAB: The language of technical computing (2010)
51. Cardoso-Cachopo, A.: Improving methods for single-label text categorization. Ph.D. thesis Instituto Superior Tecnico, Universidade Tecnica de Lisboa Portugal (2007)
52. Porter, M.F.: An algorithm for suffix stripping. Program **14**, 130–137 (1980)
53. Chen, J.; Huang, H.; Tian, S.; Qu, Y.: Feature selection for text classification with Naïve Bayes. Expert Syst. Appl. **36**, 5432–5435 (2009)