DOCUMENT CLASSIFICATION USING UNIQUE AND ELITE KEYWORDS BASED ON

ENTROPY BASED PARTITIONING

A Dissertation submitted in partial fulfillment of the requirement for

the award of degree of

Master of Technology

In

Information Technology

Submitted by

Juli keshari

(Roll No.- 2K15/ISY/10)

Under the guidance of

Dr. Seba Susan

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

BAWANA ROAD, DELHI2015-2017

CERTIFICATE

This is to certify that the dissertation entitled **"Document Classification using Unique and Elite Keywords based on Entropy Based Partitioning"** has been submitted by **Ms. Juli Keshari (Roll Number: 2K15/ISY/10),** in partial fulfillment of the requirements for the award of Master of Technology degree in Information System. This work is carried out by her under my supervision and has not been submitted earlier for the award of any degree or diploma in any university to the best of my knowledge.

Date: **Dr. Seba Susan**

(Department of Information Technology)

Delhi Technological University

ACKNOWLEDGEMENT

I am very thankful to **Dr. Seba Susan** (Assistant Professor, Information Technology Department.) and all the faculty members of the Information Technology Dept. of DTU. They all provided us with immense support and guidance for the project.

I would also like to express my gratitude to the university for providing us with the laboratories, infrastructure, testing facilities and environment which allowed us to work without any obstructions.

I would also like to appreciate the support provided by our lab assistants, seniors and our peer group who aided us with all the knowledge they had regarding various topics.

Juli Keshari

2K15/ISY/10

ABSTRACT

In this project, we investigate the selection of significant keywords for document classification.

We proposed two different schemes for selecting significant keywords, elite and unique elite.

Elite Keywords are those keywords that have high term frequency in each class. This is irrespective of the frequencies of these terms in other classes. To get the high occurring terms in each class, we employ entropy based partitioning technique that is usually used in the field of information theory and coding to generate partition between symbol probabilities. So our method has the advantage as compared to other feature selection schemes that we get the exact subset of significant keywords for each class, and we do not rely on hit and trial methods. Unique elite keywords are those that are elite for a particular class and at the same time have higher occurring frequency only in that class. To measure this, we compute the entropy of each elite keyword across all classes, sort the entropies in ascending order and again employ entropy partitioning to shortlist those elite keywords that occur uniquely in this class. Comparison with the state-of-the-art methods on benchmark data sets establishes the efficiency of our method from the high percentage accuracy obtained.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Now a days, while digitization is at the peak, there are a tremendous number of documents from several fields like finance, business, social media, banking sectors, railways. The development of efficient document categorization software is a crucial link in the digitization process. Some documents are sensitive than others and should be protected in a more careful way. Here document classification takes place to classify documents in a different label of confidentiality and criticality (low, medium, high, very high etc.).

Documents obtained from social media can be classified and used for advertising.

To deal with the question of customers on various topics, industries and businesses provide customer supports for their customers and end users. The end users or customers can easily access the required information if industries and businesses use document classification to classify documents based on topics [1].

In the Modern digital world, while the use of internet grows day by day rapidly, the need for document classification, because the source of information to be managed such as social websites, web pages, financial data, chat databases, medical databases, are growing very fast and become more essential [2], [19].

The backbones of finance, businesses, science, and economy are computer networks and this leads to the immense quantity of digital document. Text mining takes place to extract high-quality hidden information and then it is classified using different classifiers to serve different purposes [4], [17].

Classification means dividing the object into one or more predefined classes. Actually, classification is part of machine learning. There are two types of machine learning techniques Supervised and Unsupervised.

Classification is supervised learning. In supervised learning, learning is done through training data. Training data consists of training examples. Each example is a pair of an input data and labelled class.

e.g. X={(d1,Y1),(d2,Y2),(d3,Y3)……….(dm,Ym) } where 'X' is training data set and each example is pair of input data 'di' where i=1,2,3,4…………….n and corresponding labelled classes 'Yj' where j = 1,2,3…………….m. More than one document can be labelled with a particular class and a document can have more than one class label. Supervised learning is done under supervision (teacher).

In Document Classification, a Model (like neural network) is trained by training document set. After training a new document is given to the model to predict the correct class for that document.

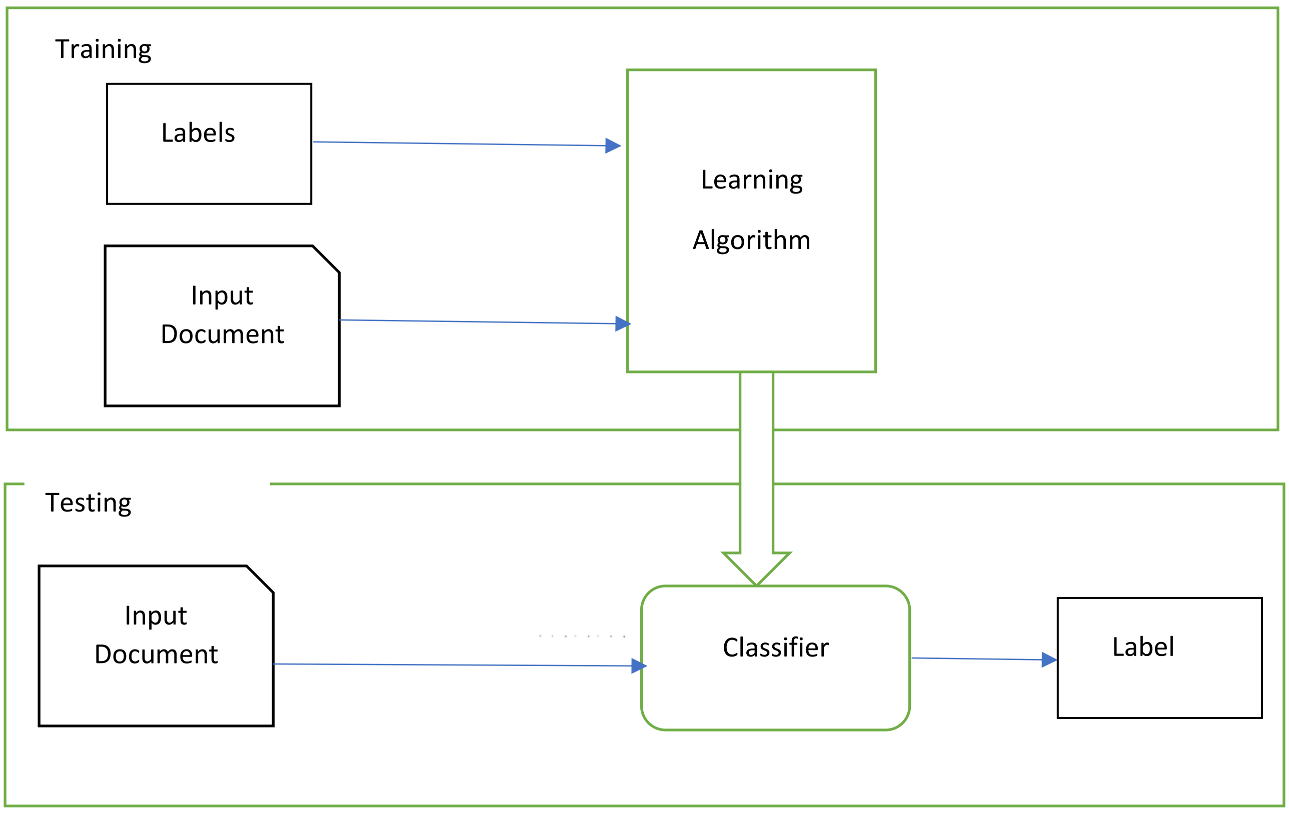

Figure1.1 Supervised Classification Overview

1.1 Motivation

In this modern era while technology growing day by day and everything gets digitally automated that leads to tons of digital document. These document needs to be classified or clustered for various purposes. There are lots of applications where document classification plays a vital role and these applications serve many purposes that give us the enthusiasm to work in this challenging field. Some of these applications are as follows [3]:

1) Filtering of Spam emails.
2) Identification of language i.e. automatically identifies language of a text.
3) Email routing i.e. sending emails to general addresses to specific addresses based on topics.
4) Genre classification i.e. classifying the genre of text automatically.
5) Quickly Searching information about an interesting topic from large databases.

1.2 Problem Statement

Given a set of documents, we have to find most significant keywords (that we call elite keywords), that will help us to categorize the documents.

For this, we have followed entropy partitioning using different schemes for two different entropies, Shannon [14] and Non-extensive entropy with Gaussian gain [15], [20].

Chapter 2

Literature Review of Existing Document Categorization Methods

2.1 Bag of Words

Document classification has a great impact on information retrieval, web search, and text mining. Mostly, Bag of words model is used. A vector is used to represent a document. Each component of a vector has a value of corresponding feature [5].

The feature can be term frequency (number of time a word or term occurs in the document) or it can be relative term frequency (the ratio of the number of time a word occurs in a document and the number of time total words occur in a document) [5], [16].

When term frequency is the feature then document D_i is represented using following equation [6].

$$D_i = \{t_1, t_2, t_3, t_4 \dots \dots .. \dots .. t_n\} \dots \dots \dots \dots \dots \dots \dots (1)$$

When the feature is relative term frequency then document D_i is represented as follows:

$$D_i = \{r_1, r_2, r_3, r_4 \dots \dots \dots .. \dots . r_n\} \dots \dots \dots \dots \dots \dots (2)$$

Where r_i is represents the ratio of the number of times a word occurs in a document and the number of times total words occur in a document. If total number of occurrence of all words is denoted by $'s'$

$$s = \sum_{i=1}^{n} r_i \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots (3)$$

Then equation for each of the, $'r_i'$ is given as:

$$r_i = \frac{t_i}{S} \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots .. \dots .. (4)$$

Where each $'t_i'$ denotes the number of times the $'ith'$ word appears in the documentSuch that the first word appears t_1 times, the second word appears t_2 times and so on the nth word appears t_n times.

E.g. A document ‘d' of project type whose word vector is represented as:

[

project

version

work

real

robot

molecular

]

And the corresponding feature vector using Eq. (1) for the document $'d'$ is as follow:

$$d \ = \{4\ 3\ 2\ 0\ \ 1\ 6\}$$

Which means the feature ‘project’ appears ‘4’ times and the feature ‘real’ not appears at all that’s why the value is ‘0’.

The document $'d'$ using eq. (2) can be represented as follows:

$$d \ = \{0.25, 0.18, 0.12\ , 0, 0.06, 0.37\}$$

That means value of relative term frequency of the term ‘project’ is ‘0.25’.

2.2 TF-IDF (Term Frequency-Inverse Document Frequency)

In information retrieval, web mining, text summarization, and text classification, most popularly used term-weighting scheme is tf-idf. In the domain of digital libraries, 83% of text based recommender system uses tf-idf [6].

In the various field like text classification, concept mining and text mining tf-idf can be used to filter stop words (most common words) [6].

Tf-idf evaluates the importance of a word 'w' to a document, 'd^' in a collection. Tf-idf value increases when term frequency increases but it is offset by the frequency of words in the collection [7].

The equation for the tf-idf of the word 'w' of the document, 'd' in a collection,' D'can be obtained by multiplying tf value of 'w^' with idf value of 'w^' and this can be shown below:$tfidf(w,d,D) = tf(w,d).idf(w,D) \dots\dots\dots\dots\dots\dots\dots\dots\dots (5)$

Using eq. (5), Text documents can be vectorised and will be given to the Classifiers and accuracy achieved is good.

Steps to victories documents and classify them is as follows:

Step 1: Create dictionary using training data set.

Step 2: Calculate ‘tf’ using one of the variants given in the table 2.1 for each of the word in each of the document training as well as testing.

Step 3: Calculate ‘idf’ of the word in using one of the variants given in the table 2.2 for each of the word in whole collection.

Step 4: Obtain ‘tf-idf’ of the word for the corresponding document by multiplying the ‘tf’ values with the corresponding ‘idf’ values obtained in step 2 and step 3.

Step 5: Give the training vectors obtained in step 5 to one of the classifier to train the model.

Step 6: Classify the test vector into one of predefined labels.

2.2.1 Term Frequency (Tf)

It is weight of a word or term in a document. In the simplest form, it is the Number of times a term $'w'$ occurs in the document $'d'$ and term frequency will be denoted as $'tf(w,d)'$. And the simplest form of $tf(w,d)$ is as follows:

$$tf(w,d) = f(w,d) \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots..\dots (6)$$

Table 2.1 variation in the Term frequency

Weighting Schemes	**Term frequency weight**
Binary	0,1
Frequency count	$f(w,d)$
Relative term frequency	$f(w,d)/\sum_{w\,\varepsilon\,d} f(w,d)$
Log Normalization	$1 + \log(f(w,d))$
Augmented frequency	$0.5 + \frac{f(w,d)}{\max_{w\,\varepsilon\,d} f(w,d)}$

For the above table 2.1, the different weighting scheme is described as follows:

Binary weighting scheme gives value ‘0’ if the word ‘w’ is not present at all, otherwise, the value will be‘1’.

The frequency count weighting scheme is the simplest one which gives value for the term as a count of occurrences of the term ‘w’ in the document, ‘d’.

Relative term frequency is a ratio between the number of times the term ‘w’ occurs in the document, ‘d’ and the total occurrences of all the term in the document, ‘d’.

Augmented frequency is used to prevent bias towards longer document.

2.2.2 Inverse Document Frequency (Idf)

It measures the quantity of information provided by the word or term. Basically, it finds out whether the word is common or rare across the collection. To find out Idf of a word $'w'$ in a document $'d'$, most commonly used equation is as below [6]:

$$idf(\mathrm{w},\mathrm{D}) = log\frac{N}{|\{\, d \in D{:}\, w\}|} \ldots\ldots\ldots\ldots\ldots\ldots\ldots..(7)$$

Where, $'N'$ denotes total number of documents in the collection.

$n_t = |\{\, d \in D{:}\, w\}|$ is the number of documents in which the term $'w'$appears.

Table 2.2 Variation in the Inverse Document Frequency

Weighting Schemes	**Term frequency weight**
Unary	1
Inverse document frequency	$\log(N\,/n_t\,)$
Inverse document frequency max	$\log\frac{\max\limits_{\mathrm{w}\,\in\mathrm{d}}(n_t)}{1+n_t}$
Probabilistic Inverse document frequency	$\log\frac{N-n_t}{n_t}$

2.3 Similarity Measures for Text Classification

How much two documents are similar, is a significant concept in document classification or information retrieval field. When it will be known that a document $'d_1'$, which has to be classified into one of two classes, is more similar to document $'d_2'$ of class 1 than document $'d_3'$ of class 2. Then the document $'d_1'$ can be easily classified into class 1 because it is more similar to document $'d_2'$ [5]. There are various measures that have been dominantly used for measuring the similarity between two documents [5].

Let $'d_1'$ and $'d_2'$ are two documents, having $'n'$ number of features, represent using Bag of words or tfidf as vectors i.e. $d_1 = \{w_{11}, w_{12}, w_{13}, \dots\dots\dots\dots, w_{1n}\}$ and $d_2 = \{w_{21}, w_{22}, w_{23}, \dots\dots\dots\dots, w_{2n}\}$.

Some of popular measures are as follows:

(a) Euclidian Distance measures: From geometry, Euclidian distance is well-known distance measures that can be used to measure the similarity between two documents in $'n'$ dimensions as [8]:

$$D_{euc} = \sqrt{\sum_{n} (d_1 - d_2)^2} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (8)$$

(b) Cosine Similarity measures: It measures the cosine of the angle between two documents $'d_1'$ and $'d_2'$ as follows [9]:

$$D_{cos} = \frac{d_1 \,.\, d_2}{||d_1||\,||d_2||} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (9)$$

(c) The Extended Jaccard coefficient: For data information retrieval, it is extended version of jaccard coefficient [5].

$$D_{jac} = \frac{d_1 \,.\, d_2}{d_1 \,.\, d_1 + d_2 \,.\, d_2 - \, d_1 \,.\, d_2} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (10)$$

(d) Similarity Measure for Text Processing: By embedding following characteristic a similarity measures for text processing have been defined [5]. A feature can be present or absent in both documents. Or a feature can be present in one document and absent in another one. All of these value impact at a different level in measuring the similarity between two documents.

(i) When a feature appears in only one document, then the difference of values of this feature associated with documents is more significant than the difference of values associated with the feature which present in both documents.

E.g. $d_1 = \{5,3,1, \ldots \ldots \ldots ..\}$ and $d_1 = \{2,0,1. \ldots \ldots \ldots ..\}$, in this feature $'w_1^{'}$ appears in both document as $'w_{11} = 5'$ and $'w_{21} = 2'$ and hence difference between $'w_{11}'$ and $'w_{21}^{'}$ is 3. On the other hand, feature $'w_2'$ is present in document $'d_1'$ and absent in $'d_2'$ as $'w_{12}^{'} = 3$ and $'w_{22}^{'} = 0$ having the difference as 3. In both the cases difference is same but despite of differences being same, feature $'w_2'$ have more impact on finding similarity between documents than the feature, $'w_1^{'}$.

(ii) When a specific feature is present in both of the documents, then similarity degree will decrease, if difference between the associated feature values is increased.

E.g. the similarity is more when the difference is equals to 3 for the feature $'w_1'$, when $'w_{11}^{'} = 5$ and $'w_{21}^{'} = 2$, than when the difference is equals to 20 for the same feature $'w_1'$with values associated as$'w_{11}^{'} = 22$ and $'w_{21}^{'} = 2$.

(iii) If the type of features, which are present in one document and absent in other, will increase in number, then the similarity degree between two documents will decrease.

E.g.$'d_1'$and $'d_2'$ are more similar when$d_1 = \{1,2,0\}$ and $d_2 = \{2,1,1\}$than when $d_1 = \{1,2,0\}$ and $d_2 = \{2,0,1\}$. Because in the first case number of the feature of type present/absent is less than the second case.

Similarity degree between the two documents is very less when all are the features of the type that are present in one document and absent in other. For each of feature $'w_i'$ if following situation occurs:

$$w_{1i} \,.\, w_{2i} = 0 \text{ and } w_{1i} + w_{2i} > 0$$

(iv) Similarity degree between $'d_1'$ and $'d_2'$ should be same as that between $'d_2'$ and $'d_1'$ i.e. it should be symmetric.

(v) The standard deviation of features also contributed while calculating the similarity degree between the two documents.

By considering all of the above six characteristic following equation is derived to calculate similarity degree,$'S_{smtp}(d_1, d_2)'$, between two documents $'d_1'$ and $'d_2'$ [5]:

$$S_{smtp}(d_1, d_2) = \frac{F(d_1, d_2) + \lambda}{1 + \lambda} \ldots \ldots \ldots \ldots \ldots \ldots .. \ldots . \ldots \ldots \ldots \ldots \ldots \ldots . \ldots . \ldots . (11)$$

$$F(d_1, d_2) = \sum_{i=1}^{n} \frac{N_*(d_{1i}, d_{2i})}{N_\cup(d_{1i}, d_{2i})} \ldots\ldots\ldots\ldots\ldots\ldots\ldots.\ldots\ldots\ldots\ldots.\ldots\ldots\ldots\ldots\ldots\ldots.(12)$$

$$N_*(d_{1i},\ d_{2i}) = \begin{cases} 0.5\ \left(\ 1 + exp\left\{\ -\left(\frac{d_{1i} - d_{2i}}{\sigma_i}\right)^2\right\}\right), \\ \qquad\qquad if\ d_{1i} \,.\, d_{2i} > 0 \\ 0,\ if\ d_{1i} = 0\ and\ d_{2i} = 0 \\ -\lambda,\ otherwise \end{cases} \ldots\ldots\ldots\ldots.\ldots\ldots..(13)$$

$$N_U(d_{1j},\ d_{2j}) = \begin{cases} 0,\ if\ d_{1j} = 0\ and\ d_{2j} = 0 \\ \qquad 1,\quad otherwise \end{cases} \ldots\ldots\ldots\ldots\ldots\ldots\ldots.\ldots\ldots\ldots..(14)$$

The above, $'S_{smtp}(d_1, d_2)'$ consider the following three cases:

(a) The feature present in both of the documents.

(b) The feature present only in one document.

(c) The feature does not appear in any of the document.

For the first case, 0.5 has been set as a lower bound and decreasing the value when the difference of the associated value increases and is scaled by Gaussian function as shown in Eq. (13), where $'\sigma_i'$ is a standard deviation of features.

For the second case, the value is set to a negative constant $' - \lambda'$ without considering the magnitude of the feature value.

For the third case, the value is set to zero, by considering the fact that if a feature is not present in any of the document then it contributes nothing while calculating the similarity degree between the two documents.

Following steps that have been applied to classify documents to one of the predefined class is as follows:

Step 1: Create Bag of words using training data set. And these words will be features on the basis of which documents will be classified.

Step 2: Represent training document and test document as a vector using Eq. (1) by considering the features obtained in step 1.

Step 3: For each of test vector calculate similarity degree with all of the documents of training data set using Eq. (11), Eq. (12), Eq. (13) and Eq. (14).

Step 4: Store the classes of training vector correspond to top 'k' similarity degrees.

Step 5: The test document will be classified to the class which occurs the maximum number of times in the $'k'$ selected classes obtained from step 4.

The working principle of Similarity measures can be easily understood by considering above steps and following figure 2.1.

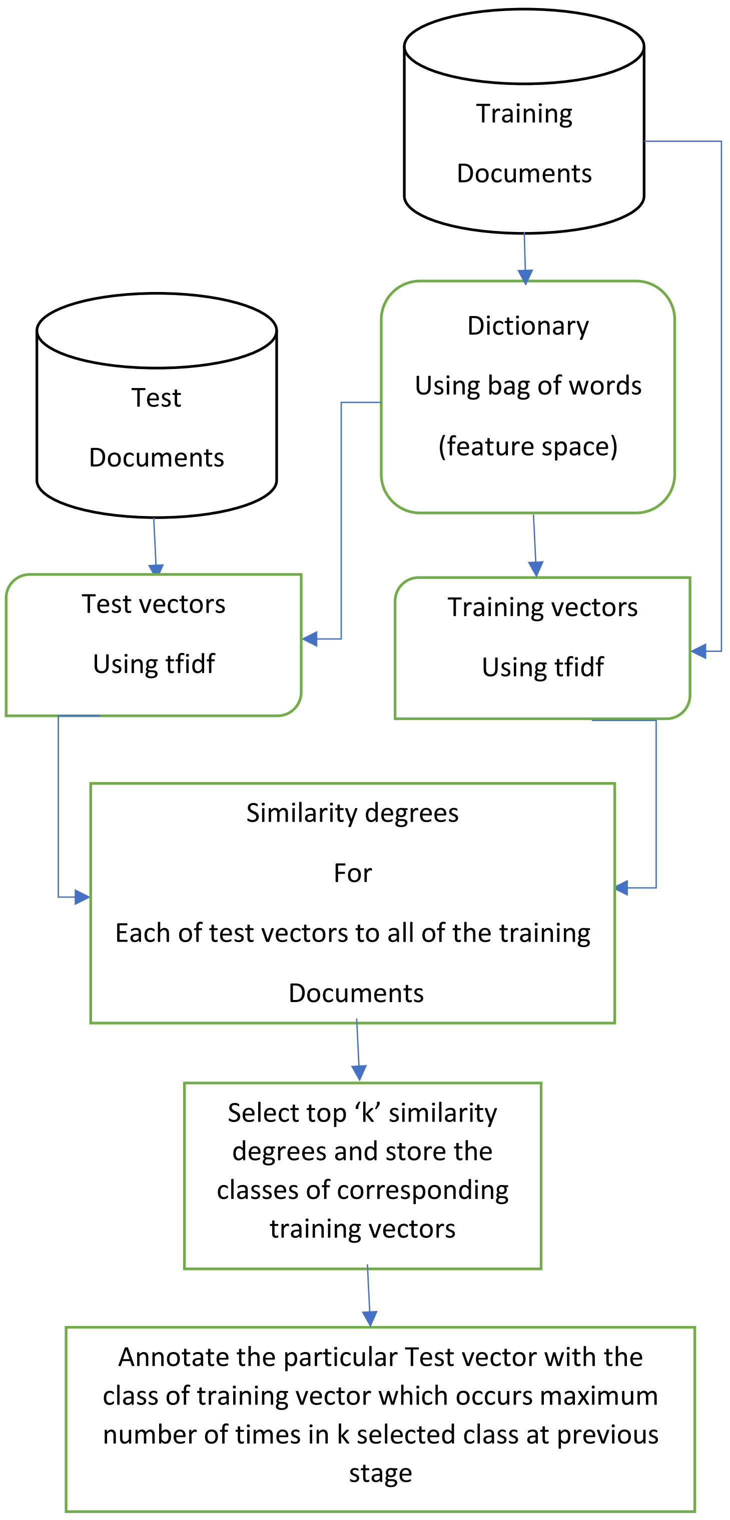
Training
Documents
Dictionary
Using bag of words
(feature space)
Test
Documents
Test vectors
Using tfidf
Training vectors
Using tfidf
Similarity degrees
For
Each of test vectors to all of the training
Documents
Select top 'k' similarity degrees and store the classes of corresponding training vectors
Annotate the particular Test vector with the class of training vector which occurs maximum number of times in k selected class at previous stage

Figure 2.1 Classification using Similarity Measures

2.4 Classification using Bayes Formula and SVM [18]

SVM is classifier that can classify document more accurately than other classifiers. SVM is most accurate classifier because it is capable of classifying document that is independent of the dimensionality of feature space. But the main problem is that when text data is transformed into numeric vectors, using Eq. (5), Eq. (6) and Eq. (7), i.e. tfidf vectors, then dimensions become as large as the number of words (features). Large dimension requires a large amount of training time and a large amount of classification time [10].

To reduce the training time and classification time dimension has to be reduced. Dimensionality can be reduced using Bayes formula [10]. Figure 2.2, depicts the flow of converting training vectors and test vectors using Naïve Bayes vectorizer and then train the SVM classifier using naïve Bayes train vectors. After training, classification of test vector represented using naïve Bayes vectorizer has been done. Here, Naïve Bayes is serving as pre-processor which vectorize the text documents before giving to the SVM classifier for classification.

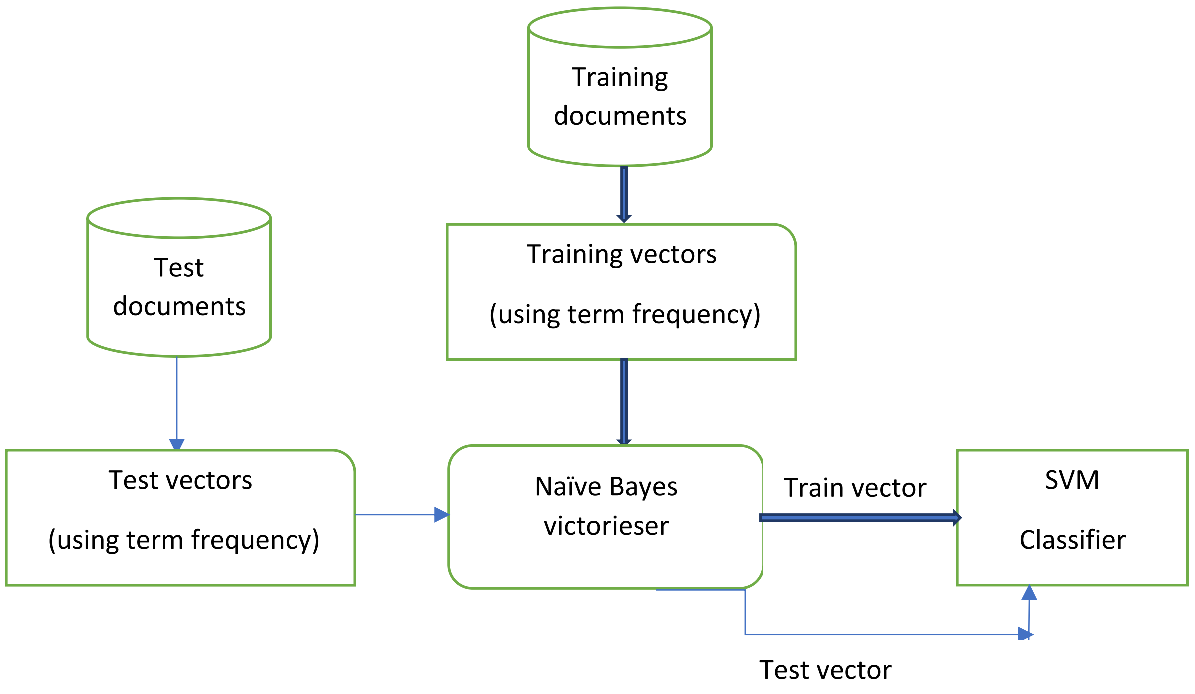

Figure 2.2 Classification using Naïve Bayes and SVM classifier

Following Naïve Bayes formulas are used to create Naïve Bayes train and test vectors:

Let there are $'n'$words in the dictionary, created using training documents, i.e. $w_1, w_2, w_3, \dots \dots \dots, w_n$. And Let there are $'N'$ categories i.e.$c_1, c_2, c_3, \dots \dots \dots \dots \dots \dots, c_N$, and the text document represented with test vector $'X'$ has to be classified to one of the $'N'$ categories

$$P(C_i) = \frac{total\ number\ of\ words\ in\ c_i}{total\ number\ of\ words\ in\ training\ data\ set} \dots \dots \dots \dots \dots \dots \dots \dots \dots (15)$$

$$P(w_j) = \frac{total\ number\ of\ occurance\ of\ w_j in\ training\ data\ set}{total\ number\ of\ occurance\ of\ all\ words\ in\ training\ data\ set} \dots \dots \dots (16)$$

$$P\left({}^{w_j}/_{C_i}\right) = \frac{total\ number\ of\ times\ w_j\ occurs\ in\ C_i}{total\ number\ of\ times\ all\ word\ occur\ in\ C_i} \dots \dots \dots \dots \dots \dots \dots \dots \dots (17)$$

$$P\left({}^{C_i}/_{w_j}\right) = \frac{P\left({}^{w_j}/_{C_i}\right).P(C_i)}{P(w_j)} \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots (18)$$

Steps which is to be followed to classify the documents using this approach is as follows:

Step 1: Create Bag of words by extracting words from training documents.

Step 2: Using Bag of words calculate term frequency for each of the word corresponding to each of text documents.

Step 3: Calculate the probability of particular category $'C'_i$ with respect to particular word $'w_j'$i.e. $'P\left({}^{w_j}/_{C_i}\right)'$as shown in Eq. (17), likelihood of a particular word $'w_j'$ i.e. $'P(w_j)'$ as shown in Eq. (16) and prior probability of a particular category $'C'_i$ i.e. $'P(C_i)'$ as shown in Eq. (15).

Step 4: Calculate posterior probability of a particular word$'w_j'$, being annotated to a particular category$'C'_i$, using Eq. (18).

Step 5: Fill the Matrix of words versus category as shown in table 2.3 with the probability obtained in step 4.

Step 6: Now feed the train documents, obtained from last entry of table 2.3 as probability for input document, to the SVM for training.

Step 7: Follow the same method to victories training documents. Then feed it to SVM for classification.

Table2.3 Probability Matrix of Input Documents

Words	**Probability** C_1	**Probability** C_2	–	–	–	–	–	–	**Probability** C_N
w_1	$P(C_1/w_1)$	$P(C_2/w_1)$	–	–	–	–	–	–	$P(C_N/w_1)$
w_2	$P(C_1/w_2)$	$P(C_2/w_2)$	–	–	–	–	–	–	$P(C_N/w_2)$
–	–	–	–	–	–	–	–	–	–
w_n	$P(C_1/w_n)$	$P(C_2/w_n)$	–	–	–	–	–	–	$P(C_N/w_n)$
Total	$\sum P(C_1/W)$	$\sum P(C_2/W)$	–	–	–	–	–	–	$\sum P(C_N/W)$
Probability of input document	$\frac{\sum P(C_1/W)}{n}$	$\frac{\sum P(C_2/W)}{n}$	–	–	–	–	–	–	$\frac{\sum P(C_N/W)}{n}$

2.5 Document Classification using Class Specific Features and Bayesian Approach

In the previous method, It has been shown that dimensionality can be reduced using Naïve Bayes formula and then classification will be conducted using SVM classifier.

This classification method uses Bayesian approach for classification where features are specific to a particular class. Instead of taking all features at once, it takes the subset of features that are important for a class and then using these specific features apply Bayesian approach for training and classification. Dimensionality reduction can be effectively done using feature selection method in which the only subset of features are taken for classification and remainder are discarded [11]. Bayesian classification formula, for document classification with class specific features, was proposed using Baggenstoss's PDF project theorem [11].

There are three categories of feature selection method as shown in figure 2.3.

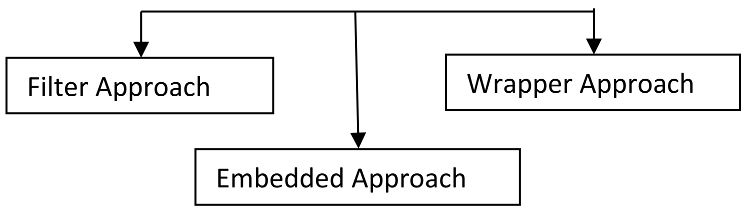

Figure 2.3 Approaches for Feature Selection Method

In filter approach, the importance of each of the feature is calculated and a score is assigned to each of the features without involving any learning criteria. It is the simplest of three and because of its simplicity, most of the feature selection method uses this approach in the field of document classification.

Wrapper approach involves learning criteria for calculating importance of a feature. It requires train classifiers at each step and thus involves higher computation. In spite of better performance, wrapper approach is not used in practice because of its high computation.

Embedded is the combination of both of the approaches: filter approach and wrapper approach.

A document can be represented as a vector of features where features are formed using Bag of words. Features can be real-valued or binary valued. Binary valued features can take a value of either 0 or 1 if the feature is present in the document then the feature value for the document will be 1 otherwise 0.In the real valued model, a feature will have the value as the count of occurrences of that feature in the document. For feature selection, binary valued feature model is preferably used. For classification, real-valued feature model gives the best performance.

Let's consider a binary feature $w_k \epsilon \{0,1\}$ and category C_i, i ϵ {1,2,3…………………N} then some of the popular filter approaches, for feature selection to measure significant of a feature for a particular class in the field of document classification, has been proposed and given as follows:

1) Mutual Information

$$MI(w_k\,, C_i\,) = log\frac{p(w_k\,, C_i)}{p(w_k)p(C_i)} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (19)$$

2) Information Gain (IG)

$$IG(w_k, C_i) = p(w_k, C_i)log\frac{p(w_k\,,C_i)}{p(w_k)p(C_i)} + p(\overline{w}_k, C_i)log\frac{p(\overline{w}_k,C_i)}{p(\overline{w}_k)p(C_i)} \dots\dots\dots\dots\dots\dots\dots (20)$$

3) Relevance Score(RS)

$$RS(w_k, C_i) = log\frac{p(w_k|C_i)}{p(\overline{w}_k|\bar{C}_i)} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (21)$$

4) GSS coefficient (GSS)

$$GSS(w_k, C_i) = p(w_k\,, C_i)p(\overline{w}_k, \bar{C}_i) - p(w_k, \bar{C}_i)p(\overline{w}_k, C_i) \dots\dots\dots\dots\dots\dots\dots\dots (22)$$

5) Chi-square statistic (Chi)

$$Chi(w_k, C_i) = \frac{[p(w_k\,, C_i)p(\overline{w}_k, \bar{C}_i) - p(w_k, \bar{C}_i)p(\overline{w}_k, C_i)]^2}{p(w_k\,, C_i)p(\overline{w}_k, \bar{C}_i)p(w_k, \bar{C}_i)p(\overline{w}_k, C_i)} \dots\dots\dots\dots\dots\dots\dots (23)$$

$\overline{w}_k$: Denote that the feature w_k does not appear in the document

$\bar{C}_i$: Denotes classes other than C_i .

Using one of the above equations class specific features can be selected and then Bayesian approach, based on Baggenston's PPT [12], has been proposed for document classification.

Given a multiple class classification problem, for each of the class $C_i\;, i = \{1,2,3, \dots\dots\dots, N\}$, class-specific feature subset $z_i = f_i\,(w)$, using one of the above equations i.e. Eq.(19), Eq.(20), Eq.(21), Eq.(22) or Eq.(23), can be created that reduces the dimensionality, drastically.

According to Baggenston's PPT, class specific classification formula will be defined.PDF $p(w\ /\ C_i\ )$ will be redefined, from the class-specific feature space into original space when there is a hypothesis reference class C_0 and both the PDFs i.e. $p(w\ /\ C_i\ )$ and $p(z_i\ /\ C_0\ )$ is known, as follows:

$$p(\boldsymbol{w}|C_i\ ) = \frac{p(^{\boldsymbol{w}}/_{C_0})}{p(^{z_i}/_{C_0})}\ p\left(^{z_i}/_{C_i}\right) \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (24)$$

Now Bayesian classification rule, together with redefined PDF as shown in eq.(24), can be defined as follows:

$$C^* = \max_{i\epsilon\{1,2,3,\dots,N\}} log\frac{p(\boldsymbol{w}|C_0)}{p(z_i|C_0)} + \log p(z_i|C_i) + \log p(C_i) \dots\dots\dots\dots\dots (25)$$

When there is training data set and test data set are given for multiclass document classification then the following procedure will be followed for the classification by incorporating class-specific feature subset as:

Step 1: Create feature space using training data set by incorporating Bag of words model. And use Binary valued model to assign feature values.

Step 2: Use one of the above-described equations for calculating the score for the importance of class-specific feature i.e. eq.(19), eq.(20), eq.(21), eq.(22) or eq.(13).

Step 3: Rank the features in descending order based on the score obtained in step 2.

Step 4: Choose indices of top K^{th} features to select only those feature for classification.

Step 5: Now for each of the specific class calculate PDF shown in Eq.(24).

Step 6: Use the Bayesian classification rule described in Eq.(25), to classify the Test document..

Chapter 3

Proposed Method

3.1 Definitions Used in Our Project

1) Elite Keywords: Elite Keywords are those keywords that have high term frequency in each class. This is irrespective of the frequencies of these terms in other classes. To get the high occurring terms in each class, we employ entropy based partitioning technique that is usually used in the field of information theory and coding to generate partition between symbol probabilities. So our method has the advantage as compared to other feature selection schemes that we get the exact subset of significant keywords for each class, and we do not rely on hit and trial methods.

2) Unique Elite Keywords: Unique elite keywords are those that are elite for a particular class and at the same time have higher occurring frequency only in that class. To measure this, we compute the entropy of each elite keyword across all classes, sort the entropies in ascending order and again employ entropy partitioning to shortlist those elite keywords that occur uniquely in this class.

3.2 Overall Methodology for Finding Elite, Unique Elite Keywords [15]

Elite Keywords for a particular class play important role in document classification. Main problem is that given a set of keywords with their probabilities how to partition into elite and non-elite keywords.

Entropy based partitioning is used to partition the keywords between elite and non elite subsets.

Shannon-Fano codes based partitioning is efficient when features are more or less equally likely. It gives best result when sum of probabilities in both partitions are almost equal.

The overall proposed methodology involves following steps to classify documents using elite keywords based on entropy based partitioning:

Step 1: Represent the documents as a vector of terms using Bag of keywords model, where each value corresponds to term frequency (Number of times a term occurs in the document).

Step 2: Find out the probability $\boldsymbol{p}$ for each of term as follows:

$$\boldsymbol{p_i} = \frac{w_i}{\sum_{i=1}^{n} w_i} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (\mathbf{26})$$

Where $\boldsymbol{w_i}$ is term frequency value of $\boldsymbol{i^{th}}$ keyword (term) and there are n number of keywords in the whole corpus.

Step 3: Sort the obtained probabilities in descending order.

Step 4: Find out Partitions $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ using non-extensive entropy partitioning (explained in section 3.2) or Shannon-Fano codes based partitioning.

Step 5: Partition $\boldsymbol{P_1}$ is required partition which indicates the number of keywords to be elite for that category.

Step 6: Take the sorted indices (obtained in step 3) till the point, $\boldsymbol{P_1}$.These indices will be elite keywords for the specific category.

Step 7: Repeat the above steps for all of the categories.

Step 8: Take the tf-idf values for the elite keywords only corresponding to training documents and test documents.

Step 9: Combine elite keywords of all categories and then feed the training vector to one of the classifiers (SVM or TreeBagger)

Step 10: Classify the test documents, using trained SVM or TreeBagger, into one of the categories.

3.3 Entropy Based Partitioning [15]

Partitioning into elite and non elite keywords can be achieved by using maximum entropy based partitioning.

According to non-extensive entropy, to obtain optimal partition, the sum of entropies of two partitions should be maximum such that each of the two partitions contains equally likely features at that point. Maximum entropy based partitioning use the fact that one partition contains the probabilities of the keywords whose sum of the entropy may be 0.9 and other having 0.1. And this situation is helpful in creating a partition between elite and non-elite Keywords.

Step 1: Given a set of keywords with their probabilities, find out probabilities equals to one minus probabilities i.e.$\boldsymbol{p = 1 - p}$.

Step 2: Arrange the probabilities obtained, in step 1, in increasing order.

Step 3: Divide the set of sorted keywords into two probabilities set p1 and p2 temporarily (the iterative testing of all available partitions is done with top single symbol versus remaining symbols in iteration $i = 1$ and ends up with the complete list of symbols versus last symbol in the end most iteration).

Step 4: Obtain the entropy values$H_i(\boldsymbol{p1}_N)$ and $H_i(\boldsymbol{p2}_N)$ corresponding to the probability partitions $\boldsymbol{p1}$ **and** $\boldsymbol{p2}$ for the i^{th} iteration, after normalizing the probabilities to give the total sum as 1 for each partition as given below [reference].

$$p1_N = \frac{p1}{\sum_i p_{1i}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots..\ldots (27)$$

$$p2_N = \frac{p2}{\sum_i p_{2i}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (28)$$

$$H_i(P1_N) = -\sum_{x \in p1_N} xlogx \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots (29)$$

$$H_i(P2_N) = -\sum_{y \in p2_N} ylogy \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots (30)$$

Step 5: Obtain the weighted sum of two entropies $H_i(P1_N)$ and $H_i(P2_N)$ that is the entropy for the i^{th} iteration.

$$H_i(S) = \frac{1}{2}H_i(P1_N) + \frac{1}{2}H_i(P2_N) \ldots\ldots\ldots..(31)$$

Step 6: Repeat step 4-5 for all of the partitions. Find the partition that corresponds to maximum entropy as:

$$m = {}^{max}_{\forall i}\, H_i(S) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.\ldots..(32)$$

Step 7: First partition corresponds to set of elite keywords. Take the sorted indices from partition$P1$.

Step 8: Repeat this for all of the categories.

Chapter 4

Results and Discussion

4.1 Experimental Setup:

System Specification:

1) **Hardware Specification:**

- Ram – 4GB
- Processor - Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz, 2501 Mhz, 2 Core(s), 4 Logical Processor(s)

- System Manufacturer - Dell Inc.
- System Model - Inspiron 5420
- System Type - x64-based PC

2) **Software Specification:**

- Operating System - Microsoft Windows 10 Pro
- Matlab
    - Version – 8.5.0.197613 (R2015a)
    - Release date – February 12, 2015
- Python
    - IDE(Integrated Development Environment) – PyCharm
    - Version – 2016.3.2

4.2 Data Sets

We have used three data sets, WebKb [13], Reuter 8[13], and Reuter 52[13] for the document classification. Table 4.1 shows the important characteristic of these data sets.

Table4.1 Characteristics of Data Sets

Data set	No. of Training Document	No. of Testing Document	No. of Keywords	No. of Categories
WebKb	2785	1383	7289	4
Reuter8	5485	2189	14577	8
Reuter52	6532	2568	16168	52

4.3 Experiment Results

Table 4.2 shows average accuracy, standard deviation, and the number of keywords using different classifiers (SVM and TreeBagger) for the WebKb data set. This Table includes accuracy using Binary classifier and also using multi-class classifier (TBscore based). That gives a clear view about performance using all of the approaches and different classifiers.

Table 4.2 Average Accuracy and Standard Deviation for WebKb using Different Classifiers and Different Approaches

Data set	Method	Average Accuracy (%)	No. of Old Keywords	No. of New keywords
WebKb	Direct + SVM	89.82 ± 4.77	7289	7289
	Direct + Knn (k=15)	71.99 ± 12.70	7289	7289
	Direct + Tree-Bagger (Binary)	92.53 ± 3.59	7289	7289
	Direct + Tree-Bagger (multiclass)	88.43	7289	7289
	Bayes +SVM [10]	87.20 ± 4.29	7289	7289
	SMTP [5]	86.40	7289	7289
	Proposed Method using Tree-Bagger classifier			
	P: Elite (Shannon)	92.67 ± 3.17	7289	4292 [2114,1906,2738,2673]
	P: Unique Elite (Shannon)	93.54 ± 2.89	7289	3024 [1071 995 1399 1386]
	P: Elite (Non -extensive)	92.66 ± 3.58	7289	7129 [3369,3679,5083,4861]
	P: Unique Elite(Non -extensive)	93.40 ± 2.87	7289	4890 [1699 1873 2575 2476]
	1- P : Elite (Shannon)	92.40 ± 3.59	7289	6349 [3645,3645,3645,3645]
	1-P: Unique Elite (Shannon)	93.47 ± 2.60	7289	4412 [1846 1897 1873 1932]
	1-P: elite (Non-extensive)	92.38 ± 3.63	7289	6348 [3645,3645,3644,3645]
	1-P: Unique Elite(Non -extensive)	93.32 ± 2.82	7289	4245 [1840,1856, 1851,1865]
	Proposed Method using SVM classifier			
	P: Elite (Shannon)	90.31 ± 4.19	7289	4292 [2114,1906,2738,2673]
	P: Unique Elite (Shannon)	88.52 ± 5.32	7289	3024 [1071 995 1399 1386]

	P: Elite (Non - extensive)	89.69 $\pm$ 4.90	7289	7129 [3369,3679,5083,4861]
	P: Unique Elite(Non - extensive)	87.27 $\pm$ 4.50	7289	4890 [1699 1873 2575 2476]
	1- P : Elite (Shannon)	89.73 $\pm$ 4.67	7289	6349 [3645,3645,3645,3645]
	1-P: Unique Elite (Shannon)	81.38 $\pm$ 12.2	7289	4412 [1846 1897 1873 1932]
	1-P: elite (Non-extensive)	87.92 $\pm$ 6.49	7289	6348 [3645,3645,3644,3645]
	1-P: Unique Elite(Non - extensive)	86.35 $\pm$ 4.61	7289	4245 [1840,1856, 1851,1865]
	Multiclass Result using TBscore			
	P: Elite (Shannon)	89.80	7289	4292
	P: Unique Elite (Shannon)	90.60	7289	3024
	P: Elite (Individual class) (Shannon)	87.78	7289	
	P: Unique Elite (individual class) (Shannon)	87.99	7289	

Table 4.3 shows Feature Extraction Time, Training Time, and the time for classification of WebKb data set using different classifiers and different approaches.

Table 4.3 Feature Extraction, Training, and Classification Time for WebKb

Data set	**Method**	**T1 (sec)**	**T2 (sec)**	**T3 (sec)**
	Direct + SVM	69	13	3
	Direct + Knn (k=15)	69	50	

WebKb	Direct + Tree-Bagger	69	30	7
	Bayes +SVM	78		
	SMTP	7 days (approx)		
	Proposed Method using Tree-Bagger Classifier			
	P: Elite (Shannon)	141	22	4
	P: Unique Elite (Shannon)	95	18	3
	P: Elite (Non -extensive)	252	31	6
	P: Elite of Elite(Non -extensive)	121	28	5
	1- P : Elite (Shannon)	180	32	5
	1-P: Unique Elite (Shannon)	134	29	4
	1-P: Elite (Non-extensive)	96	30	6
	1-P: Unique Elite(Non -extensive)	173	27	4
	Proposed Method using SVM Classifier			
	P: Elite (Shannon)	141	5	1
	P: Unique Elite (Shannon)	95	4	2
	P: Elite (Non -extensive)	252	6	2
	P: Unique Elite(Non -extensive)	121	4	2
	1- P : Elite (Shannon)	180	4	2
	1-P: Unique Elite (Shannon)	134	5	1

	1-P: Elite (Non-extensive)	96	5	3
	1-P: Unique Elite(Non - extensive)	173	3	2

Table 4.4 gives a concrete view for the performance of different approaches and different classifiers for the Reuter8 data set. It includes average accuracy using Binary classifiers and accuracy using multi-class classifier (based on TBscore)

Table 4.4 Average Accuracy and Standard Deviation for Reuter8 using Different Classifiers and Different Approaches

Data set	**Method**	**Average Accuracy (%)**	**No. of Old Keywords**	**No. of New keywords**
	Direct + SVM	97.88 ± 1.31	14577	14577

Reuter8	Direct + Knn (k=11)	94.32 ± 9.80	14577	14577
	Direct + Tree-Bagger (Binary)	98.36 ± 0.82	14577	14577
	Direct + Tree-Bagger (Multiclass)	95.01	14577	14577
	Bayes +SVM [10]	96.17 ± 2.49	14577	14577
	SMTP [5]	85.50	14577	14577
	Proposed Method using Tree-Bagger classifier			
	P: Elite (Shannon)	98.49 ± 0.83	14577	5207
	P: Unique Elite (Shannon)	98.72 ± 0.73	14577	4358
	P: Elite (Non -extensive)	98.25 ± 0.83	14577	12653
	P: Unique Elite (Non - extensive)	98.64 ± 0.72	14577	9895
	1- P : Elite(Shannon)	98.32 ± 0.88	14577	13920
	1-P: Unique Elite (Shannon)	98.52 ± 0.94	14577	11844
	1-P: Elite (Non-extensive)	98.41 ± 0.83	14577	13921
	1-P: Unique Elite(Non - extensive)	97.32 ± 2.33	14577	8993
	Proposed Method using SVM classifier			
	P: Elite (Shannon)	98.01 ± 1.01	14577	5207
	P: Unique Elite (Shannon)	97.89 ± 1.38	14577	4358
	P: Elite (Non - extensive)	97.92 ± 1.37	14577	12653
	P: Unique Elite (Non - extensive)	97.79 ± 1.30	14577	9895

	1- P : Elite (Shannon)	97.85 ± 1.37	14577	13920
	1-P: Unique Elite (Shannon)	95.80 ± 2.77	14577	11844
	1-P: Elite (Non-extensive)	97.85 ± 1.36	14577	13921
	1-P: Unique Elite (Non -extensive)	92.65 ± 5.07	14577	8993
	Multiclass Result using TBscore			
	P: Elite (Shannon)	96.11	14577	8993
	P: Unique Elite (Shannon)	95.52	14577	8993
	P: Elite (Individual class) (Shannon)	90.80	14577	
	P: Unique Elite (Individual class) (Shannon)	92.33	14577	

Table 4.5 shows different times (Feature extraction time, training time, and classification time) for different approaches and classifiers of Reuter8 dataset

Table 4.5 Feature Extraction, Training, and Classification Time for Ruter8

Data set	**Method**	**T1 (sec)**	**T2 (sec)**	**T3 (sec)**
	Direct + SVM	423	44	8
	Direct + Knn (k=15)	423	227	
	Direct + Tree-Bagger	423	300	39

Reuter8	Bayes +SVM	423	100	10
	SMTP	Some days		
	Proposed Method using Tree-Bagger Classifier			
	P: Elite (Shannon)	400	300	20
	P: Unique Elite (Shannon)	440	228	18
	P: Elite (Non -extensive)	398	279	22
	P: Unique Elite(Non -extensive)	360	266	44
	1- P : Elite (Shannon)	425	223	25
	1-P: Unique Elite (Shannon)	470	287	33
	1-P: Elite (Non-extensive)	455	121	32
	1-P: Unique Elite(Non -extensive)	500	331	23
	Proposed Method using SVM Classifier			
	P: Elite (Shannon)	400	110	10
	P: Unique Elite (Shannon)	440	118	9
	P: Elite (Non -extensive)	398	97	7
	P: Unique Elite(Non -extensive)	360	90	5
	1- P : Elite (Shannon)	425	119	16
	1-P: Unique Elite (Shannon)	470	121	11
	1-P: Elite (Non-extensive)	455	125	10

	1-P: Unique Elite(Non - extensive)	500	126	6

In Table 4.6, Average accuracy, standard deviation and number of keywords for the Reuter 52 data set, has been shown, using different approaches and different classifiers. It shows average accuracy for the binary classifiers and accuracy for the multi-class classifier (Based on TBScore). And in this way, a clear picture of the performance of different approaches and classifier can be obtained.

Table 4.6 Average Accuracy and Standard Deviation for Reuter52 using Different Classifiers and Different Approaches

Data set	**Method**	**Average Accuracy (%)**	**No. of Old Keywords**	**No. of New keywords**
	Direct + SVM	99.33$\pm$ 1.15	16168	16168

	Direct + Knn (k=11)	99.05 ± 3.46	16168	16168
	Direct + Tree-Bagger (Binary)	99.45 ± 0.66	16168	16168
	Direct + Tree-Bagger (Multiclass)	86.05	16168	16168
	Bayes +SVM [10]	98.77 ± 2.40	16168	16168
	SMTP [5]	83.33	16168	16168
	Proposed Method using Tree-Bagger classifier			
	P: Elite (Shannon)	99.49 ± 0.60	16168	6079
	P: Unique Elite (Shannon)	99.50±0.56	16168	5040
Reuter 52	P: Elite (Non -extensive)	99.47 ± .60	16168	13775
	P: Unique Elite(Non - extensive)	99.48 ± 0.60	16168	10958
	1- P : Elite(Shannon)	99.45 ± 0.65	16168	16158
	1-P: Unique Elite (Shannon)	99.47 ± 0.76	16168	13563
	1-P: Elite (Non-extensive)	99.46 ±0.64	16168	16158
	1-P: Unique Elite (Non - extensive)	99.39 ± 1.23	16168	9863
	Proposed Method using SVM classifier			
	P: Elite (Shannon)	99.43 ± 0.98	16168	6079
	P: Unique Elite (Shannon)	93.30± 1.10	16168	5040
	P: Elite (Non - extensive)	99.35 ± 1.08	16168	13775
	P: Unique Elite (Non - extensive)	99.14 ± 1.40	16168	10958

	1- P : Elite(Shannon)	99.34 ± 1.10	16168	16158
	1-P: Unique Elite (Shannon)	96.19 ± 5.00	16168	13563
	1-P: Elite (Non-extensive)	99.34 ± 1.10	16168	16158
	1-P: Unique Elite(Non -extensive)	86.44 ± 9.90	16168	9863
	Multiclass Result using TBscore			
	P: Elite (Shannon)	87.38	16168	6079
	P: Unique Elite (Shannon)	87.81	6168	5040
	P: Elite (Individual class) (Shannon)	90.49	16168	
	P: Unique Elite (individual class) (Shannon)	91.50	16168	

Table 4.7 shows different times for Reuter52 data set using different approaches and different classifiers.

Table 4.7 Feature Extraction, Training, and Classification Time for Ruter52

Data set	**Method**	**T1 (sec)**	**T2 (sec)**	**T3 (sec)**
	Direct + SVM	668	280	21
	Direct + Knn (k=15)	668	270	

Reuter52	Direct + Tree-Bagger	668	450	45
	Bayes +SVM	668	300	22
	SMTP [5]			
	Proposed Method using Tree-Bagger Classifier			
	P: Elite (Shannon)	2500	330	75
	P: Elite of Elite (Shannon)	2500	300	67
	P: Elite (Non -extensive)	2089	400	69
	P: Elite of Elite(Non -extensive)	2767	337	67
	1- P : Elite(Shannon)	2200	421	68
	1-P: Unique Elite (Shannon)	2400	325	89
	1-P: Elite (Non-extensive)	2300	422	77
	1-P: Unique Elite(Non -extensive)	2700	333	69
	Proposed Method using SVM Classifier			
	P: Elite (Shannon)	2640	222	45
	P: Unique Elite (Shannon)	2590	224	40
	P: Elite (Non -extensive)	2089	260	37
	P: Unique Elite(Non -extensive)	2767	202	36
	1- P : Elite (Shannon)	2633	209	54
	1-P: Unique Elite (Shannon)	2777	222	44

	1-P: Elite (Non-extensive)	2888	234	48
	1-P: Unique Elite(Non - extensive)	2099	266	49

4.4 Discussions

In this project, different classifiers (SVM, TreeBagger) and different approaches for Document classification has been used. By involving overall keywords (without selecting significant keywords) classifier low accuracy as compared to when we do the classification using only significant keywords as Shown in above the tables.

We have done experiments with combining the elite keywords of all of the classes and also by taking elite keywords of each class at a time. But the accuracy achieved with combining over all elite keywords is much better than the accuracy achieved without combining elite keywords. We have achieved the best accuracy when we do the classification using elite keywords with

TreeBagger classifier. Elite keywords are selected based on Shannon entropy based partitioning.

For the data set WebKb, as shown in the table 4.2 the best accuracy of 93.54 ± 2.89 is obtained with unique elite keywords (unique elite keywords has been obtained by applying Shannon entropy based partitioning) using TreeBagger classifier which is slightly better than the accuracy of 92.53 ± 3.59 achieved with all keywords using same classifier (TreeBagger). Also the keywords have been reduced to 3024 from 7289 by feature selection method. The No. of New keywords column of the Table 4.2 includes Square brackets [] that indicates Number of elite or unique elite keywords for the individual classes obtained using different entropy based partitioning methods.

As shown in table 4.3, Training time and Testing Time using elite keywords and unique elite keywords is also reduced largely.

For the data set Reuter8, as shown in the table 4.4 the best accuracy of 98.64 ± 0.74 is obtained with unique elite keywords (unique elite keywords has been obtained by applying Shannon entropy based partitioning) using TreeBagger classifier which is slightly better than the accuracy of 98.36 ± 0.89 achieved with all keywords using same classifier (TreeBagger). Also the keywords have been reduced to a greater extent from14577 to 4358 by feature selection method.

As shown in the table 4.5, training time and classification time is also reduced with unique elite keywords.

For the data set Reuter52, as shown in the table 4.6 the best accuracy of 99.50 ± 0.56 is obtained with unique elite keywords (unique elite keywords has been obtained by applying Shannon entropy based partitioning) using TreeBagger classifier which is slightly better than the accuracy of 99.45 ± 0.89 achieved with all keywords using same classifier (TreeBagger). Also the keywords have been reduced to a greater extent from 16168 to 5040 by feature selection method.

As shown in the table 4.5, training time and classification time is also reduced with unique elite keywords.

Chapter 5

Conclusion and Future Scope

5.1 Conclusion

In this project, we have investigated various approaches to select significant keywords for document classification and achieve the better percentage accuracy by incorporating only elite and unique elite keywords (explained in Chapter 3).

With one of two proposed schemes i.e. Shannon based entropy partitioning, all of the benchmarked data set (WebKb, Reuter8, and Ruter52) gives a remarkable result. Comparison with the state-of-the-art methods on benchmark data sets establishes the efficiency of our method from the high percentage accuracy obtained.

5.2 Future Scopes

The entropy based partitioning have a unique property that it gives the exact set of elite or unique elite keywords. This property can be used for a large Data set to discard unwanted keywords without hurting the percentage accuracy. Also, with this property no hit and trial is needed that saves a lot of time as the size of data set increases time becomes important parameter with percentage accuracy. And because of digitization, in the future enormous amount of data will be produced.

So in future, we can apply our method on more variety of data set (like medicine, satellite data etc.).

References

[1] Gupta, Vishal, and Gurpreet S. Lehal. "A survey of text mining techniques and applications." *Journal of emerging technologies in web intelligence* 1, no. 1 (2009): 60-76.

[2] Namburu, Setu Madhavi, Haiying Tu, Jianhui Luo, and Krishna R. Pattipati. "Experiments on supervised learning algorithms for text categorization." In *Aerospace Conference, 2005 IEEE*, pp. 1-8. IEEE, 2005.

[3] https://en.wikipedia.org/wiki/Document_classification

[4] Hotho, Andreas, Andreas Nürnberger, and Gerhard Paaß. "A brief survey of text mining." In *Ldv Forum*, vol. 20, no. 1, pp. 19-62. 2005.

[5] Lin, Yung-Shen, Jung-Yi Jiang, and Shie-Jue Lee. "A similarity measure for text classification and clustering." *IEEE transactions on knowledge and data engineering* 26, no. 7 (2014): 1575-1590.

[6] https://en.wikipedia.org/wiki/Tf%E2%80%93idf

[7] Wang, Xin, Devinder Kumar, Nicolas Thome, Matthieu Cord, and Frederic Precioso. "Recipe recognition with large multimodal food dataset." In *Multimedia & Expo Workshops (ICMEW), 2015 IEEE International Conference on*, pp. 1-6. IEEE, 2015.

[8] Schoenharl, Timothy W., and Greg Madey. "Evaluation of measurement techniques for the validation of agent-based simulations against streaming data." In *International Conference on Computational Science*, pp. 6-15. Springer Berlin Heidelberg, 2008.

[9]D'hondt, Joris, Joris Vertommen, Paul-Armand Verhaegen, Dirk Cattrysse, and Joost R. Duflou. "Pairwise-adaptive dissimilarity measure for document clustering." *Information Sciences* 180, no. 12 (2010): 2341-2358.

[10] Isa, Dino, Lam H. Lee, V. P. Kallimani, and Rajprasad Rajkumar. "Text document preprocessing with the Bayes formula for classification using the support vector machine." *IEEE Transactions on Knowledge and Data engineering* 20, no. 9 (2008): 1264-1272.

[11] Tang, Bo, Haibo He, Paul M. Baggenstoss, and Steven Kay. "A Bayesian classification approach using class-specific features for text categorization." *IEEE Transactions on Knowledge and Data Engineering* 28, no. 6 (2016): 1602-1606.

[12] Baggenstoss, Paul M. "Class-specific feature sets in classification." *IEEE Transactions on Signal Processing* 47, no. 12 (1999): 3428-3432.

[13] http://ana.cachopo.org/datasets-for-single-label-text-categorization

[14] Shannon, Claude E. "A mathematical theory of communication, Part I, Part II." *Bell Syst. Tech. J.* 27 (1948): 623-656.

[15] Susan, Seba, Nandini Aggarwal, Shefali Chand, and Ayush Gupta. "Image coding based on maximum entropy partitioning for identifying improbable intensities related to facial expressions." *Sādhanā* 41, no. 12 (2016): 1393-1406.

[16] Berry Michael, W. "Automatic Discovery of Similar Words." *Survey of Text Mining: Clustering, Classification and Retrieval", Springer Verlag, New York*200 (2004): 24-43.

[17] Salton, Gerard, and Christopher Buckley. "Term-weighting approaches in automatic text retrieval." *Information processing & management* 24, no. 5 (1988): 513-523.

[18] Lam, Wai, Kon Fan Low, and Chao Yang Ho. "Using a Bayesian network induction approach for text categorization." In *IJCAI (1)*, pp. 745-750. 1997.

[19] Sebastiani, Fabrizio. "Machine learning in automated text categorization." *ACM computing surveys (CSUR)* 34, no. 1 (2002): 1-47.

[20] Susan, Seba, and Gitin Kakkar. "Decoding facial expressions using a new normalized similarity index." In *India Conference (INDICON), 2015 Annual IEEE*, pp. 1-6. IEEE, 2015.