A Hierarchical Text Categorization Algorithm Based on Feature Selection

A dissertation submitted in partial fulfilment of the requirements of
M.Tech (Computer Science)
degree of Indian Statistical Institute, Kolkata.

By

Durgesh Singh

Under the supervision of

Dr. Swapan K. Parui

Computer Vision and Pattern Recognition Unit

Indian Statistical Institute, Kolkata
203, Barrackpore Trunk Road
Kolkata-700 108
India
July 12, 2015

To my parents and grandmother

In memory of my grandfather.

Ceritificate of Approval

This is to certify that this thesis titled **"A Hierarchical Text Categorization Algorithm Based on Feature Selection"** submitted by Durgesh Singh, Roll Number CS1314, Indian Statistical Institute Kolkata, in partial fulfillment for the award of the degree of Master of Technology is a bonafide record of work carried out by him under my supervision.

Dr. Swapan K. Parui
Comuter Vision and Pattern Recognition Unit,
Indian Statistical Institute,
Kolkata

Abstract

We are working extensively on the Reuters-21578 dataset which is a multi-label dataset. It works well with the multi-label classification techniques. When we use conventional multi-class classifier, the dataset has a few sets of classes which are highly overlapping which causes a lot of misclassification of documents for these overlapping classes.

We are trying to improve the results for the conventional multi-class classification techniques by understanding those features which are going to discriminate appropriately among overlapping classes.

We have used 2 levels of features selection techniques. At the first level we perform Okapi BM25 $tf * idf$, mutual information and $tf * df\_normalized$ (Chapter 4) based features selection methods.

At second level we have tried to further distinguish a particular set of classes labeled corn, wheat and grain represented as class 2, 5 and 10 respectively in our thesis. This set contains overlapping classes which accounts for a very high misclassfication. We have studied here the ways to select viable features which will distinguish between the overlapping classes. This section includes bigrams and unigrams using certain criteria and the $tf * df\_normalized$ approach which was mentioned for first level feature selection. We have also discussed one more feature selection method which is described in feature selection section in Chapter 6.

We have built a hierarchical classifier, where at first level we are using Self Organizing Map (SOM) algorithm on the training set to get the similar documents grouped together. Then we are using our feature selection techniques to get the most viable features set and we perform Naive Bayes at each cluster of the SOM algorithm. Here the training set is the documents at each of the clusters.

Acknowledgment

At the end of this course, it is my pleasure to thank everyone who has helped me along the way.

First of all, I would like to express my sincere gratitude to my supervisor, Prof. Swapan K. Parui for introducing me to the area of Text Classification and showing me the immense application oriented to this field. I have learnt a lot from him. For his patience, for all his advice and encouragement and for the way he helped me to think about problems with a broader perspective, I will always be grateful.

I would like to thank all the professors at ISI Kolkata who have made my educational life exciting and helped me to gain a better outlook on Computer Science. I would also like to express my gratitude to Prof. Mandar Mitra, Prof. Utpal Garain, Prof. Debapriyo Majumdar for interesting discussions. I would like to thank Prof. Sandip Das, Prof. Arijit Bishnu, Prof. Sasthi C. Ghosh, Prof Ansuman Banerjee and all other under whom I got the opportunity to learn new things.

I would like to thank everybody at ISI for providing me a nice platform for pursuing my studies. I thank all my classmates who have made the academic and non-academic experience very delightful. Special thanks to all of my friends Amit-da, Ayan-da, Kripa-da, Raghu-da, Tamal-da, Anshuman, Archan, Arindam, Ashwin, Chirag, Dnyaneshwar, Kushal, Mayur, Harmender, Ranjan, Ravinder, Rishabh, Shubhadeep and many others who made my campus life so enjoyable. It has been great to have them around at all times.

My most important acknowledgement goes to my family and friends who have filled my life with happiness, most significantly, to my parents and brother and sister who have always supported and encouraged me to pursue my dreams. A special thanks to Ridhi who have been a very good friend who motivated me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Text Categorization

The digitization of information has been increasing exponentially as the expansion of internet has prevailed. The text in digital form is growing in every possible domains and sectors like finance, banking, service, health, government data and all types of statistical data in all these areas. For an enterprise, automated text categorization is important because of the large quantity of documents that need to be properly processed and classified. An online newspaper/magazine might want to automatically classify incoming articles under various topics or class labels like agriculture, economy and sport.

Text categorization, or text classification (TC) is the task of automatically classifying a set of documents with unknown labels into various categories from a given set of labels by going through the contents of the text. TC model uses the contents of given documents to learn some useful information from them which is used in classification process for a new document.

Yet another application of text classification is text filtering. In filtering,

the newspaper could classify the upcoming news as relevant and irrelevant based on their topics. For example, a news-site concerning about sports might want to consider only the articles which are relevant to sports and block all the other articles as irrelevant. A text filter could also categorize incoming e-mail as normal and junk mail.

In order to deal with this huge amount of digital information the increasing demand of automated text categorization can be best understood and it has become one of the most popular and researched problems in information retrieval and machine learning. The challenges which are posed to researcher are given the vast dimensions of the text. Documents can be of type : plain text, semi-structured text and structured text, in this thesis we are dealing with the plain texts. Challenges basically arise due to the size of text to be classified, the nature of text, the datastructure we are using to represent the text, by the amount of size of training set that we have for learning the classifier.

Text categorization/Text Classification is of two kind: *unsupervised* and *supervised* text categorization.

- *Unsupervised Text categorization*: This type of TC assumes no prior information of the labels of the documents. There is no human expert who has assigned documents to classes. It is the distribution and makeup of data that will define cluster membership [8].
- *Supervised Text categorization*: In this technique, the documents are well labelled to classes and on the basis of this information new documents are classified to predefined classes. So here we have the supervisor or human who defines the classes and labels training documents [8].

As mentioned the text is of high dimension which form the feature space,

lots of features are not of importance to the text categorization algorithms as the contribution of such features are not large percentage and affects the text categorization very minimal. So we can exclude such features which do not capture more information and we can find out those features which captures the most information. There are several known techniques for feature selection and we have developed our own too.

We have used a dataset which is multilabel. It has a subset of classes which are overlapping and can be classified with multilabel classification. This gives good results. We experimented with conventional multi-class classification on this data set. We have built a hierarchical classifier where at first level Self Organizing Map (SOM), a clustering algorithm is applied and then we apply Naive Bayes at each clusters of SOM. We did extensive experiments Naive Bayes as well as k-NN rule for various feature selection techniques. We first obtained the features from the training set and then tested on the training set to see the performance of our algorithm and then we experimented on the test set.

1.2 Dataset

Datasets are collections of pre-classified documents. They are essential to develop and evaluate a text classification (TC) system. They are used to train the system which will be used later on to classify the test documents.

For the first step of training phase we know all the labels in the training documents. These training documents are used to learn some important information and in turn train the TC system. We then enter in testing phase where we input the unlabeled test documents of test dataset to the already built and trained TC system.

The quality of the TC system is decided by the fact that how well it classifies

the unlabeled test documents.

The Reuters-21578 Collection

All the documents contained in the Reuters-21578 collection[1] appeared on the Reuters newswire and were manually classified by personnel from Reuters Ltd. This collection is very skewed, with documents very unevenly distributed among different classes.

We worked rigorously on the ModApte split of Reuters-21578 dataset and noted the result under various conditions. This is a multi-label dataset where each test document can be classified to multiple labels.

We tried to make the conventional classification for this dataset much better using various features selection techniques which we will mention in later chapters in detail.

Text Representation:

Each document is represented as vectors of words which is done in popular vector representation for information retrieval Salton and McGill [12].

Reuters-21578 (ModApte Split)

We used 9980 stories which has been classified into 10 classes. The stories are about 200 words in length. The Mod Apt train/test split is generally used for classification tasks, in which 7193 stories are used to train the classifier and rest 2787 stories are used as the test set. The stories are temporal in nature, i.e. the training set occur before the test set in time. Number of documents in test set (defined in Notations chapter) for the top 10 categories are given below in Table 1.1.

[1] `https://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection`

Table 1.1: Number of Documents in each 10 classes

Class Label	#Documents
acq	719
corn	56
crude	189
earn	1087
grain	149
interest	131
money-fx	179
ship	89
trade	118
wheat	71

We have defined the following notations for these class labels as shown in Table 1.2. We have used these notations in rest of the thesis report.

Table 1.2: Class Names for each Class Label in top 10 category of Reuters-21578 that we are using in thesis

Label	acq	corn	crude	earn	grain	interest	money-fx	ship	trade	wheat
Name	class1	class2	class3	class4	class5	class6	class7	class8	class9	class10

Evaluation Metric

We evaluated our results using the Accuracy as the metric. Accuracy is defined as the percentage of correctly classified documents. Accuracy is a real value between 0 and 1.

$$Accuracy = \frac{Number\ of\ correctly\ classified\ documents}{Total\ number\ of\ documents}$$

We are using percentage of this accuracy by multiplying with 100.

1.3 Related Work

The area of text classification has seen a lot of research due to the increase in demand of digitized and well-defined information. Various TC concepts, applications, dimensionality reduction, classifier types and the way to evaluate

them is very elegantly described in [14].

There has been a lot of experimental work done on the popular Reuters collection. Here we are using Reuters-21578 top 10 classes ModApte split. Yang [15] performed comparative evaluation of various TC methods. The result of his evaluation were published on the Reuters corpus. The work of Dumais et al. [2], Yang et al. [16], and Erkan et al. [3] further describes the Reuters-21578 data set. Various TC methods were compared in Dumais et al. [2]; they used mutual information (MI) to select most important features. MI is described in Yang et al. [17]. They also described other feature selection methods such as document frequency thresholding, Information gain, chi-square statistics and term strength. Dasgupta et al. [1] discusses about various feature selection strategies and gave an unsupervised feature selection method. We get an idea to choose an important term based on the term frequency and the document frequency from Salton et al. [13]. Salton and Buckley [11] discuses about the $tf * idf$ term weight measure.

SOM is studied in Saarikoski et al. [10] and in books [5]. Eyheramendy et al. [4] discusses about using various probability fuctions for Naive Bayes, it has been studied in Lewis [7] and can be found in few books mentioned at [8], [9]. Naive Bayes is used in many other research papers: Dumais et al. [2], [6], Yang and Liu [16]. k-NN algorithms are studied and work has been done in Yang and Liu [16].

1.4 Organization of the Dissertation

We would like to divide our work in two levels.

1. Clustering (unsupervised classification)

2. Supervised classification

(a) Naive Bayes

(b) k-NN rule

Chapter 2 is about the preliminaries an notations used in this thesis. It includes all the technical little information related to TC and technical terms that builds the basic structure of the thesis.

Chapter 3 talks about the clustering where the implementation of self organizing map algorithm is done. It describes about the feature selection that we used, the results and observations from the result and finally conclusion is shown which will be helpful in further 2^{nd} level of supervised classification in our TC model.

From these results, we can observe that these are the overlapping classes sets: *class* $\{2, 5, 10\}$, *class* $\{6, 7\}$ and *class* $\{3, 8\}$

We can think them as the documents belonging to these three sets of classes are highly similar to one another. The documents of each of these sets may get wrongly classified to one of the other classes from the same set. We are emphasizing that the probability of misclassification is high among the documents of these sets. This is the reason that a particular cluster might not be pure and have documents belonging to above mentioned subsets. This experimentation is concluded in this chapter.

Chapter 4 talks about a probabilistic classifier: Naive Bayes classifier. We gave an introduction to the algorithm and several probabilistic distribution functions like multinomial, Poisson and an empirical based probability distribution of terms is done. We used several feature selection methods to get the most important features out of the whole feature set.

We did two levels of classification here to try to reduce the misclassification rate which is caused by the classes 2, 5 and 10. At second level we choose

bigrams features that are contributing most to the documents. The process of choosing the bigrams is explained in this section.

We sorted the bigrams features in decreasing order of the appropriate score to experiment the variations in the result and we got notable differences in the results at the second level of Naive Bayes classifier. We also merge the bigrams with unigrams chosen using the same approach we applied to choose bigrams.

Chapter 5 discusses the k-NN method, it starts with the introduction and then describes the feature selection methods we used contrasts the results on the basis of choosing two similarity functions : *cosine similarity* and *euclidean distance.*

Chapter 6 introduces the main task of this thesis, the hierarchical algorithm using feature selection, on the first level of which, run the SOM algorithm and where we get the clusters from training set. At the second level we perform Naive Bayes classifier. We train naive Bayes classifier at each cluster level which we got from the SOM algorithm output.

We show the results on the basis of two possible selection of similarity measures, we are using those features selection methods which were experimented in Chapter 3.

In this chapter we incorporate all the research work done in previous chapters to get some useful result out of our classifier.

Chapter 7 shows the conclusion and future work.

Chapter 2

Preliminary and Notations

In this section we have given few definitions and notations of terms which are involved in this thesis. These terms serves as the basis for describing the key idea behind each concept which is building block to several other mathematical notations and terminology used in this thesis.

We are given,

1. Training set:

   (a) A training set is a set of documents which is applied in a classifier to learn some useful information from it, which is used to classify new documents on the basis of that learned information. A collection of documents $X = \{\vec{X_1}, \vec{X_2}, \vec{X_3}, \cdots, \vec{X_N}\}$ is the training set. Size of this training set is considered to be N.

   (b) $\vec{X_i} \in X, \quad \forall i = \{1, 2, \cdots, N\}$ is a document from the training set X which is represented in vector space: $\vec{X_i} = \{w_{i,1}, w_{i,2}, \cdots, w_{i,|V|}\}$ where $w_{i,j}$ is the weight associated for the j^{th} term of the document $\vec{X_i}$ $\forall i \in [1, N]$ and $\forall j \in [1, |V|]$, weight $w_{i,j}$ is a positive real number. $|V|$ is the number of unique terms in all the docu-

ments. A term is basically defined according to the application where classifier is used, it can be a single word (*unigrams*), group of two words (*bigrams*), group of several words (*phrases*). We use $\langle t_j \rangle$ to denote the j^{th} term. Corresponding to each term in each document $\vec{X_i}$ $\forall i \in [1, N]$ the associated weight $w_{i,j}$ is a positive real number and we find out the weights using below mentioned ways:

i. *Term frequency:* For each i^{th} document, weight of the j^{th} term is considered as frequency of occurrence of this term in that document, which is denoted as $f(\langle t_j \rangle, \vec{X_i})$, where $\langle t_j \rangle$ denotes the j^{th} term.

ii. $Tf * idf$*:* It is product of the term frequency $f(\langle t_j \rangle, \vec{X_i})$, as described above and *inverse document frequency (idf$_j$)* which is defined for a j^{th} term:
$idf_j = \log_{10} \frac{N}{df_j}$
Hence, $tf * idf = f(\langle t_j \rangle, \vec{X_i}) * idf_j$
Where, df_j is the document frequency of the j^{th} term of the set X, irrespective of a particular document. It is a measure of how much information the word provides, that is, whether the term is common or rare across all documents. It gives the count of number of documents in which this term occurs. N is same as the size of training set as defined above, which is basically total number of documents in the training set X.

iii. *Okapi* $BM25$ $tf * idf$: We have used this as an alternative for the weights of terms of the documents. This is calculated for a j^{th} term of i^{th} document X_i as:
$(BM25\ tf * idf)_{i,j} = idf_j * \frac{f(\langle t_j \rangle, \vec{X_i}) * (k_1 + 1)}{f(\langle t_j \rangle, \vec{X_i}) + k_1 * (1 - b + b * \frac{|\vec{X_i}|}{avg_{dl}})}$
$idf_j = \log \frac{N - df_j + 0.5}{df_j + 0.5}$

$|\vec{X_i}|$ is the length of the document $\vec{X_i}$, k_1 and b are constants, avg_{dl} is the average length of all the documents in the set X, df_j is the document frequency of term $\langle t_j \rangle$ which is the number of documents in which this term is present.

(c) Vector space Model: It encodes the documents and queries into vectors, which makes the processing of these documents easier. The matching of documents and queries is made using distance or similarity calculations between the documents which are represented as vectors in $|V|$ dimensional space.

(d) The dimension of the document vector is called the *Vocabulary* of the document set. It is a set of unique words in the document set. For the training set of document we denote it by V. The *Vocabulary* size for training set is denoted as $|V|$.

2. Test Set:

Test set $T = \{\vec{T_1}, \vec{T_2}, \cdots, \vec{T_M}\}$ is also a set of documents defined in same way as training set. Where $\vec{T_i}$ $\forall i = \{1, 2, \cdots, M\}$ is a test document in the test set which are not known to the classifier beforehand. It is task of the classifier to classify the test documents by using the information learned from the training set.

Test documents are represented as vectors : $\vec{T_i} = \{w_{i,1}, w_{i,2}, \cdots, t_{i,|p|}\}$, $\forall i \in [1, M]$ where M is size of test set, p is the vocabulary set of test set, and $|p|$ is the vocabulary size. Rest notations are similar to training set, except the term weight/metric $tf * idf$, since test documents are coming one by one we do not have measure to find out idf for a term belong to test set. One alternative is to use term frequency of j^{th} term from the test document $\vec{T_i}$ and idf of that term from the training set.

3. A set of classes C=$\{c_1, c_2, \cdots, c_{10}\}$:

   Class c_i, $\forall i = \{1, 2, \cdots, 10\}$ of a document is basically the label or the category in which a given documents belongs. The class for training set documents is know beforehand, and class of the test document has to be estimated using the text classifiers system. In our thesis we have top 10 classes for the Reuters-21578 dataset on which we are testing our classifier algorithm. Hence,class size is $|C|$= 10. We are using the notations for each class label as shown in Table 1.2.

   No of documents in each class c_i from training set is assumed to be n_{c_i}. We often use $TF_{j,i}$ to represent collective frequency of j^{th} term in class c_i.

4. Clusters: These are the output of a clustering/unsupervised classification algorithm.We run the algorithm on a training set and we get those documents in one cluster which are most similar to each other on some similarity measure.

5. Similarity functions: To measure the closeness of one document with another document we must have some similarity measure on the documents

   (a) Cosine similarity

   This concept applies to documents in their vector space model representation. This measures the similarity between documents vectors $\vec{X_i}$ and $\vec{T_j}$ by finding the cos theta of angle formed between them in $|V|$ dimensional space. If this value is near to 1 then the two documents are similar, when this value is 0 then two documents are orthogonal and not related to each other, similarly when this value is negative then also, the documents are very dissimilar to each other.

cos-similarity$(\vec{X_i}, \vec{T_j})$=$\frac{\vec{X_i} * \vec{T_j}}{|\vec{X_i}| * |\vec{T_j}|}$

Where $\vec{X_i} \in X$ and $\vec{T_j} \in T$. $|\vec{X_i}|$ and $|\vec{T_j}|$ are the absolute value of the vector $\vec{X_i}$ and $\vec{T_j}$ respectively.

(b) Euclidean Distance

The Euclidean distance between points p and q is the length of the line segment connecting them. We also use (L_2) norm as its notation. In Cartesian coordinates, if $p = (p_1, p_2, \cdots, p_{|V|})$ and $q = (q_1, q_2, \cdots, q_{|V|})$ are two points in Euclidean $|V|$-space, then the distance (d) from p to q, or from q to p is given by:

$d(p, q) = \sqrt{\sum_{i=1}^{|V|}(q_i - p_i)^2}$

We also represent this distance using this notation that we are going to use in this thesis: $d(p, q) = ||p - q||$.

We choose that documents from the set of training documents closest to the given test document, for which we have the minimum distance of the test document from the training documents.

6. M-class classifier: this is a kind of classifier that assigns only one of the possible classes to the test document, e.g. if $C = \{c_1, c_2, c_3\}$ then a test document $\vec{T_1}$ can only be classified to only one of either c_1, c_2 or c_3.

7. M-label classifier: this is a kind of classifier that assigns one or many of the possible classes to the test document, e.g. if $C = \{c_1, c_2, c_3\}$ then a test document $\vec{T_1}$ can only be classified to one or many of the classes which are subsets of C.

8. Feature selection: TC suffers from the drawback of high dimension of feature space. Feature space is basically the set of terms which are represented as vectors in a document vector representation defined

earlier. There are some terms which are not important and their inclusion does not give much impact to the TC systems so we want to neglect such terms on the basis on some ranking of the terms which gives the importance of terms. There are various measures used in our thesis to select appropriate features that reduces the feature space to the most important terms and improves the TC system. We discuss this in details in upcoming chapters.

9. Probabilistic classifier: The use of probabilities to predict a class is evident in these types of classifiers. They are based on Bayes rule:

   $P(A|B) = \frac{P(B|A).P(A)}{P(B)}$

   where A and B are events.

   $P(A)$ and $P(B)$ are the probabilities of A and B without regard to each other. $P(A|B)$, a conditional probability, is the probability of A given that B is true. $P(B|A)$, is the probability of B given that A is true.

   Given an evidence in terms of training documents and prior probability of training documents we try to figure out the probability of the test document.One important and widely researched probabilistic classifier is Naive-Bayes classifier.

10. k-Nearest Neighbour (k-NN) classifiers: For each test document,we select k neighbourhood documents based on some similarity measure to find the similar training documents in its neighbourhood which was described above. Among the k nearest neighbours which ever class label has got majority votes, that label is assigned to the test document.

Chapter 3

Self Organizing Map

3.1 Introduction

Self organizing map (SOM) [5], also known as Kohonen map, or self organizing feature map, is an artificial neural network using unsupervised learning to cluster data samples. SOM inspired from the fact that how neurons interact with one another. SOM are used widely in data clustering and visualization tasks, as well as in classification. We are trying to take help of this clustering algorithm: self organizing maps, to find out how effective it is in text classification tasks? We are applying it as the first level of our hierarchical classifier, where we get clusters of similar training documents. This will be useful at second levels of our main hierarchical algorithm. Here we are using this algorithm to try to distinguish between the overlapping classes of the given dataset .

SOM algorithm

Our first algorithm is self organizing map, which is an unsupervised classification algorithm. Following are the steps to the algorithm, we have used the notations as described in the previous chapter.

Steps :

From the training set X, we randomly choose $m << N$ number of documents, which we call as set of weights vectors and denote them as: $W = \{\vec{W}_1, \vec{W}_2, \cdots, \vec{W}_m\}$, where $\vec{W}_i \in X$ and $\vec{W}_i = \{w_{i,1}, w_{i,2}, w_{i,3}, \cdots, w_{i,|V|}\}$, $\forall i \in [1, m]$. Where $w_{i,j}$, as defined in previous chapter is the metric for j^{th} term in the i^{th} weight vector.

Let us denote the randomly chosen weights by $\vec{W}_{1(old)}, \vec{W}_{2(old)}, \cdots, \vec{W}_{m(old)}$.

We have α_1 and α_2 which are initialized to 0.02 and 0.01 respectively. There is a factor m^f which is initialized to 1.

1. Start:

   We run a for loop for 500 times and do following steps for each of the loop: m^f reduces to 0.9 of its initial value at the 200^{th} iteration of the loop, at each increment of loop by 100, it is reduced by further 0.9 of its previous value then successively. We multiply the factor m^f with α_1 and α_2 in each loop.

   (a) for each $\vec{X}_i$ $\forall i \in [1, N]$ do following:

      i. Update rule:
      find $\vec{W}_{j1} \in W$ that is nearest to $\vec{X}_i \in X$.
      find $\vec{W}_{j2} \in W$ that is second nearest to $\vec{X}_i \in X$.
      $\vec{W}'_{j1} = \vec{W}_{j1} + \{m^f * \alpha_1\}[\vec{X}_i - \vec{W}_{j1}] = [1 - \{m^f * \alpha_1\}]\vec{W}_{j1} + \{m^f * \alpha_1\}\vec{X}_i$.
      $\vec{W}'_{j2} = \vec{W}_{j2} + \{m^f * \alpha_2\}[\vec{X}_i - \vec{W}_{j2}] = [1 - \{m^f * \alpha_2\}]\vec{W}_{j2} + \{m^f * \alpha_2\}\vec{X}_i$.

   After this modification step we get a new set of weights denoted as: $\vec{W}_{1(new)}, \vec{W}_{2(new)}, \cdots, \vec{W}_{m(new)}$.

   We now proceed to below mentioned terminating case.

(b) Terminating case:

i. If $||\vec{W}_{j(old)} - \vec{W}_{j(new)}|| < \epsilon,\ \forall j \in [1, m]$ then exit out of the loop and go to step 2. Left hand side of the inequality measures the distance of old weight with respect to the newly modified weight using euclidean distance. Here ϵ is pre-determined constant, its value is chosen as 0.05.

ii. Otherwise, let $\vec{W}_{j(old)} = \vec{W}_{j(new)} \forall j \in [1, m]$. Repeat step 1.

2. Forming of clusters:

Algorithm is stopped either by running all the loop iterations or by the given terminating conditions, At the termination of above steps, we get a set of weights, which we consider as stable weights and we denote them as: $\{\vec{W}_1^*, \vec{W}_2^*, \vec{W}_3^*, \cdots, \vec{W}_m^*\}$.

Now, we will find out the most similar cluster of each of the documents.

(a) If the similarity measure is Euclidean distance then the formula to get clusters is:
$CL_j = \{\vec{X}_i \in X : ||\vec{X}_i - \vec{W}_j^*|| \leq ||\vec{X}_i - \vec{W}_k^*||\ \ \forall\, k \in [1, m]\}\ and\ \forall j \in [1, m]\ and\ \forall i \in [1, N]$, N is total number of training set documents. CL_j are the clusters formed in the SOM algorithm.

(b) If similarity function is cosine similarity then we measure the most similar cluster as:
$CL_j = \{\vec{X}_i \in X : \text{cos-similarity}(\vec{X}_i, \vec{W}_j^*) \geq \text{cos-similarity}(\vec{X}_i, \vec{W}_k^*)$ $\forall\, k \in [1, m]\}\ and\ \forall j \in [1, m]\ and\ \forall i \in [1, N]$, N is total number of training set documents. CL_j are the clusters formed in the SOM algorithm.

We are finding the most similar stable weight for a document X_i using the above two mentioned similarity functions measure. We get our cluster set as:

$S' = \{CL_1, CL_2, CL_3, \cdots, CL_m\}$.

S' is set of clusters CL_j $\forall j = \{1, 2, \cdots, m\}$.

3.2 Feature Selection

In the training set applied document threshold frequency, where we deleted all those terms that appear in more than $N/2$ documents, where N is the total number of training documents.

We used Okapi $BM25\_tfidf$ and term frequency as the metric of the features.

3.3 Training and Results

We are running the SOM algorithm on the Reuters training set of size 7193.

We trained the clustering algorithm for the four cases:

1. Measure for similarity of Nearest weight for a document in Update step and closest cluster per document in cluster forming step is cosine similarity

2. Measure for similarity of Nearest weight for a document in Update step and closest cluster per document in cluster forming step is Euclidean distance

We are showing one of the results of the SOM algorithm which was run for 500 iterations. Here we showed the output of Okapi $BM25$ $tf * idf$, we have

also performed it for the term frequencies. This output of SOM and the stable weights that we get at the end of the SOM algorithm, will be used as the first level of our hierarchical categorization algorithm.

Table 3.1: Results on SOM

Class labels→	1	2	3	4	5	6	7	8	9	10
cluster1	134	0	1	12	0	1	1	0	0	0
cluster2	101	0	3	96	0	2	0	0	22	0
cluster3	4	0	95	2	0	3	0	1	0	0
cluster4	16	0	0	14	0	13	0	0	0	0
cluster5	0	6	0	0	26	0	0	2	0	18
cluster6	0	0	0	209	0	0	0	0	0	0
cluster7	0	0	0	3	0	0	0	0	0	0
cluster8	0	0	0	0	0	30	43	0	0	0
cluster9	0	0	0	132	0	0	0	0	0	0
cluster10	0	0	0	94	0	0	0	0	0	0
cluster11	75	0	0	0	0	0	0	0	0	0
cluster12	0	4	0	0	30	1	0	1	0	22
cluster13	9	0	0	39	0	0	0	0	0	0
cluster14	0	0	0	259	0	0	0	0	0	0
cluster15	9	0	0	14	0	0	0	0	0	0
cluster16	2	12	0	1	14	0	0	0	2	0
cluster17	2	3	5	0	9	6	23	4	95	3
cluster18	3	0	1	0	0	0	0	5	0	0
cluster19	0	0	0	0	0	12	37	0	0	0
cluster20	91	0	1	11	1	0	0	0	0	1
cluster21	0	9	0	0	32	0	0	1	2	20

Continued on next page

Table3.1 – continued from previous page

Class labels→	1	2	3	4	5	6	7	8	9	10
cluster22	90	0	0	0	0	0	0	0	0	0
cluster23	6	8	0	4	13	0	0	0	0	8
cluster24	60	0	0	6	0	0	0	0	0	0
cluster25	0	0	0	0	0	0	0	0	1	0
cluster26	56	0	0	16	0	0	1	3	1	0
cluster27	35	0	3	211	2	14	5	2	0	1
cluster28	12	0	1	9	0	0	0	2	0	0
cluster29	49	0	0	0	0	0	0	0	0	0
cluster30	0	0	0	143	0	0	0	0	0	0
cluster31	7	0	0	15	0	0	0	0	0	0
cluster32	0	0	12	0	1	0	0	24	0	1
cluster33	0	15	0	0	24	0	0	0	0	15
cluster34	0	0	0	38	0	0	0	0	0	0
cluster35	20	0	0	16	0	1	0	0	0	0
cluster36	1	0	0	85	0	0	0	0	0	0
cluster37	0	0	0	15	0	11	12	0	3	0
cluster38	11	0	0	1	0	0	0	1	0	0
cluster39	1	0	0	25	0	0	0	0	0	0
cluster40	0	0	0	0	0	7	20	0	0	0
cluster41	22	0	0	37	0	2	0	0	1	0
cluster42	0	0	0	0	0	5	7	0	0	0
cluster43	0	2	2	0	13	0	0	12	0	1
cluster44	7	0	0	22	0	0	0	0	0	0
cluster45	16	0	2	4	2	20	37	0	6	1
cluster46	9	0	0	60	0	0	0	0	0	0

Continued on next page

Table3.1 – continued from previous page

Class labels→	1	2	3	4	5	6	7	8	9	10
cluster47	7	0	0	19	0	0	0	4	0	0
cluster48	0	22	0	0	43	0	0	8	0	31
cluster49	2	0	1	126	0	0	0	0	0	0
cluster50	9	0	5	42	0	0	0	0	0	0
cluster51	25	0	0	41	0	0	0	0	0	0
cluster52	0	0	0	0	0	0	2	0	1	0
cluster53	137	0	0	2	0	0	1	0	0	0
cluster54	18	0	69	11	0	0	0	0	0	0
cluster55	0	0	1	0	1	42	39	1	0	1
cluster56	0	0	0	176	0	0	0	0	0	0
cluster57	22	0	0	8	0	0	0	0	0	0
cluster58	0	0	0	46	0	0	0	0	0	0
cluster59	2	0	0	4	0	0	0	0	0	0
cluster60	0	19	0	0	49	0	0	1	2	17
cluster61	6	0	0	0	1	0	0	0	89	0
cluster62	0	19	0	0	19	0	0	0	0	7
cluster63	5	0	0	9	0	0	0	0	0	0
cluster64	19	2	0	4	3	2	8	0	0	1
cluster65	0	0	0	159	0	0	0	0	0	0
cluster66	68	0	0	0	0	0	0	1	0	0
cluster67	2	0	0	1	2	0	0	11	0	0
cluster68	0	0	21	0	0	0	0	27	0	0
cluster69	1	0	0	41	0	0	0	1	0	0
cluster70	0	0	0	0	0	15	19	0	0	0
cluster71	4	0	0	4	0	0	0	0	0	0

Continued on next page

Table3.1 – continued from previous page

Class labels→	1	2	3	4	5	6	7	8	9	10
cluster72	11	4	0	31	4	1	0	0	0	0
cluster73	5	0	81	1	0	0	0	0	0	0
cluster74	20	5	7	0	18	14	53	8	82	5
cluster75	0	0	0	0	7	0	0	32	0	0
cluster76	4	0	3	1	0	0	0	0	0	0
cluster77	63	0	7	38	0	1	0	1	2	0
cluster78	16	0	1	51	0	0	0	0	1	0
cluster79	0	0	0	0	0	36	36	0	0	0
cluster80	3	0	0	6	0	0	0	1	2	0
cluster81	18	0	0	41	0	9	9	0	3	0
cluster82	0	0	0	169	0	0	0	0	0	0
cluster83	0	0	43	0	0	0	0	0	0	0
cluster84	146	0	0	4	0	0	0	0	0	0
cluster85	0	36	0	0	66	0	0	3	0	22
cluster86	0	0	2	0	0	12	92	0	30	0
cluster87	5	0	0	54	0	1	0	1	0	0
cluster88	18	0	1	11	0	0	1	0	4	0
cluster89	1	3	0	0	4	0	1	1	0	0
cluster90	5	0	0	1	0	0	0	0	0	0
cluster91	0	0	0	0	0	60	2	0	0	0
cluster92	6	0	1	1	0	0	0	0	0	0
cluster93	1	0	1	6	0	4	68	3	2	0
cluster94	0	12	1	0	48	0	0	2	9	36
cluster95	0	0	14	0	1	0	0	31	5	1
cluster96	0	0	2	0	0	22	21	0	4	0

Continued on next page

Table3.1 – continued from previous page

Class labels→	1	2	3	4	5	6	7	8	9	10
cluster97	1	0	0	102	0	0	0	0	0	0
cluster98	76	0	0	35	0	0	0	0	0	0
cluster99	9	0	2	38	0	0	0	1	0	0
cluster100	67	0	0	1	0	0	0	1	0	0

Correlation between classes

We found out from the results of the above algorithm, the correlation between various classes and tabulated the result. We can figure out that few classes are very overlapping and take part in misclassification.

Table 3.2: Results for Correlation on class pairs for SOM output

Class	2	3	4	5	6	7	8	9	10
1	0.6628	0.8904	0.6405	0.8767	0.7599	0.7855	0.9229	0.7649	0.8737
2		0.6962	0.3773	0.9189	0.5403	0.5232	0.6620	0.6592	0.8812
3			0.4427	0.8550	0.6831	0.6979	0.9252	0.8390	0.8550
4				0.5114	0.4594	0.4887	0.4471	0.3756	0.4900
5					0.7499	0.7351	0.8264	0.8274	0.9840
6						0.7989	0.6344	0.7965	0.7344
7							0.6241	0.7840	0.7415
8								0.7262	0.8186
9									0.8430

An association between classes from another angle

After finding out the correlation between the classes we tried to found out the association between various classes (total 45 associations for 10 classes) from another angle to emphasize and support the results that we got from

the covariance output as shown above. It also shows which classes are overlapping. For a higher value the association between classes is high, and hence overlap is high. The method shown below to calculate the association:

$$assoc(c_i, c_j) = \frac{2 * S_{i,j}}{n_i + n_j} \quad \forall c_i, c_j \in C, \ \forall i, j \in [1, 10]$$

$$S_{i,j} = \sum_{k=1}^{|CL|} (min(n_{i,k}, n_{j,k})) \quad where \ |CL| = 100$$

$$n_i = \sum_{k=1}^{|CL|} n_{i,k} \quad \forall i \in [1, 10]$$

Where, $assoc(c_i, c_j) \in [0, 1]$, $n_{i,k}$ and $n_{j,k}$ is number of documents which belong to class c_i and c_j $\forall c_i, c_j \in C, \ \forall i, j \in [1, 10]$ respectively which gets classified to a particular cluster CL_k : $\forall CL_k \in CL, \forall k \in [1, 100]$ CL is the set of clusters which we get as output from the SOM algorithm, $|CL|$ is the size of total number of clusters which is 100 in our implementation and $min(n_i, n_j)$ is a function to find minimum between n_i, n_j and n_i, n_j is the number of documents that get clustered in the class c_i and c_j.

Table 3.3: Results for Association on class pairs for SOM output

	2	3	4	5	6	7	8	9	10
1	0.0218	0.0863	0.1965	0.0355	0.0701	0.0639	0.0389	0.0703	0.0150
2		0.0631	0.0078	0.5895	0.0568	0.0417	0.1640	0.1090	0.6819
3			0.0440	0.1313	0.1250	0.1057	0.2184	0.1715	0.0632
4				0.0169	0.03598	0.0298	0.0234	0.0301	0.0084
5					0.0871	0.0823	0.1968	0.1521	0.6573
6						0.5627	0.0772	0.2150	0.0536
7							0.0653	0.2491	0.0346
8								0.0848	0.1515
9									0.0860

3.3.1 Conclusion:

We conclude from the result that the classes which are overlapping, are as follows:

Observation from the covariance

Class 1, Class 3 : 0.890493

Class 1, Class 5 : 0.876720

Class 1, Class 8 : 0.922923

Class 1, Class 10 : 0.873736

Class 2, Class 5 : 0.918989

Class 2, Class 10 : 0.881240

Class 3, Class 5 : 0.855086

Class 5, Class 10 : 0.984013

Observation from the association formula

Class 2, Class 5 : 0.589577

Class 2, Class 10 : 0.681934

Class 5, Class 10 : 0.657364

Class 6, Class 7 : 0.562712

So these mentioned classes are highly going to participate in high misclassification in our classification algorithm. We would like to narrow them down with the help of appropriate feature selection algorithm.

We formed five groups from above information and named them:

$G1 = \{2, 5, 10\}$,

$G2 = \{6, 7, 9\}$,

$G3 = \{3, 8\}$,

$G4 = \{4\}$,

$G5 = \{1\}$

The number in these sets belong to a class number, hence $G1$ consists of class 2, 5 and 10. These groups are formed on the basis of results of the fore coming supervised classification chapters, these groups are used in second level of classification in Naive Bayes classifier which is the topic of next chapter.

Chapter 4

Naive Bayes

4.1 Introduction

Naive Bayes [8], [9] is Bayesian or probabilistic classifiers. It uses the joint probabilities of words and classes to estimate the probabilities of each class given a document. Given a set of r document vectors $D = \{\vec{d_1}, \vec{d_2}, \cdots, \vec{d_r}\}$, classified along a set C of s classes, $C = c_1, c_2, \cdots, c_s$. Bayesian classifiers estimate the probabilities of each class c_i given a document d_j as:

$$P(c_i|\vec{d_j}) \propto P(c_i) * P(\vec{d_j}|c_i)$$

Where,

Given the document d_j, $P(c_i|d_j)$ is the probability of it belongs to class c_i.

$P(c_i)$ is the prior class probability of documents occurring in class c_i. If a documents terms do not provide clear evidence for one class versus another, we choose the one that has a higher prior probability.

$P(d_j|c_i)$ is the conditional probability of document d_j belonging to class c_i. It is interpreted as a measure of how much evidence d_j contributes that c_i

is the correct class.

Naive Bayes assumes that the probability of a given term is independent of other terms that appear in the same document. Though this assumptions makes the classifier named Naive, but the comparable results of its performance are surprising.

If $|V|$ is the vocabulary size. We get,

$$P(\vec{d_j}|c_i) = P(\langle t_1\rangle|c_i) * P(\langle t_2\rangle|\langle t_1\rangle * c_i)*, \cdots, *P(\langle t_{|V|}\rangle|\langle t_{|V|-1}\rangle*, \cdots, *\langle t_1\rangle * c_i) \tag{4.1}$$

$$P(\vec{d_j}|c_i) = P(\langle t_1\rangle|c_i) * P(\langle t_2\rangle|c_i)*, \cdots, *P(\langle t_{|V|}\rangle|c_i)$$

$$P(\vec{d_j}|c_i) = \prod_{k=1}^{|V|} P(\langle t_k\rangle|c_i)$$

The term independence assumption simplifies the determination of $P(\vec{d_j}|c_i)$ as the product of the probabilities of each term that appears in the document. Here, $\langle t_k\rangle$ denotes the k^{th} term and $P(\langle t_k\rangle|c_i)$ is its conditional probability of number of its occurrence in class c_i.

We find the best class for the test document. The best class in Naive Bayes classification is the most likely or *maximum a posteriori (MAP) class*

$$c_{map} = argmax_{c_i \in C} P^*(c_i|d_j) \tag{4.2}$$

$$= argmax_{c_i \in C} P^*(c_i) * P^*(d_j|c_i)\,. \tag{4.3}$$

In above equation, many conditional probabilities are multiplied, one for each position $k \in [1, |V|]$ and this can cause in a floating point underflow. Hence we perform the computation by adding logarithms of probabilities instead of multiplying probabilities. The class with the highest log probability score is still the most probable as the logarithm function is monotonic. So

we get log of right hand side of above equation in order to get :

$$c_{map} = argmax_{c_i \in C}[\log P^*(c_i) + \sum_{k=1}^{|V|} \log P^*(\langle t_k \rangle | c_i) \, \forall i \in [1, 10] \,]. \quad (4.4)$$

We are denoting $P(\langle t_k \rangle | c_i)$ as defined earlier as $P^*(\langle t_k \rangle | c_i)$ because we are using the training set to estimate $P(\langle t_k \rangle | c_i)$. Hence, $P(c_i)$ and $P^*(\langle t_k \rangle | c_i)$ as above, are estimated from the training set as explained in below sections in details.

$$P(c_i) = \frac{n_{c_i}}{N} \;\; \forall c_i \in C, \; \forall i \in [1, 10]. \quad (4.5)$$

where n_{c_i} is the number of documents in class c_i and N is the total number of documents.

We estimate the conditional probability $P^*(\langle t_k \rangle | c_i)$ as according to probability distribution functions mentioned below.

4.1.1 Multinomial Naive Bayes

For each class we sum up the term frequency for each term and calculate the probability of occurrence of that term according to this formula:

$$P^*(\langle t_k \rangle | c_i) = \frac{TF_{k,i}}{\sum_{l=0}^{|V|} TF_{l,i}} \; \forall c_i \in C, \; \forall i \in [1, 10] \; and \; \forall k \in [1, |V|]$$

where, $TF_{k,i}$ is the number of occurrence of k^{th} term $\langle t_k \rangle$, $\forall k \in [1, |V|]$ in documents belonging to class $c_i \; \in C, \; \forall i \in [1, 10]$. Denominator indicates the sum of frequencies of all the terms in class c_i.

To avoid the terms which have zero term frequency we use the Laplace

smoothing as follows:

$$P^*(\langle t_k \rangle | c_i) = \frac{TF_{k,i} + 1}{\sum_{l=0}^{|F|} TF_{l,i} + |F|} \quad \forall c_i \in C \quad \forall i \in 1, 10 \; and \; \forall k \in [1, |F|] \quad (4.6)$$

$|F|$ is the size of the SELECTED_FEATURES set F which is selected during the feature selection step, where $F \subseteq V$.

4.1.2 Poisson Naive Bayes

We are using Poisson probability distribution to calculate the probability $P(d_t|c_i)$. Which is defined as :

$$P(\vec{d_t}|c_i) = \prod_{j=1}^{|V|} (\exp^{-\lambda_{i,j}} * \frac{\lambda_{i,j}^{f_j}}{f_j!}) \quad \forall c_i \in C \quad and \quad \forall i \in [1, 10] \quad (4.7)$$

Where, $\vec{d_t}$ is the test document, $\lambda_{i,j}$ is mean of the term j in class i, f_j is the frequency count of term j and $f_j!$ is factorial of the term frequency of j^{th} term. Since $f_j!$ is common for the given test document when measured across all the classes, we can neglect it and our new equation becomes:

$$P(\vec{d_t}|c_i) = \prod_{j=1}^{|V|} (\exp^{-\lambda_{i,j}} * \lambda_{i,j}^{f_j}) \quad \forall c_i \in C \quad and \quad \forall i \in [1, 10] \quad (4.8)$$

where $\lambda_{i,j}$ can be calculated as :

$$\lambda_{i,j} = \frac{\sum_{k=0}^{n_{c_i}} TF_{k,j,i}}{n_{c_i}} \quad \forall \; c_i \in C \quad and \quad \forall \; j \in [1, |V|] \quad (4.9)$$

where,$\lambda_{i,j}$ is mean of term t_j, $TF_{j,i}$ is the term frequency of the term t_j, $\forall j \in [1, |V|]$ and n_{c_i} is the no of documents in class c_i. We are using Laplace smoothing when the $\lambda_{i,j} = 0$, as shown:

$$\lambda_{i,j} = \frac{\sum_{k=0}^{n_{c_i}} TF_{k,j,i} + 1}{n_{c_i} + |F|} \quad \forall \; c_i \in C \quad and \quad \forall \; j \in [1, |F|] \quad (4.10)$$

$|F|$ is the size of the SELECTED_FEATURES set F which is selected during feature selection step, where $F \subseteq V$.

4.1.3 Empirical Prob. Distribution of Terms

Here we are defining the probability of each term using the data from the training set as follow:

1. We are calculating the highest term frequency from the training set documents and then we calculate $table\_size$ which is an integral value as,

   $= ceil(\frac{log_2(1+max\_tf)}{INTERVAL\_SIZE})$

   where, $ceil(x)$ is the function that return an integer greater than or equal to x, max_tf is the maximum term frequency in over all training set. $INTERVAL\_SIZE$ is the size of the interval which is predetermined as 0.25, $table\_size$ gives the total number of intervals corresponding to each term of the documents.

2. Now once we get the maximum table size we are going to create a datastruture which has for each class and each document, $table\_size$ number of entries for these many interval for each terms.We initialize them to be zero.

3. We get the index for the k^{th} term with term frequency tf_k as : $table\_index = \frac{log_2(1+tf_k)}{INTERVAL\_SIZE}$ we increment the location indexed by $table\_index$ position by one.

4. We calculate the probability of the k^{th} term using $table\_index$ as follows:

$$P(\langle t_k \rangle | c_i) = \frac{freq(table\_index)_{t_k}}{\sum_{l=0}^{table\_size} freq(table\_index)_{t_l}} \quad \forall c_i \in C \tag{4.11}$$

where, $\langle t_k \rangle$ is notation for the k^{th} term, $freq(table\_index)_{t_k}$ is the count at index $table\_index$ of the term t_k in the given class $c_i \in C$. $|F|$ is the size of the SELECTED__FEATURES set F, which is selected during feature selection step, where $F \subseteq V$.

5. We are using Laplace transform to avoid terms having zero occurrence to avoid zero overall probability, $\forall c_i \in C$

$$P(t_k|c_i) = \frac{freq(table\_index)_{t_k} + 1}{\sum_{l=0}^{table\_size} freq(table\_index)_{t_l} + table\_size} \ \forall k \in [1, |F|] \tag{4.12}$$

4.2 Feature Selection

4.2.1 First Level

We are neglecting those terms which occur in more than $N/2$ documents, where N is the size of training set X.

Okapi BM25 Tf*idf

From the training set, we get the Okapi $BM25tf * idf$ values and sum them up for each terms corresponding to a particular class $c_i \in C$, tf corresponds to term frequency, which gives importance to a term in a document and idf is inverse document frequency which takes care of the term's relevance across the documents. We sort them in decreasing order class wise and then choose f_i set features for each class $c_i \in C \forall i \in [1, 10]$.

$BM25tf * idf$ for a term $\langle t_j \rangle$, $\forall j \in [1, |V|]$ is given as:

$$(BM25\ tf * idf)_{i,j} = idf_j * \frac{f(\langle t_j \rangle, \vec{X}_i) * (k_1 + 1)}{f(\langle t_j \rangle, \vec{X}_i) + k_1 * (1 - b + b * \frac{|\vec{X}_i|}{avg_{dl}})}$$

$idf_j = \log \frac{N - df_j + 0.5}{df_j + 0.5}$

where, $\vec{X_i} \in X$ is the document being considered. $|\vec{X_i}|$ is the length of the document $\vec{X_i}$. k_1 and b are constants and avg_{dl} is the average length of all the documents. $f(\langle t_j \rangle, \vec{X_i})$ is the frequency of term $\langle t_j \rangle$ in document $\vec{X_i}$. N is the total number of training documents, df_j is the total number of documents in which term $\langle t_j \rangle$ is present.

Mutual Information (MI)

Mutual information, $MI(x_j, c_i)$ between a feature(term), x_j and a class $c_i \in C$ is defined as:

$$MI(x_j, c_i) = \sum_{x_j \in \{0,1\}} \sum_{c_i \in \{0,1\}} P(x_j, c_i) \log \frac{P(x_j, c_i)}{P(x_j)P(c_i)} \tag{4.13}$$

let us assume term $\langle t_j \rangle$ is now represented as $x_j \ \forall j \in [1, |V|]$.

Here we are using binary features i.e. either a term occurs or does not occurs in a document. We find out $MI(x_j, c_i)$ foreach x_j and c_i, we sort them in decreasing order class wise, then for each class $c_i \in C, \ \forall i \in [1, 10]$ we choose a set of features f_i, whose size is $|f|$. We get final selected features $F = \bigcup_{i=1}^{10} f_i$. where, $|F|$ is the size of final selected feature set. We ran the test for various values of f_i, which is tabulated in the result section.

Tf*df_normalized

For each class c_i where $c_i \subset C$ and $\forall i \in [1, 10]$, we find out for each term $\langle t_j \rangle, \ \forall j \in [1, |V|]$:

$$\Delta_{i,j} = tf_{i,j} * df_{i,j}\_normalized \tag{4.14}$$

Where,

$$df_{i,j}\_normalized = \frac{\sum_{i=0}^{|C|} df_{i,j}\_mean}{|C|} \quad where,\ |C|= 10$$

$$df_{i,j}\_mean = df_{i,j}/n_{c_i}$$

where $df_{i,j}$ is document frequency of term $\langle t_j \rangle$ belonging to class c_i, $tf_{i,j}$ is term frequency of term $\langle t_j \rangle$ belonging to class c_i, $tf_{i,j} = \sum_{k=1}^{n_{c_i}} (f(\langle t_j \rangle, \vec{X_k}))$ where, for each of i^{th} class c_i we have, $f(\langle t_j \rangle, \vec{X_k})$ is the frequency of j^{th} term in k^{th} document, $k = \{1, 2, \cdots, n_{c_i}\}$.

n_{c_i} is number of documents contained in class c_i.

Using this score we find out variance of each term with respect to each class and we sort these variance values in decreasing order. We select top $|F|$ number of features based on those having top variance in the sorted list, these features comprises the required feature set F. Using this set F, we perform the Naive Bayes classification algorithm.

4.2.2 Second Level

Looking at the previous sections results and the high correlation between few classes, it can be observed that the misclassification rate is higher in these groups: G1,G2. We particularly look and try to do our experimentation for the group $G1 = \{class\ 2, class\ 5, class\ 10\}$. We are performing this second level for these 3 classes only, to try reducing the misclassification.

Bigrams

With single word features in the first level we have restricted details of the behaviour of the features, so to identify more properties of these features we

choose to get the position independent bigrams from the input vector space of the documents. This approach enable us to identify few more properties and relation among features and hence to discriminate between them.

We did following to choose the appropriate discriminating bigrams:

1. Let us get the bigrams set of each of the classes c_i where $i \in \{2, 5, 10\}$ and name them as $b_j^i \ \forall i \in \{2, 5, 10\}$, so we get, b_j^2, b_k^5, b_l^{10} where $j \in [1, |bigrams\ of\ class2|]$, $k \in [1, |bigrams\ of\ class5|]$, and $l \in [1, |bigrams\ of\ class10|]$ respectively, we take union of all of these bigrams and call them:

$$b_u, \quad u \in \{j \cup k \cup l\}$$

2. Get the mean document frequency for each bigrams from classes: $2, 5, 10$ from the training set. Denote them as $dm_i, \ \forall i \in \{2, 5, 10\}$, so we get, $\{dm_2, dm_5, dm_{10}\}$.

3. Get the absolute difference of the maximum and minimum among $dm_2, \ dm_5$, and dm_{10} for each of the bigrams as:

   $b_g = (\max(dm_2, dm_5, dm_{10}) - \min(dm_2, dm_5, dm_{10})), \ \forall g \in u$

4. Sort with respect to $b_g, \ \forall g \in u$. We call it b_g^{sorted}.

5. In the sorted list b_g^{sorted}, we assign the feature $b_{g,i}^{sorted}$ to class c_i when $\max(dm_2, dm_5, dm_{10})$ belongs to i^{th} class where $i \in \{2, 5, 10\}$, this feature is the discriminating feature of the class c_i.

6. Now we select top $f_i' \ \forall i \in \{2, 5, 10\}$ features set of size $|f_i'|$, which have features from each of the class 2, 5 and 10 and we take union of them $F_b' = \{f_2' \cup f_5' \cup f_{10}'\}$ of size $|F_b'|$.

We choose $|f_i'|$ number of bigrams from each of the class 2, 5 and 10. We ran the test for various values of $|f'|$ that can be seen in the result section.

Union of unigrams (chosen using special criteria) and bigrams

Here unigrams are selected on some criteria. We call them as unigrams only now onwards, until unless stated specifically for some other meaning. Selection of Bigrams is same as stated in above section.

Bigrams selected feature set: F_b'.

For unigrams we do same as above for bigrams but this time we choose only with unigrams, which is explained below:

1. Let us get the unigrams set of each of the classes c_i where $i \in \{2, 5, 10\}$ and name them as $un_j^i \ \forall i \in \{2, 5, 10\}$, so we get, $un_j^2, un_k^5, un_l^{10}$ where $j \in [1, |unigrams\ of\ class2|]$, $k \in [1, |unigrams\ of\ class5|]$, $l \in [1, |unigrams\ of\ class10|]$ respectively, we take union of all of these unigrams and call them:

$$un_{u'}, \quad u' \in \{j \cup k \cup l\}$$

2. Get the mean document frequency for each unigrams from classes: $2, 5, 10$ from the training set. Denote them as $dm_i \ \forall i \in \{2, 5, 10\}$, so we get, $\{dm_2, dm_5, dm_{10}\}$.

3. Get the absolute difference of the maximum and minimum among dm_2, dm_5, and dm_{10} for each of the unigrams as:

   $un_g = (\max(dm_2, dm_5, dm_{10}) - \min(dm_2, dm_5, dm_{10})) \forall g \in u$

4. Sort with respect to $un_g \ \forall g \in u$. We call it un_g^{sorted}.

5. In the sorted list un_g^{sorted}, we assign the feature $un_{g,i}^{sorted}$ to class c_i when $\max(dm_2, dm_5, dm_{10})$ belongs to i^{th} class where $i \in \{2, 5, 10\}$, this feature is the discriminating feature of that class i.

6. Now we select top $f_i' \; \forall i \in \{2, 5, 10\}$ features set of size $|f_i'|$, which have features from each of the class 2, 5 and 10. We take union of these to get, $F_u' = \{f_2' \cup f_5' \cup f_{10}'\}$ of size $|F_u'|$.

Now new feature set is $F_{new} = F_b' \bigcup F_u'$

Our Score Variance for class 2,5,10

Here we choose unigrams based on the same approach as mentioned in the feature selection in first level. But we did here only for classes 2, 5 and 10.

We choose top best $|F|$ features in overall, where F is the feature set as defined earlier, but here it consists of features belonging to class 2, 5 and 10 only.

4.3 Training and Results

4.3.1 First Level

In the first level of Naive Bayes, we are training over the 7193 ModApte split of Reuters-21578. The vocabulary size $|V| = 13455$. We are finding the features for each class from the whole training set.

F is the new features space which we got from the training set and we test the test-documents using this new feature space. As mentioned in feature selection section, in first level we are taking out top $|f_i|$ number of features for each $c_i \in C$ features for MI and BM25 $tf * idf$. For $tf * df\_normalized$ feature selection strategy we are taking top features set F across all the classes.

We are applying Laplace transforms to avoid zero probabilities in case a term frequency happens to be zero.

We are including prior probability as defined in introduction section of this chapter.

Test set is of size 2787 docs. Test documents comes one by one and they are tested on the new feature space of size $|F|$ which we selected from the training set.

Okapi BM25 TF*IDF

We are selecting features set f_i per class on the basis of top most Okapi BM25 $tf * idf$ values from each of the classes. We are performing Naive Bayes on the selected feature.

Below is the result for this feature set run on three probability distributions:

Table 4.1: Results on top Okapi BM25 $tf * idf$ selected features

pdf → feature size per class↓	MN	PO	Empirical
10	83.81	81.98	66.02
15	85.36	83.17	68.03
20	85.71	83.45	69.46
25	85.79	83.56	70.14
30	85.57	84.21	71.61
35	86.25	84.67	72.44
40	86.54	84.93	72.65
50	86.47	84.93	72.65
100	86.97	85.82	75.24
150	87.01	85.46	75.67
300	86.97	85.14	76.13
1345	87.33	83.60	76.53

We have shown the two confusion matrices, one for the best separation between the classes 2, 5 and 10 in Table 4.2. Second one in Table 4.3 is for the best accuracy case. For the first table we can see that the accuracy for class 2 is 53.57%, class 5 is 40.26% and class 10 is 46.47%. It reduces for class 2 and 10 when we look at the second confusion matrix which is of higher

overall accuracy for a higher feature set. Although the overall accuracy of second confusion matrix is not significantly greater than the accuracy of the first confusion matrix

Table 4.2: Confusion Matrix for best case for Okapi BM25 $tf * idf$ feature selection method, where we selected 50 features per class with accuracy 86.47

Class	1	2	3	4	5	6	7	8	9	10
1	694	0	11	7	1	0	1	2	3	0
2	0	30	0	0	20	0	0	0	2	4
3	4	0	139	0	0	0	1	43	2	0
4	22	0	7	1054	0	1	3	0	0	0
5	0	34	0	0	60	0	0	9	10	36
6	1	0	0	0	1	102	26	0	1	0
7	5	0	0	0	0	48	118	0	8	0
8	0	0	9	0	0	0	0	76	3	1
9	0	0	2	0	2	1	6	0	104	2
10	0	11	0	0	23	0	0	3	1	33

Table 4.3: Confusion Matrix for Okapi BM25 $tf * idf$ feature selection for the highest accuracy 87.33

Class	1	2	3	4	5	6	7	8	9	10
1	698	0	7	10	0	0	1	0	3	0
2	0	4	0	0	50	0	0	0	2	0
3	3	0	163	0	1	0	0	20	2	0
4	20	0	3	1061	0	1	0	0	2	0
5	0	4	0	0	119	0	1	8	11	6
6	0	0	0	0	1	82	47	0	1	0
7	0	0	1	1	0	29	143	0	5	0
8	3	0	23	0	2	0	0	55	6	0
9	0	0	2	0	3	0	9	0	103	0
10	0	2	0	0	59	0	0	3	1	6

MI:

Here we are using binary features i.e. either a term occurs or does not occurs in a document. From the training set, we get the term frequency values and sum them up for each terms corresponding to a particular class $c_i \in C$.

For a given term $\langle t_j \rangle \quad \forall j \in [1, |V|]$, we calculate Mutual Information (MI):

$$MI(\langle t_j \rangle, c_i) = \frac{n11}{n} * \log \frac{n * n11}{n1p * np1} + \frac{n01}{n} * \log \frac{n * n01}{n0p * np1} + \frac{n10}{n} * \log \frac{n * n10}{n1p * np0} + \frac{n00}{n} * \log \frac{n * n00}{n0p * np0}$$

n11=no of documents in which term $\langle t_j \rangle$ is present and they belong to class c_i

n01=no of documents in which term $\langle t_j \rangle$ is not present and documents belongs to class c_i

n10=no of documents in which term $\langle t_j \rangle$ is present and they do not belong to class c_i.

n00=no of documents in which term $\langle t_j \rangle$ is not present and they do not belong to the class c_i.

n1p=n11+n10 : no. of docs which contains term $\langle t_j \rangle$ irrespective of class c_i.

np1=n11+n01 : no. of docs which are in class c_i.

n0p=n01+n00 ; no. of docs which do not contains term $\langle t_j \rangle$ irrespective of class c_i.

np0=n10+n00 : no. of docs which do not belong to class c_i.

We have shown the two confusion matrices, one for the best separation between the classes 2, 5 and 10 in Table 4.5. Second confusion matrix in Table 4.6 is for the best accuracy case .

The overlapping of classes 2, 5 and 10 is very high for this dataset. For the first matrix, we can analyze that the separation between the classes 2 and 10 is very good, with 92.85% and 54.92% respectively. But for class 5, the accuracy is not that good. These features are able to distinguish between class 2 and 10 very effectively.

For the second matrix, it can be seen that the accuracy is highest for 1345

Table 4.4: Results on top MI selected features

pdf → feature size per class↓	MN	PO	Empirical
10	82.95	82.13	65.08
15	83.17	83.27	66.95
20	84.78	83.92	69.46
25	85.10	84.42	70.43
30	84.85	84.64	70.82
35	84.89	84.28	71.43
40	85.03	84.39	72.12
50	85.46	84.89	72.90
100	86.22	85.28	74.99
150	86.50	84.96	75.45
300	86.61	84.17	75.85
1345	86.76	80.94	76.31

features per class. We can conclude that as the features size increases, the accuracy increases but the effect of features to distinguish between overlapping class decreases as can be seen in the second matrix.

Table 4.5: Confusion Matrix for best case for MI feature selection method, where we selected 10 features per class with accuracy 82.95

Class	1	2	3	4	5	6	7	8	9	10
1	598	0	19	78	2	7	6	2	7	0
2	0	52	0	0	2	0	0	0	0	2
3	1	1	154	1	1	1	1	28	1	0
4	15	0	3	1065	0	3	1	0	0	0
5	0	64	0	0	29	0	0	2	10	44
6	1	1	0	1	0	103	25	0	0	0
7	3	0	0	5	0	53	115	0	3	0
8	1	5	13	0	5	0	0	63	1	1
9	0	11	1	0	1	1	8	0	94	1
10	0	23	0	0	7	0	0	1	1	39

Table 4.6: Confusion Matrix for MI feature selection for the highest accuracy of 86.76

Class	1	2	3	4	5	6	7	8	9	10
1	700	0	8	8	0	0	1	0	2	0
2	0	0	0	1	53	0	0	0	2	0
3	7	0	168	2	1	0	0	8	3	0
4	19	0	2	1066	0	0	0	0	0	0
5	0	0	0	1	131	0	1	4	11	1
6	0	0	0	0	1	62	67	0	1	0
7	0	0	1	1	0	16	157	0	4	0
8	3	0	36	0	7	0	0	36	7	0
9	0	0	2	1	5	0	12	0	97	0
10	0	0	0	0	68	0	0	1	1	1

Tf*df_normalized:

We selected the top F feature set across the classes and performed experiments as mentioned below:

Table 4.7: Results on Tf*df_normalized selected features

pdf → feature size across all classes↓	MN	PO	Empirical
100	84.93	83.10	66.63
150	86.47	84.64	70.07
200	86.97	85.32	71.58
250	86,58	85.36	72.51
300	86.65	85.43	73.08
350	86.93	85.50	73.62
400	86.79	85.50	74.38
500	86.76	85.50	74.59
1000	86.86	85.71	76.13
1500	86.72	85.61	76.35
3000	86.86	85.14	76.53
13450	87.15	83.60	76.31

We have shown the three confusion matrices, one for the best separation between the classes 2, 5 and 10 in Table 4.8. Second matrix in Table 4.9 shows the confusion matrix for the groups $G1$, $G2$, $G3$, $G4$ and $G5$. We

are using this matrix's $G1$ for second level of the feature selection. Third confusion matrix in Table 4.10 is for the best accuracy.

For the first matrix, we can analyze that the separation between the classes 2, 5 and 10 is average. The accuracy for class 2 is 51.78%, class 5 is 41.61% and class 10 is 45.07%. These features are able to distinguish between class 2,5 and 10 at an average rate, which is good in comparison to other mentioned feature selection method's results.

For the second matrix, it can be seen that the accuracy is highest for 13450 features across all the class. We can conclude that as the features size increases, though the accuracy increases, but the effect of features to distinguish between overlapping class decreases. For class 2 and class 5 the accuracy is very low, which can be seen in the second matrix.

Table 4.8: Confusion Matrix for best case for tf*df_normalized feature selection where we selected 350 features across all the classes with accuracy 86.93

Class	1	2	3	4	5	6	7	8	9	10
1	695	0	11	5	2	1	0	2	3	0
2	0	29	0	0	21	0	0	0	2	4
3	3	0	142	0	0	0	0	42	2	0
4	21	0	9	1054	0	1	2	0	0	0
5	0	34	0	0	62	0	0	9	8	36
6	0	0	0	0	0	102	28	0	1	0
7	1	0	1	0	0	47	124	0	6	0
8	1	0	7	0	0	0	0	80	1	0
9	0	0	2	0	1	1	7	0	103	3
10	0	10	0	0	25	0	0	3	1	32

Table 4.9: Confusion Matrix for the groups G1, G2, G3, G4 and G5 formed from the above shown confusion matrix of the 350 features across all the classes with accuracy 86.93

Class	G1	G2	G3	G4	G5
G1	253	11	12	0	0
G2	4	419	3	0	1
G3	0	3	271	0	4
G4	0	3	9	1054	21
G5	2	4	13	5	695

Table 4.10: Confusion Matrix for tf*df_normalized feature selection for the highest accuracy 87.15

Class	1	2	3	4	5	6	7	8	9	10
1	694	0	8	13	0	0	1	0	3	0
2	0	4	0	0	50	0	0	0	2	0
3	3	0	154	0	1	0	0	29	2	0
4	20	0	4	1061	0	1	0	0	1	0
5	0	4	0	0	117	0	1	8	10	9
6	0	0	0	0	0	88	42	0	1	0
7	0	0	1	1	0	32	140	0	5	0
8	3	0	17	0	2	0	0	61	6	0
9	0	0	2	0	4	0	9	0	102	0
10	0	2	0	0	58	0	0	3	0	8

4.3.2 Second Level

In this level of feature selection we get the input documents from the output of the first level of Naive Bayes algorithm implementation. We did this level specifically for classes 2, 5 and 10 to reduce the misclassification. These classes are highly overlapping as shown in previous chapter, hence they result in high misclassification.

In the output of first level, we have taken note of all those test documents which get classified in either of class 2, 5 or 10. We call it group $G1$ as described in previous chapters. These sets of documents are the input at this second level. In particular, we are taking those documents which we

got from $tf * df\_normalized$ feature selection technique and number of such features selected at first level was 350 and we applied multinomial probability distribution. We have a total of 253 documents in this group, which we got from the first level of the Naive Bayes algorithm. We performed the feature selection at this 2^{nd} level again on the training set. We used multinomial distribution for each of the cases of feature selection.

Distribution of documents for each class 2, 5 and 10 is as mentioned below :

Class 2: 54 documents.

Class 5: 132 documents.

Class 10: 67 documents.

Bigrams

We selected bigrams as discussed in the feature selection section. We selected bigrams on per class. We choose Multinomial distribution here and for all other techniques below, for various values of bigrams feature set size we run our test as shown:

Table 4.11: Results for 2^{nd} level bigrams features for classes 2, 5 and 10

feature size per class → distribution type↓	100	150	200	1800	2000	3500	10000
MN	49.40	48.61	47.82	46.24	45.05	44.26	52.17

We can see that due to overlap of classes 2, 5 and 10 the misclassification is very high. We have shown below the confusion matrix for few cases in Table 4.12, Table 4.13, Table 4.14 and Table 4.15. We can figure it out that as the feature size increases the documents of classes 2, 10 all go to class 5. For lower feature size the distinguishing capability of the features set for class 2 and 10 is very good. Best accuracy for these cases as shown in Table 4.12 for class 2 is 70.37%, class 5 is 34.84% and class 10 is 61.19%. We calculated

accuracy on the basis of the total number of documents in the group $G1$.

Table 4.12: Confusion Matrix for bigrams feature selection for the feature size of 100 bigrams per class for classes 2, 5 and 10

Class	2	5	10
2	38	13	3
5	42	46	44
10	15	11	41

Table 4.13: Confusion Matrix for bigrams feature selection for the feature size of 200 bigrams per class for classes 2, 5 and 10

Class	2	5	10
2	39	10	5
5	45	39	48
10	15	9	43

Table 4.14: Confusion Matrix for bigrams feature selection for the feature size of 1800 bigrams per class for classes 2, 5 and 10

Class	2	5	10
2	37	12	5
5	47	40	45
10	17	10	40

Table 4.15: Confusion Matrix for bigrams feature selection for the feature size of 10000 bigrams per class for classes 2, 5 and 10

Class	2	5	10
2	0	54	0
5	0	132	0
10	0	67	0

Union of unigrams and bigrams (chosen using special criteria)

We selected unigrams using some special criteria as discussed in the feature selection section. We selected bigrams on per class basis. Here for various values of unigrams and bigrams feature set size we run our test as shown:

We can see that due to overlap of classes 2, 5 and 10 the misclassification is very high. We have shown below the confusion matrix for few cases in Table 4.16, Table 4.17, Table 4.18 and Table 4.19. We can figure it out that as the unigram feature set size increases the documents of classes 5 and 10 all go to class 2. For lower feature set size the distinguishing capability of the features set for class 2 and 10 is again very good. We can get the accuracy from the Table 4.16, which is 83.33% for class 2, 62.68% for class 10 and 10.60% for class 5. This does not work good for class 5.

Table 4.16: Confusion Matrix for union of unigrams and bigrams feature selection method for the feature size of 200 bigrams per class and 50 unigrams per class for classes 2, 5 and 10

Class	2	5	10
2	45	4	5
5	69	14	49
10	18	7	42

Table 4.17: Confusion Matrix for union of unigrams and bigrams feature selection method for the feature size of 200 bigrams per class and 600 unigrams per class for classes 2, 5 and 10

Class	2	5	10
2	54	0	0
5	132	0	0
10	67	0	0

Table 4.18: Confusion Matrix for union of unigrams and bigrams feature selection method for the feature size of 100 bigrams per class and 50 unigrams per class for classes 2, 5 and 10

Class	2	5	10
2	46	3	5
5	70	14	48
10	20	5	42

Table 4.19: Confusion Matrix for union of unigrams and bigrams feature selection method for the feature size of 100 bigrams per class and 100 unigrams per class for classes 2, 5 and 10

Class	2	5	10
2	54	0	0
5	132	0	0
10	67	0	0

Tf*df_normalized features

We selected as discussed in the feature selection section. We selected these features across all the classes on the basis of features having topmost variance. Here we choose features for only classes 2, 5 and 10. Total documents for the group $G1$ is 253. These are those set of documents which are classified at the first level of the Naive Bayes classification in one of the classes 2, 5 or 10. We used multinomial distribution.

For cases with lower feature size we can see that the class 2 and class 10 documents are getting classified to correct classes upto some extent. We have shown various confusion matrix in Table 4.21, Table 4.22, Table 4.23 and Table 4.24. We can see that as the feature size increases, the classes 2 and 10 gets nothing classified into it. For feature size of 3000 as shown in Table 4.22, we have comparable accuracy for class 5 and 10 with values 61.36% and 55.22% respectively. From Table 4.21 we can find that the accuracy for class 2, 5 and 10 is balanced and almost similar to each other with the value of 64.8% for class 2, 58.20% for class 10 and 40.9% for class 5. For larger feature set and for class 5, all documents get classified to this class.

Table 4.20: Results for 2^{nd} level tf*df_normalized features

feature size → distribution type↓	100	150	200	300	500	1000	1500	3000	13450
MN	50.19	50.59	49.80	50.59	50.59	50.98	51.77	52.17	52.17

Table 4.21: Confusion Matrix for tf*df_normalized feature selection method for the feature size of 150 features across all the classes 2, 5 and 10

Class	2	5	10
2	35	11	8
5	37	54	41
10	10	18	39

Table 4.22: Confusion Matrix for tf*df_normalized feature selection method for the feature size of 1500 features across all the classes 2, 5 and 10

Class	2	5	10
2	19	22	13
5	19	61	52
10	5	11	51

Table 4.23: Confusion Matrix for tf*df_normalized feature selection method for the feature size of 3000 features across all the classes 2, 5 and 10

Class	2	5	10
2	14	31	9
5	14	81	37
10	1	29	37

Table 4.24: Confusion Matrix for tf*df_normalized feature selection method for the feature size of 13450 features across all the classes 2, 5 and 10

Class	2	5	10
2	0	54	0
5	0	132	0
10	0	67	0

Chapter 5

k-Nearest Neighbour (k-NN) Rule

5.1 Introduction

k nearest neighbor or k-NN classification determines the decision boundary locally. k is a user-defined constant, and an unlabeled test document is classified by assigning the class-label which is most frequent among the k training samples nearest to that query point. For 1-NN we assign each document to the class-label of its closest neighbor. For k-NN we assign each document to the majority class-label of its k closest neighbors where k is a parameter chosen by user. For k=1 the classification decision of each test document relies just on the class-label of a single training document, which may be incorrectly labeled.

We use several similarity measures for a test document with the training documents set. A commonly used distance metric for continuous variables is Euclidean distance. We are also using cosine similarity. A shortcoming of the k-NN algorithm is that it is sensitive to the local structure of the data.

Algorithm:

1: For each test document T_j find out the first k nearest documents from the training set using some similarity function.

2: Similarity function can be Euclidean distance based measure or cosine similarity.

3: Find out the majority of labels which are in the top k nearest neighbours from training set. Assign that label as the class of the test document.

5.2 Feature Selection

The best choice of k depends upon the data. Generally, larger values of k reduce the effect of noise on the classification, but make boundaries between classes less distinct. The accuracy of the k-NN algorithm can be severely degraded by the presence of noisy or irrelevant features. We experimented on the following feature sets:

1. Term frequency : We used the full features set and the weights for features as term frequency.
2. Tf*idf
   (a) Choosing as features: This is same as we did in feature selection section of Chapter 4. We are dealing with Okapi BM25 $tf * idf$, we chose top l features from each of the class and take union of them and this is our new feature set. We are choosing term frequency as feature weight.
   (b) $Tf * idf$ as feature weight: Here we selected the inverse-document frequency (idf) for each term from the training set and we get

the term frequency of test documents and we multiplied them to get a new feature: $\Lambda_i = tf_i * idf_{i_{train}} \quad \forall i \in [1, |V|]$

In below mentioned all of the approaches, we are using the same concept as given in the Naive-Bayes section, and we are selecting top l_i features from each class. We are choosing term frequency as feature weight.

3. Mutual Information
4. Based on a new measure:
5. Bigrams
6. Unigrams based on specific criteria : Here we used only the unigrams which we get from the classes 2, 5 and 10 as explained in previous chapter's feature selection section. We want to experiment these for improving the results of misclassification among classes 2, 5 and 10.

5.3 Training and Results

The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

Here we are using Reuters-21578 training sets which consists of 7193 training documents and 2787 test documents as described in earlier sections.

For all these except part (b) of Tf*idf we are operating on term frequency of each term as a measure used in similarity function. For part (b) of Tf*idf we are using the measure as $\Lambda(t_f)$ as defined in previous section.

We are taking best features selected during feature selection step and we perform the test set document classification using them. They are mentioned

in each section.

5.3.1 Term Frequency

First of all we performed for the full features set and noted the results. We performed two measures for measuring similarity for documents: Euclidean distance and cosine similarity.

Table 5.1: Results of k-NN algorithm for full features

$(k) \rightarrow$ similarity $f^n\downarrow$	1	5	11	15	17	21	31	51
Cosine Similarity	81.41	84.03	83.74	83.38	83.10	83.06	82.41	81.73
Euclidean Distance	76.67	78.90	76.85	76.06	75.09	74.63	73.37	69.50

Table 5.2: Confusion Matrix for full features using cosine similarity for k-NN where k=5

Class	1	2	3	4	5	6	7	8	9	10
1	627	0	11	74	1	0	3	0	3	0
2	0	34	2	2	17	0	0	0	1	0
3	6	4	160	2	0	0	1	13	3	0
4	7	0	1	1076	0	1	0	0	2	0
5	0	45	7	2	81	0	1	4	9	0
6	2	0	0	2	0	88	37	0	2	0
7	1	1	1	1	0	30	143	0	2	0
8	3	2	34	2	7	0	0	36	5	0
9	2	4	0	3	3	1	7	0	97	0
10	0	20	3	0	45	0	0	2	1	0

5.3.2 Tf*idf

Feature weights as tf*idf

Here as described in previous section we are using the feature weights of test set for the k^{th} term as the product of term frequency of test document and idf of the k^{th} term of the training set, if it exists.

We got not so good results with this experimentation:

Results when we performed for cosine similarity:

k=5, accuracy = 73.08

k=11, accuracy = 72.87

k=21, accuracy=73.16

Also the rate of misclassification among classes 2, 5 and 10 is high. So we discarded the further experiments.

Top Okapi BM25 tf*idf features set per class

We are choosing best 50 Okapi $BM25\ tf * idf$ features per class from the feature selection method as discussed in the previous chapter.

Table 5.3: Results of k-NN algorithm for top Okapi $BM25\ tf * idf$ features selection method.

$(k) \rightarrow$ similarity $f^n\downarrow$	1	5	11	15	17	21	31	51
Cosine Similarity	83.31	85.43	85.75	86.04	86.00	85.71	85.14	84.67
Euclidean Distance	81.98	81.62	81.37	81.09	80.91	80.33	78.97	77.17

We choose one confusion matrix as shown in Table 5.4 for the best distinction among classes 2, 5 and 10 using cosine similarity and Table 5.5 for the best distinction among classes 2, 5 and 10 using Euclidean distance based similarity. In Table 5.4, accuracy for class 2 documents is 69.64% and for class 5 it is 53.69%, but for class 10 all the documents either classified to class 2 or class 5. In Table 5.5, accuracy for class 2 documents is 55.35% and for class 5 it is 48.99%, but for class 10 almost all the documents either classified to class 2 or class 5. Here, we can see that it does not work well for class 10.

Table 5.4: Confusion Matrix for top Okapi $BM25\ tf * idf$ features selection method using cosine similarity for k-NN where k=5

Class	1	2	3	4	5	6	7	8	9	10
1	677	2	8	30	0	1	0	0	1	0
2	0	39	0	0	14	0	0	0	3	0
3	8	2	151	5	2	0	0	15	6	0
4	6	0	2	1079	0	0	0	0	0	0
5	1	47	5	1	80	0	2	4	9	0
6	5	0	1	4	0	92	28	0	1	0
7	9	0	2	9	0	34	121	1	3	0
8	2	0	31	4	7	0	0	41	4	0
9	3	1	1	1	4	3	3	0	101	0
10	1	23	2	1	41	0	0	2	1	0

Table 5.5: Confusion Matrix for top Okapi $BM25\ tf * idf$ features selection method using Euclidean distance for k-NN where k=5

Class	1	2	3	4	5	6	7	8	9	10
1	693	0	2	22	1	0	0	0	1	0
2	5	31	0	1	14	1	0	0	3	1
3	51	0	114	7	3	0	2	8	4	0
4	13	0	1	1072	0	1	0	0	0	0
5	20	36	2	4	73	1	0	4	8	1
6	20	0	1	3	0	86	20	0	1	0
7	40	1	1	9	0	40	86	0	2	0
8	17	0	27	6	5	0	0	30	4	0
9	11	1	0	5	3	2	6	0	89	0
10	5	15	1	1	45	1	0	1	1	1

5.3.3 Mutual Information

We are choosing best 50 mutual information features per class from the feature selection method as discussed in the previous chapter.

Table 5.6: Results of k-NN algorithm for MI

$(k) \rightarrow$ similarity $f^n\downarrow$	1	5	11	15	17	21	31	51
Cosine Similarity	83.17	85.43	86.11	85.68	85.46	85.39	85.14	84.78
Euclidean Distance	82.56	83.38	82.81	82.13	81.84	81.37	80.08	78.04

We choose one confusion matrix as shown in Table 5.7 for the best distinction among classes 2, 5 and 10 using cosine similarity and Table 5.8 for the best distinction among classes 2, 5 and 10 using Euclidean distance based similarity. In Table 5.7, accuracy for class 2 documents is 51.78% and for class 5 it is 61.74%, but for class 10 all the documents either classified to class 2 or class 5. In Table 5.8, accuracy for class 2 documents is 51.78% and for class 5 it is 53.02%, but for class 10 all the documents either classified to class 2 or class 5. Here, we can see that again it does not work well for class 10.

Table 5.7: Confusion Matrix for MI feature selection method using cosine similarity for k-NN where k=11

Class	1	2	3	4	5	6	7	8	9	10
1	668	0	7	37	2	0	2	0	3	0
2	0	29	0	1	20	1	0	0	5	0
3	8	1	147	2	1	1	1	24	4	0
4	9	0	1	1076	0	1	0	0	0	0
5	0	30	3	5	92	2	1	3	13	0
6	1	0	0	2	0	93	31	0	4	0
7	7	0	2	2	0	27	137	0	4	0
8	2	1	20	2	8	0	0	53	3	0
9	0	1	0	4	1	1	5	0	105	0
10	0	13	3	4	46	1	0	2	2	0

Table 5.8: Confusion Matrix for MI feature selection method using Euclidean distance for k-NN where k=5

Class	1	2	3	4	5	6	7	8	9	10
1	682	0	3	31	0	0	1	0	2	0
2	4	29	2	1	17	0	0	0	3	0
3	22	1	145	5	1	2	3	8	2	0
4	11	1	0	1073	0	1	1	0	0	0
5	7	37	3	5	79	3	2	4	9	0
6	10	0	0	2	0	94	24	0	1	0
7	20	0	1	3	0	50	102	0	3	0
8	10	1	36	4	5	1	1	28	3	0
9	3	2	0	8	2	2	8	0	92	0
10	1	17	2	3	44	1	0	1	2	0

5.3.4 Tf*df_normalized

We selected top 500 features as mentioned in previous chapter and performed k-NN algorithm on those features.

Table 5.9: Results of k-NN algorithm for TF*df_normalized

$(k) \rightarrow$ similarity $f^n\downarrow$	1	5	11	15	17	21	31	51
Cosine Similarity	83.60	85.25	85.14	85.03	85.07	85.28	84.39	83.67
Euclidean Distance	81.95	81.84	81.91	81.41	81.12	80.57	79.76	77.89

We choose one confusion matrix as shown in Table 5.10 for the best distinction among classes 2, 5 and 10 using cosine similarity with k=5 and Table 5.11 for the best distinction among classes 2, 5 and 10 using Euclidean distance based similarity. In Table 5.10, accuracy for class 2 documents is 58.92% and for class 5 it is 57.04%, but for class 10 all the documents either classified to class 2 or class 5. In Table 5.11, accuracy for class 2 documents is 51.78% and for class 5 it is 53.02%, but for class 10 all the documents either classified to class 2 or class 5. Here, we can see that again it does not work well for class 10.

Table 5.10: Confusion Matrix for Tf*df_normalized feature selection method using cosine similarity for k-NN where k=5

Class	1	2	3	4	5	6	7	8	9	10
1	669	1	11	35	0	0	2	0	1	0
2	0	33	2	1	19	0	0	0	1	0
3	10	0	155	2	1	1	0	18	2	0
4	6	1	3	1077	0	0	0	0	0	0
5	0	42	8	2	85	0	1	4	7	0
6	4	0	0	2	0	85	39	0	1	0
7	9	2	1	2	0	27	134	0	4	0
8	3	1	31	3	7	0	0	41	3	0
9	4	2	1	2	2	1	8	0	97	0
10	0	19	4	1	44	0	0	2	1	0

Table 5.11: Confusion Matrix for Tf*df_normalized feature selection method using Euclidean distance for k-NN where k=5

Class	1	2	3	4	5	6	7	8	9	10
1	682	0	1	32	1	0	1	0	2	0
2	3	29	2	0	16	1	1	0	4	0
3	43	2	129	3	3	0	1	6	2	0
4	15	0	2	1069	0	1	0	0	0	0
5	13	36	2	3	79	1	1	4	10	0
6	16	0	0	0	0	88	25	0	2	0
7	34	1	1	2	0	42	97	1	1	0
8	17	2	35	3	6	1	0	25	0	0
9	11	1	0	7	7	2	5	1	83	0
10	4	16	1	2	45	1	0	1	1	0

5.3.5 Bigrams

Here we selected top 200 features from classes 2, 5 and 10 only and did our experimentation on the whole classes to find out the overall results and interaction between the classes 2, 5 and 10.

Table 5.12: Results of k-NN algorithm for Bigrams

$(k) \rightarrow$ similarity $f^n\downarrow$	1	5	11	15	17	21	31	51
Cosine Similarity	76.92	78.79	76.85	75.88	75.45	74.63	73.26	69.64

We find out the confusion matrix for the best result for k=5 in Table 5.13. We try to analyze the effect of top bigrams of the classes 2, 5 and 10 in this resultant matrix. But the accuracy of class 10 is not that good as can be seen in the results.

Table 5.13: Confusion Matrix for best case for Bigrams feature selection of classes 2, 5, 10 for k=5

Class	1	2	3	4	5	6	7	8	9	10
1	588	1	6	107	1	11	3	0	2	0
2	0	30	3	5	15	2	0	0	1	0
3	34	3	132	11	0	4	1	2	2	0
4	7	0	2	1076	0	1	1	0	0	0
5	6	38	6	12	69	8	1	2	7	0
6	4	0	0	0	0	101	24	0	2	0
7	16	0	1	3	0	44	112	0	3	0
8	28	2	28	9	6	2	0	11	3	0
9	12	5	0	9	4	5	5	0	77	0
10	3	16	2	5	42	2	0	0	1	0

5.3.6 Unigrams based on specific criteria

Here we selected top 600 features from classes 2, 5 and 10 only and did our experimentation on the whole classes to find out the overall results and interaction between the classes 2, 5 and 10.

Table 5.14: Results of k-NN algorithm for union of bigrams and unigrams

$(k) \rightarrow$ similarity $f^n\downarrow$	5	11	21
Cosine Similarity	67.16	72.94	74.20

We figured out that these results did not give good results. Classes 2, 5 and 10 missclassification rate is also high. Results shows that our feature selection experiments do well for classes 2 and 5, but for class 10 it does not work good.

Chapter 6

Hierarchical Classification Algorithm

6.1 Introduction

We are making use of our unsupervised and supervised classification which we discussed in previous chapters. At first level, we are performing the SOM algorithm mentioned in Chapter 3. After this we get about $|CL|= 100$ clusters. Each of the training documents $\vec{X_l} \in X,\ \forall\ l = \{1, 2, \cdots, N\}$ gets classified to one of the clusters $CL_i \in CL,\ \forall\ i = \{1, 2, \cdots, |CL|\}$. We will store the value of the final stable weights which we get at the end of the SOM algorithm in a file. Now we find the test documents $\vec{T_k} \in T$ which are closest to one of the clusters $CL_i \in CL$. We use one of the two similarity measures to get the closest cluster: cosine similarity and Euclidean distance. We have noted down the training documents in a file which gets classified to a cluster during the SOM algorithm step, those documents in CL_i act as the training sets for the new test document which has to be classified. We pass these new sets of training documents to our 2^{nd} level supervised classifier to

get the class of that test document. Using the appropriate feature selection technique we get a new feature set and then we process each test document using the new feature set in our 2^{nd} level classifier model. We have shown the block diagram in Fig. 6.1.

Algorithm:

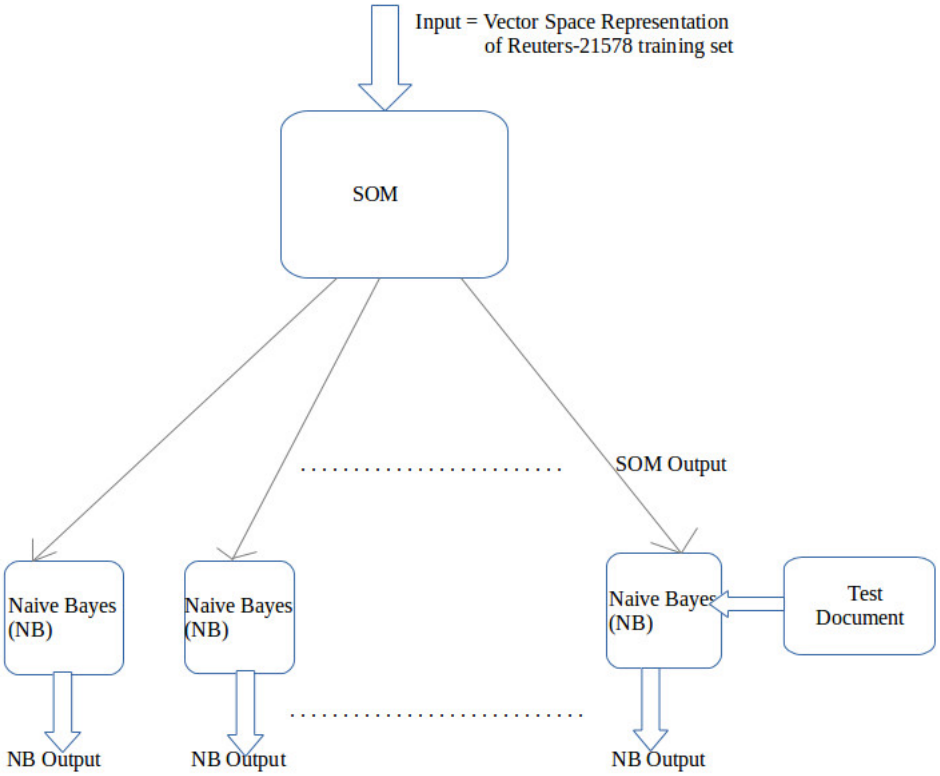
$X = training\ set$

$T = test\ set$

1. First Level:
   (a) Apply SOM algorithm to training set, we get output as clusters set CL. We make a matrix of cluster number by class to see which cluster has documents of which class. We note down the documents belonging to each cluster and its corresponding class.
   (b) We store the stable weight vectors(as described in $Chapter3$) in a file, which will be used later in next level.
2. Second Level:
   (a) For each test document we find the closest cluster $CL_j \in CL$ using the stored stable weight vectors.
      i. If the closest cluster has documents which belong to only one class, say $c_i \in C$ then we classify the given test document to that particular class c_i. Otherwise, we move to next step ii.
      ii. Feature selection step: In case for the nearest cluster $CL_j \in CL$, we have documents which belongs to more than one classed then we select appropriate features for further application of our classification algorithm. This is explained in details in feature selection section.

iii. We apply Naive Bayes classifier at each clusters $CL_i \in CL$. Training documents at this level are the documents at each clusters CL_i. We are using the newly select feature set F as explained in feature selection section. We classify this test document to the highest probable class.

Figure 6.1: Block diagram representation of the hierarchical classification algorithm.



6.2 Feature Selection

Since we are training our classifiers by the documents at each level of the clusters that we got from the output of the SOM program, and the number

of documents at a particular level might be low for training so we came up with following features selection strategies to get the effective results :

1. Measure we defined: We look for those clusters which are having documents classified in more than one class, let us assume total of k such clusters are present. Let us say each such cluster is contained in a set $CL' = \{CL'_1, CL'_2, \cdots, CL'_k\}$. For each of such clusters, we figure out number of classes in that cluster which are having positive values from the stored cluster list. For each such classes we take union of the documents that are present in this class in the particular cluster, we get these documents from the training set. When we take union of the documents in each of the positive classes, we get a new feature vector for each such clusters. For example, lets say p' number of positive classes are present in a cluster CL'_f $\forall f \in [1, k]$, when we take union of the documents in each of the p' positive classes, then we get a new feature set for all such p' classes in the cluster CL'_f. Now to select the appropriate features from this new feature set, we do following:

   (a) If $p' = 2$, lets say these two classes are c_i and c_j, and lets say $\langle t_k \rangle$ is the k^{th} term in the vocabulary set V. For each of such k^{th} term $\langle t_k \rangle$ of class c_i and c_j, let us say the associated weight is $\mu_{i,k}$ and $\mu_{j,k}$ respectively. Now we select only those terms in selected feature set F which are satisfying this condition: $\frac{\mu_{i,k}}{\mu_{j,k}} > \epsilon$ or $\frac{\mu_{j,k}}{\mu_{i,k}} > \epsilon$, where we have taken $\epsilon = 1.5$.

   (b) If $p' > 2$, lets say these two classes are denoted as c_i $\forall i = \{1, 2, \cdots, p'\}$, and lets say $\langle t_k \rangle$ is the k^{th} term in the vocabulary set V. For each of such k^{th} term $\langle t_k \rangle$ of class c_i, let us say the associated weight is $\mu_{i,k}$ $\forall i = \{1, 2, \cdots, p'\}$. For each term $\langle t_k \rangle$, we find a_k=min$(\mu_{1,k}, \mu_{2,k}, \cdots, \mu_{p',k})$ and b_k=max$(\mu_{1,k}, \mu_{2,k}, \cdots, \mu_{p',k})$.

Now we select only those terms in selected feature set F which are satisfying this condition: For each term $\langle t_k \rangle$, $\frac{a_k}{b_k} > \epsilon$ or $\frac{b_k}{a_k} > \epsilon$, where we have taken $\epsilon = 1.5$.

Using this selected feature set F, we implement the rest part of the algorithm.

2. Based on previous feature selection methods.

   Here we used the features as described in the previous sections:

   (a) Tf*idf

   (b) MI score

   (c) Tf*df_normalized

6.3 Training and Results

At SOM level, the training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples. Here we are using top 10 categories of Reuters-21578 training sets which consists of 7193 training documents and 2787 test documents as described in earlier chapters.

We are first running the SOM algorithm for the training set. We are neglecting those terms which comes in more than $N/2$ documents where N is the size of training set.

When we are using metric for terms as Okapi $BM25$ $tf * idf$ from the test set. We got good clusters. But since test documents are coming one by one, we do not have any way to measure idf (inverse document frequency) values of each test term. So we took tf (term frequency) of a term belonging to

a test document and if this term occurs in the vocabulary set of training documents set than we are taking idf of that term from the training set, then we have are calculating the $tf * idf$ of the corresponding test term. But we did not get comparable results using this approach, so we choose tf only as the metric for each term.

First we ran this algorithm on the training set for the term frequency as the features. We are measuring similarity between documents and cluster weights at two places in SOM algorithm first during the update step and second during cluster forming step as mentioned in Chapter 3 . We can use two similarity measures: Cosine similarity and Euclidean distance. We are using combination of these two.

We are running SOM algorithm in 4 possible ways:

1. Measure for similarity of Nearest weight for a document in Update step and closest cluster per document in cluster forming step is cosine similarity

2. Measure for similarity of Nearest weight for a document in Update step and closest cluster per document in cluster forming step is Euclidean distance

When we are done with the SOM algorithm, we get 100 clusters. Training documents in each clusters act as the input to the second level classifier. Hence, the second level of this hierarchical classifier we have 100 classifiers. We compare a test document to the clusters weights which was stored, to get the closest cluster for that test document. Now using the training documents at this cluster which got stored to this cluster in first level of SOM algorithm, we perform the further Naive Bayes classification algorithm. The number of training documents at a cluster may be very low, so we used clever feature

selection strategy to consider only the useful features, that give us the best performance.

We took 4 different runs of our SOM algorithm and test the test-set. We had 2787 test documents set.

Full features

First of all we ran the Naive Bayes on the SOM output for full features and analyzed these results. Best accuracy came for run 4 of the SOM program and for cosine similarity as shown in Table 6.1.

Table 6.1: Result for Hierarchical classifier for Full features

similarity $f^n \rightarrow$ SOM program run $\downarrow$	Cosine Similarity	Euclidean distance
1	83.53	83.53
2	83.42	82.31
3	83.63	83.02
4	83.96	82.84

Table 6.2 shows the confusion matrix for full feature set. We can see that the documents of class 2, 5 and 10 are not able to get distinguished properly using full features. Almost all documents from class 2 and 10 gets classified to class 5.

New approach for feature selection

We are using this new approach which is mentioned in this chapter's feature selection section. Best accuracy is of 83.96% for the run 1 of SOM program and for the cosine similarity, which is shown in Table 6.3. These results are almost similar to results of full features set of previous section. They also did not distinguish features of class 2, 5 and 10 to that great extent.

Table 6.2: Confusion Matrix for full feature

Class	1	2	3	4	5	6	7	8	9	10
1	648	0	9	48	1	0	7	1	5	0
2	0	12	0	1	42	0	0	0	1	0
3	11	0	128	6	1	0	1	35	7	0
4	16	0	2	1065	0	0	1	0	3	0
5	0	12	1	9	111	0	3	5	7	1
6	1	0	0	6	0	84	38	0	2	0
7	8	0	1	4	0	30	131	0	5	0
8	3	0	5	2	6	0	1	68	4	0
9	2	0	0	12	5	0	6	0	92	0
10	0	3	1	2	60	0	0	2	2	1

Table 6.3: Result for Hierarchical classifier for our New approach

similarity $f^n \rightarrow$ SOM program run $\downarrow$	Cosine Similarity	Euclidean distance
1	83.96	79.83
2	83.45	79.83
3	83.71	80.91
4	83.92	80.33

Confusion matrix for the best case of accuracy 83.92% for both cosine similarity case as shown in Table 6.4.

Table 6.4: Confusion Matrix for the new approach for feature selection

Class	1	2	3	4	5	6	7	8	9	10
1	646	0	9	49	1	1	7	1	5	0
2	0	12	1	1	41	0	0	0	1	0
3	10	0	141	5	1	0	2	23	7	0
4	14	0	2	1067	0	0	1	0	3	0
5	0	12	2	9	111	0	3	5	7	0
6	0	0	0	6	0	83	40	0	2	0
7	4	0	1	4	0	31	135	0	4	0
8	2	0	22	2	6	0	2	51	4	0
9	2	0	0	12	4	0	6	0	93	0
10	0	3	2	2	60	0	0	2	2	0

Okapi BM25 tf*idf

Here we have used the features having top Okapi BM25 $tf * idf$ values per class. We tested the SOM algorithm for all the two cases of similarity functions as mentioned in the introduction section. We varied the size of selected features to note down the variations in the results. Table 6.5 is for cosine similarity measure with best accuracy of 85.25% for run 3 of SOM program and in Naive Bayes, we are using the top 50 selected features per class. Table 6.6 is for Euclidean distance with best accuracy of 86.76% for run 4 of SOM program and in Naive Bayes, we are using the top 50 selected features per class.

Table 6.5: Result for Hierarchical classifier for selected features on the basis of Okapi BM25 Tf*idf where both similarity functions are cosine similarity.

Number of features per class → SOM Program Run f^n ↓	20	50	100
run 1	84.75	84.78	84.39
run 2	84.49	84.89	84.53
run 3	85.18	85.25	84.39
run 4	85.14	85.03	85.18

Table 6.6: Result for Hierarchical classifier for selected features on the basis Okapi BM25 tf*idf where both similarity functions are Euclidean distance based similarity.

Number of features per class → SOM Program Run f^n ↓	20	50	100
run 1	85.18	85.61	85.93
run 2	85.54	86.61	85.46
run 3	85.32	86.65	86.00
run 4	85.64	86.76	85.57

In Table 6.7, we are showing the confusion matrix for the best case that we thought of having good accuracy as well as which distinguishes among class 2, 5 and 10 upto some good extent. Accuracy of class 2 is 37.5% , class 5 is 55.05% and class 10 is 33.80%.

Table 6.7: Confusion Matrix for the Okapi BM25 $tf * idf$ feature selection method. Features set size is chosen as top 50 features per class and with Euclidean distance similarity function and accuracy of 85.61% and run 1 of SOM program

Class	1	2	3	4	5	6	7	8	9	10
1	682	0	9	21	1	1	1	2	2	0
2	0	21	0	0	29	1	0	0	1	4
3	6	0	144	1	0	1	1	34	2	0
4	34	0	3	1046	0	2	0	2	0	0
5	1	24	0	0	82	2	1	7	5	28
6	0	0	1	0	0	85	42	0	2	0
7	4	0	0	1	0	29	137	0	8	0
8	2	0	13	0	1	0	0	66	3	4
9	1	1	1	0	6	1	5	1	99	2
10	0	7	0	0	38	0	0	2	0	24

MI

Here we used the features having top MI values per class. We tested the SOM algorithm for all the two cases of similarity functions as mentioned in the introduction section. We varied the size of selected features to note down the variations in the results. Table 6.8 is for cosine similarity measure with best accuracy of 85.49% for run 3 of SOM program and in Naive Bayes, we are using the top 100 selected features per class. Table 6.9 is for Euclidean distance with best accuracy of 86.79% for run 3 of SOM program and in Naive Bayes, we are using the top 50 selected features per class.

Table 6.8: Result for Hierarchical classifier for selected features on the basis MI where both similarity functions are cosine similarity

Number of features → SOM Program Run f^n ↓	20	50	100
run 1	85.03	84.60	84.42
run 2	84.96	84.85	84.35
run 3	85.46	85.28	85.49
run 4	85.21	85.14	85.00

In Table 6.10, we are showing the confusion matrix for the best case that we

Table 6.9: Result for Hierarchical classifier for selected features on the basis of MI where both similarity functions are Euclidean distance based similarity

Number of features → SOM Program Run f^n ↓	20	50	100
run 1	85.64	85.46	86.18
run 2	86.58	86.79	85.61
run 3	86.43	86.79	86.25
run 4	86.11	86.47	85.75

thought of having good accuracy as well as which distinguishes among class 2, 5 and 10 upto some good extent. Accuracy of class 2 is 35.71% , class 5 is 60.40% and class 10 is 23.94%.

Table 6.10: Confusion Matrix for the MI feature selection method. Features set size is chosen as top 50 features per class and with Euclidean distance similarity function and accuracy of 86.79% and run 3 of SOM program

Class	1	2	3	4	5	6	7	8	9	10
1	686	0	9	18	0	1	2	1	2	0
2	0	20	0	0	33	1	0	0	1	1
3	3	1	158	0	0	0	1	25	1	0
4	20	0	2	1063	0	1	1	0	0	0
5	0	25	0	0	90	1	1	9	4	19
6	1	0	0	0	0	82	46	0	2	0
7	1	0	0	0	0	26	151	0	1	0
8	1	2	22	1	1	0	1	59	2	0
9	0	0	2	2	13	1	6	0	93	0
10	0	12	0	0	40	0	0	2	0	17

Tf*df_normalized

Here we used the features having top $Tf * df\_normalized$ values across the classes. We tested the SOM algorithm for all the two cases of similarity functions as mentioned in the introduction section. We varied the size of selected features to note down the variations in the results. Table 6.11 is for cosine similarity measure with best accuracy of 85.18% for run 4 of SOM program and in Naive Bayes, we are using the top 500 selected features.

Table 6.12 is for Euclidean distance with best accuracy of 86.47% for run 2 of SOM program and in Naive Bayes, we are using the top 500 selected features.

Table 6.11: Result for Hierarchical classifier for selected features on the basis Tf*df_normalized where both similarity functions are cosine similarity

Number of features → SOM Program Run f^n ↓	200	500	1000
run 1	84.32	84.35	84.39
run 2	84.28	84.75	84.67
run 3	84.96	84.53	84.39
run 4	85.10	85.18	85.14

Table 6.12: Result for Hierarchical classifier for selected features on the basis Tf*df_normalized where both similarity functions are Euclidean distance based similarity

Number of features → SOM Program Run f^n ↓	200	500	1000
run 1	84.82	85.57	86.22
run 2	85.28	86.47	85.46
run 3	85.50	86.32	85.93
run 4	84.67	86.25	86.11

In Table 6.13, we are showing the confusion matrix for the best case that we thought of having good accuracy as well as which distinguishes among class 2, 5 and 10 upto some good extent. Accuracy of class 2 is 33.92% , class 5 is 53.69% and class 10 is 32.39%.

We can see that this result is very low as compared to other class accuracy, because this above mentioned accuracy is for overlapping classes 2, 5 and 10. In this section we can see that $tf * df\_normalized$ method gives an average and balanced performance for the the overlapping classes 2, 5 and 10 as compared to other feature selection techniques.

After getting these results, we can input them to the second level of Naive Bayes classifier which is mentioned in Chapter 4 where we selected features

Table 6.13: Confusion Matrix for the $tf*df\_normalized$ for feature selection. Features size is 500 and with Euclidean distance similarity function and accuracy of 86.32% for run 3 of SOM program.

Class	1	2	3	4	5	6	7	8	9	10
1	686	0	5	21	0	3	1	1	2	0
2	0	19	0	1	34	0	0	0	1	1
3	9	1	149	1	1	0	1	25	1	1
4	17	0	2	1067	0	1	0	0	0	0
5	0	25	0	1	80	0	1	9	4	29
6	1	0	0	0	0	80	46	1	3	0
7	1	0	0	0	0	27	150	0	1	0
8	1	4	20	1	2	0	0	59	2	0
9	0	0	2	0	12	1	7	0	93	2
10	0	13	0	0	33	0	0	2	0	23

at second level which shows considerable improvement to the accuracy for class 2, 5 and 10. Hence the total accuracy of the combined system should increase proportional to the results mentioned in section of second level feature selection for Chapter 4.

Chapter 7

Conclusion and Future Work

We have used Reuters-21578 ModApte split top 10 class. This is a dataset which has a lot of overlapping classes. We discussed and extensively experimented for one of such overlapping class 2, 5 and 10 which have corn, grain and wheat label respectively, as termed in the Introduction Chapter in Table 1.2.

We tried to get the best results using the conventional multi-class classification using SOM algorithm and Naive Bayes algorithm. Through several experiments for this data set which dealt with various types of feature selection techniques we tried to distinguish between overlapping classes. We achieved a little success.

We first used SOM algorithm on the training set to get an idea about the possible related documents which gets classified to a particular cluster, this helps in deciding the closest and most similar cluster for a test document and helps to reduce the possibility of test document getting misclassified.

After we got the nearest cluster for test document, Naive Bayes at second level of the hierarchical classifier helps to decide the correct class of the

test document. The cluster from which test document is closest to might have overlapping classes, since documents of these classes are similar to each other. Hence we get the need to search only those important features which gives the valuable information which will help to discriminate between documents belonging to overlapping classes. We gave two level approach for selecting features.

From the results we concluded that class 5 is super-set of class 2 and class 10. Using all the features, the documents of class 2 and 10 gets classified to class 5.

At the first level, all of the feature selection methods as described below performed very good in terms of accuracy, for multinomial probability distribution. A result may have very high accuracy but the inter-class separation between the overlapping documents may be very high. They varied how they classify documents belonging to class 2, 5 and 10. We performed all the results first on the training set than experimented on the test set.

First Level:

1. Feature selection using features having top Okapi BM25 $tf * idf$ values per class.

   For lower feature set size of 50 features per classes, this approach gave accuracy of 53.57% for class 2, 41.77% for class 10 and 40.2% for class 5, which is an average distinction between overlapping classes. For higher features this accuracy dropped to negligible value for class 2 and 10 respectively, and for class 5 it was 79.86%.

2. Feature selection using features having top Mutual Information values per class.

   For lower feature set size of 10 features per class, MI approach gave

accuracy of 92.8% for class 2, 54.9% for class 10 and 19.46% for class 5. For higher features this accuracy dropped very low for class 2 and 10 respectively, and for class 5 it was 87.91%.

3. Feature selection using features having top $tf * df\_normalized$ values across all the classes.

   For lower feature set size of 350 features across all the classes, this approach gave accuracy of 51.78% for class 2, 40.50% for class 10 and 41.61% for class 5. For higher features this accuracy dropped very low for class 2 and 10 respectively, and for class 5 it was 78.52%.

At second level of Naive Bayes classification we take the input as the documents of class 2, 5 and 10, which get classified to among themselves and we further run new feature selection algorithms to distinguish between these overlapping classes. This set of documents lie in group G1 as mentioned in Chapter 4. G1 contains 253 documents. The accuracy is measured with respect to the documents of this group only.

1. Feature selection using bigrams selected using specific criteria per class for classes 2, 5 and 10.

   For lower feature set size of 100 features per class, this approach gave accuracy of 70.3% for class 2, 61.19% for class 10 and 34.8% for class 5, which is a quite good distinction between overlapping classes as compared to above first level results. For larger feature set and for class 5, all documents get classified to this class.

2. Feature selection using union of unigrams and bigrams selected using specific criteria per class for classes 2, 5 and 10.

   For lower feature set size, this approach gave accuracy of 83.33% for class 2, 62.68% for class 10 and 10.60% for class 5. This result shows

that it considerably helped to distinguish the class 2 and class 10 documents from class 5 documents. For larger feature set and for class 5, all documents get classified to this class.

3. Feature selection using features having top $tf * df\_normalized$ values across all the classes 2, 5 and 10.

   For lower feature set size of 150 features across classes 2, 5 and 10, this approach gave accuracy of **64.8%** for class 2, **58.20%** for class 10 and **40.9%** for class 5. For larger feature set and for class 5, all documents get classified to this class.

Next we tried to see the relationship between these overlapping classes using the k-NN algorithm. We used all the above feature selection methods and figured out that it was able to distinguish between class 2 and 5 at an average rate but could not distinguish class 10 documents. Best results are for k=5 and $tf * df\_normalized$ feature selection strategy with 58.92% for class 2 and 57.04% for class 5.

We got some good results in our main algorithm for these classes 2, 5 and 10 along with overall accuracy. We saw that, given less number of training set documents at a particular cluster, this algorithm performs very well with the highest accuracy of 86.79% using MI feature selection method for size of feature set equal to top 50 features per class with Euclidean distance similarity measure. Accuracy of class 2 is 35.71% , class 5 is 60.40% and class 10 is 23.94%.

Although we could clearly see that the balance in the accuracy among classes 2, 5 and 10 is not that good. But for the $tf * df\_normalized$ feature selection strategy with overall accuracy of 86.32% for run 3 of SOM program as shown in table 6.13, we have a comparably good balance among the accuracy for classes 2, 5 and 10 with the values of class 2 is 33.92% , class 5 is 53.69%

and class 10 is 32.39%. Though the accuracy is less per class for these classes specifically, but the selected feature set for this technique is able to evenly distribute test documents of these classes to their correct individual class. We can summarize from the results of the second level of the feature selection as mentioned in Chapter 4 and also above, that we can enhance the performance of the hierarchical classifier which we saw in Chapter 6 by applying the features of second level to the output of the hierarchical classifier. We will get a proportionate improvement as shown in second level feature selection of Chapter 4.

As a future task we can think of making use of bigrams in the first level of feature selection technique because they gave very good results in second level of feature selection strategy for classes 2 and 10 in Chapter 4. The reason is because they capture more information as compared to single words as features. Also, we can think of taking union of feature sets from MI selection algorithm of first level with any other feature selection technique's feature set. MI gives best performance for class 2 and class 5, $tf * df\_normalized$ at second level gives in one of the case best for class 5 and class 10 only. Similarly union of unigrams and bigrams gives good results for class 2 and 10. So we can do an exhaustive search to find that best set which gives best performance to distinguish the documents of classes 2, 5 and 10.

Bibliography

[1] Anirban Dasgupta, Petros Drineas, Boulos Harb, Vanja Josifovski, and Michael W. Mahoney. Feature selection methods for text classification. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*, pages 230–239, 2007.

[2] Susan T. Dumais, John C. Platt, David Hecherman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the 1998 ACM CIKM International Conference on Information and Knowledge Management, Bethesda, Maryland, USA, November 3-7, 1998*, pages 148–155, 1998.

[3] Gnes Erkan, Ahmed Hassan, Qian Diao, and Dragomir R. Radev. Improved nearest neighbor methods for text classification with language modeling and harmonic functions.

[4] Susana Eyheramendy, David D. Lewis, and David Madigan. On the naive bayes model for text categorization. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics, AISTATS 2003, Key West, Florida, USA, January 3-6, 2003*, 2003.

[5] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[6] Chang-Hwan Lee, Fernando Gutierrez, and Dejing Dou. Calculating feature weights in naive bayes with kullback-leibler measure. In *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*, pages 1146–1151, 2011.

[7] David D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *Machine Learning: ECML-98, 10th European Conference on Machine Learning, Chemnitz, Germany, April 21-23, 1998, Proceedings*, pages 4–15, 1998.

[8] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[9] Tom M. Mitchell. *Machine learning*. McGraw Hill series in computer science. McGraw-Hill, 1997.

[10] Jyri Saarikoski, Kalervo Järvelin, Jorma Laurikkala, and Martti Juhola. On document classification with self-organising maps. In *Adaptive and Natural Computing Algorithms, 9th International Conference, ICANNGA 2009, Kuopio, Finland, April 23-25, 2009, Revised Selected Papers*, pages 140–149, 2009.

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

[12] Gerard Salton and Michael McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, 1984.

[13] Gerard Salton, C. S. Yang, and Clement T. Yu. A theory of term importance in automatic text analysis. *JASIS*, 26(1):33–44, 1975.

[14] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, 2002.

[15] Yiming Yang. An evaluation of statistical approaches to text categorization. *Inf. Retr.*, 1(1-2):69–90, 1999.

[16] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 15-19, 1999, Berkeley, CA, USA*, pages 42–49, 1999.

[17] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML 1997), Nashville, Tennessee, USA, July 8-12, 1997*, pages 412–420, 1997.