
Automatic Document Classification

By

Jignesh Jani

11MICT19

Guided By

Prof. Vijay Ukani



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AHMEDABAD-382481

MAY 2013

Automatic Document Classification

Major Project (Part-II)

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology in Information and Communication Technology

By

Jignesh Jani

(11MICT19)

Guided By

Prof. Vijay Ukani



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AHMEDABAD-382481

MAY 2013

Certificate

This is to certify that the Major Project(Part-II) entitled “Automatic Document Classification” submitted by Jignesh Jani (11MICT19), towards the partial fulfillment of the requirements for the degree of Master of Technology in Information and Communication Technology of Nirma University, Ahmedabad is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project part-I, to the best of my knowledge, haven’t been submitted to any other university or institution for award of any degree or diploma.

Prof. Vijay Ukani
Guide, Associate Professor,
Department of C.S.E.,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Sanjay Garg
Professor and Head,
Department of C.S.E,
Institute of Technology,
Nirma University, Ahmedabad.

Prof. Gaurang Raval
Associate Professor,
PG-Coordinator, ICT,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Ketan Kotecha
Director,
Institute of Technology,
Nirma University, Ahmedabad

Undertaking for Originality of work

I, Jignesh S. Jani, Roll. NO. 11MICT19, give undertaking that the major project entitled "Automatic Document Classification" submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in Information and Communication of Nirma University, Ahmedabad, is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Signature of Student

Date: Place:

Endorsed By

Signature of Guide

Abstract

Enormous amount of documents are generated everyday life. There is always a need to retrieve the documents over any medium. This has come up with the solution of classifying the documents with appropriate label. A lot of research has been done on this topic to classify them using different classifiers. The classifier used in this research is Nave Bayes classifier, due to its simplicity. The Nave Bayes classifier classifies a document only under one class no matter by what fraction posterior probabilities of other classes are smaller. By considering the fraction by which other associated terms are smaller we rank a document more into a specific context but also little less into another context. The Apriori algorithm is used to find the frequent patterns out of the document which will give the context of the document and will help in labeling the document with more appropriate classification tag. The proposed approach is to classify the document with Nave Bayes classifier at first level and then finding associated terms from documents and comparing them with the already mined frequent patterns from the train dataset. This two level classification gives more precise label to the document.

Acknowledgements

My deepest thanks to **Prof. Gaurang Raval**, Associate Professor, PG-Coordinator(ICT), Department of Computer Science and Engineering, Institute of Technology, Nirma University, Ahmedabad. I want to give my sincere thanks to my guide **Prof. Vijay Ukani**, under whom I have done this project. Though he knows about my internship outside college, he always has guided me whenever needed even when I reply late regarding any issue I faced. He has taken the pain to go through the project and make necessary amendments as and when needed. I thank HOD **Dr. Sanjay Garg**, for his support and providing facilities at the department level.

I would like to thank **Dr.Ketan Kotecha**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for his unmentionable support, providing basic infrastructure and healthy research environment.

I would also thank my Institution, all my faculty members in Department of Computer Science.

Last, but not the least, I thank my parents and my wife for constant support, because of which I could complete my dissertation work successfully.

-Jignesh Jani

11MICT19

Contents

Certificate	iii
Undertaking for Originality of Work	iv
Abstract	v
Acknowledgements	vi
List of Tables	ix
List of Figures	x
1 Introduction	1
2 Literature Survey	3
2.1 Document Classification	3
2.1.1 pre-processing	3
2.1.2 Feature Construction	4
2.2 Term Frequency Inverse Document Frequency	4
2.3 Dealing with bunch of letters and with whole sentences	5
2.4 Precision and Recall	7
2.5 Naïve Bayes Classifier for document classification	7
2.6 Flat Classification and Hierarchical Classification	8
2.6.1 Comparison of Flat classification with Hierarchical Classification	9
2.7 Finding Frequent Patterns to be more specific in classification	10
3 Proposed Approach	11
3.1 Problem with the existing approach	11
3.2 Proposed Approach: Naïve Bayes with Apriori Algorithm	11
4 Implementation	13
4.1 Tools	13
4.2 Integration of Apriori with Naïve Bayes	13
4.3 Classification through MALLET	13
4.4 Technique	14
4.5 Naïve Bayes Trainer	17

5	Simulation and Result Analysis	18
5.1	Document having multiple contexts	18
5.2	Extraction and listing of Associated terms with their Context	19
5.3	Results	20
6	Conclusion and Future Work	23

List of Tables

2.1	Classification Accuracy of Different Classifiers	8
2.2	Time Taken to build different Classifiers	8
5.1	Associated terms with their context	20
5.2	Results of some test documents	22

List of Figures

5.1	Pictorial Representation of Document having multiple context	18
5.2	Associated Attributes using CF's subset Evaluator	19
5.3	Naïve Bayes classifier with training time 0.04 seconds	20
5.4	DT Classifier with training time 0.17 seconds	21
5.5	True positive documents out of 60 documents	21
5.6	Time taken to train three classifiers	22

Chapter 1

Introduction

Wide variety and vast number of documents are generated if we look at one big organization or in normal work culture. These Documents belong to some domains or categories. Classification is used to label the documents with this categories so user can understand the concept behind it and can understand what information it stores, either critical, secure or normal. When these categories are not known to normal users it becomes difficult task to label them. The reason behind choosing this topic is to lessen the burden on the user to judge the classification of the document and classify the unlabeled documents in a mass.

So many research has been done and is still going on for this topic but the approach followed in this report is not classifying the document once and labeling it with the concluded classification. But to have more specific classification down the hierarchy, each document will be treated twice with two different algorithms. If we consider the hierarchy of the labels then we may have numerous labels for each document. e.g. If we have a document with the data regarding Sachin Tendulkar, it may be classified under sports. But down the hierarchy it comes under Cricket. For this kind of classification we will include the TF and IDF finding in the Apriori algorithm itself.

Naïve Bayes classifier gives the probabilistic output[13] i.e. It lists the classification tags with the probability and the tag with the highest probability is suggested by Naïve Bayes classifier. But only by calculating the occurrence of terms in the document, documents can not be classified perfectly. The Apriori algorithm is used to resolve this

issue. The frequent patterns are found from the document and they are listed under the classification tag. This will give more specific classification.

Chapter 2

Literature Survey

2.1 Document Classification

Automatic classification techniques use algorithms that learn from human classifications, so they can only do as well as the human training data provided. Different algorithms can learn different types of patterns in the data. There are different Machine Learning algorithms that have been used. They include Neural Networks, Naïve Bayes, Support Vector Machine and k-nearest Neighbors. Each of these methods has their advantages and limitations on classification performance and scalability[?]. The choice of algorithm will depend on the application, and the amount of data to be used. We can distinguish two phases in automatic document classification, the learning phase and the subsequent classification phase. In the learning phase users define categories in which they are interested by giving sample documents (training examples) for each of these categories. The main pre-processing task is removal of stop words and applying stemmer algorithm on the document[?]. Then the feature construction approach is followed.

2.1.1 pre-processing

This algorithm is mainly based on the Porter stemming algorithm. Removing suffixes by automatic means is an operation, which is especially useful in the field of information retrieval. Terms with a common stem will usually have similar meanings. The algorithm defines that any word or part of a word may be represented by the single form $[C](VC)_m[V]$. Where C a list of consonant, V a list of vowel and the square brackets denotes arbitrary presence of their contents. Using $(VC)_m$ to denote VC repeated m

times. The rules for removing a suffix will be given in the form

$$(condition)S1 \rightarrow S2 \tag{2.1}$$

This means that if a word ends with the suffix S1, and the stem before S1 satisfies the given condition, S1 is replaced by S2. Therefore REPLACEMENT will be replaced with REPLACE.

2.1.2 Feature Construction

Almost all existing learning and classification techniques require vectors of (real) numbers as input. They cannot work directly on documents. Therefore, vector representations of documents have to be constructed in order to make these methods applicable. The process of constructing these vector representations is called feature (vector) construction. We compared two kinds of features, viz. letter 5-gram features and features constructed by a morphological analysis. The number of unique features in the document collection determines the dimension of the feature vector representations for the documents and the position of each feature in an alphabetically ordered list of all features determines its position in the feature vector representations. A feature vector representation for a document is simply a vector of weights for all the features.

2.2 Term Frequency Inverse Document Frequency

Term Frequency is defined as the number of occurrence of the term in the document[?]. The more the number of occurrence the more the document closer to one category. Term frequency is known to improve recall in information retrieval, but does not always improve precision. Because frequent terms tend to appear in many texts, such terms have little discriminative power. In order to remedy this problem, terms with high frequency are usually removed from the term set. Finding optimal thresholds is the main concern in this method. Inverse Document Frequency is defined as the occurrence of a term across the set of documents, or the ratio of term occurring in documents and total no of documents in the set. This IDF concept is used since along with the highly occurring term, rare occurrence of a word is also helpful. The rare the word is more easy is the task to classify. Since IDF represents term specificity, it is expected to improve the precision. Solution proposed to combine term frequency and IDF to weight terms, and showed that

the product of them gave better performance. The factors $TF(d,t)$ and $IDF(t)$ would contribute to improve the recall and the precision respectively. Weighted IDF is an extended version of IDF[5]. A drawback of IDF is that all the texts that contain a certain term are treated equally. That is, IDF does not distinguish between one occurrence of a term in a text and many. IDF assumes that the importance of a term is inversely proportional to the number of texts that contain the term. We have to weight this factor by the frequency of each term. Unlike IDF, WIDF differs for each text.

Term Frequency is defined as

$$TF(t_i, d) = \text{count}(t_i) / W_d \quad (2.2)$$

where W_d is number of words in document d .

Inverse Document Frequency is defined as

$$IDF(T_i) = \log(|D| / DF(T_i)) \quad (2.3)$$

The document frequency $DF(T_i)$ is the number of documents in which term T_i occurs at least once. $|D|$ is the total number of documents.

Therefore TFIDF can be written as

$$TFIDCF(T_i, d) = TF(T_i, d) * IDF(T_i) \quad (2.4)$$

$TFIDF(T_i, d)$ is the weight of term T_i in document d .

2.3 Dealing with bunch of letters and with whole sentences

The approach which deals with the bunch of letters is called letter n-gram approach. Here the whole file will be tokenized with n-letter words. Here the meaning does not matter since n-letter may not bring the meaning to it. One difficulty in handling some classes of documents is the presence of different kinds of textual errors, such as spelling

and grammatical errors. The system is based on calculating and comparing profiles of N-gram frequencies[4]. First, we use the system to compute profiles on training set data that represent the various categories, e.g. language samples or newsgroup content samples. Then the system computes a profile for a particular document that is to be classified. Finally, the system computes a distance measure between the documents profile and each of the category profiles. The system selects the category whose profile has the smallest distance to the documents profile. An N-gram is an N-character slice of a longer string. Although in the literature the term can include the notion of any co-occurring set of characters in a string (e.g. an N-gram made up of the first and third character of a word). N-gram is a co-occurrence statistics data which collect every adjacent word from huge web documents. It has been applied many research field especially Natural Language Processing field. Because of linguistic advantage of n-gram it is possible to extract core features of documents written in human language. Also, n-gram applied to speech recognition,topic discovery and a system for filling out incomplete sentences. The n-gram can be expressed into two different ways. First is a method for extracting the n-gram by each adjacent English character from documents. Second is the n-gram by each adjacent word. Typically, one slices the string into a set of overlapping N-grams. We also append blanks to the beginning and ending of the string in order to help with matching beginning-of-word and ending of-word situations. In Letter n-gram approach the n characters are separated for each word. e.g. 5-grams for "very good" will be "very", "ery g", "ry go", "y goo" and " good"[4]. The total number of n-gram is depending on the number of word in a given document and what type of n-gram will be constructed. It simply follows given formula.

$$t_{Ngram} = t_{Word} - depth + 1 \quad (2.5)$$

where, tNgram is total number of constructed n-gram, tWord is total number of word in given the document, and depth is the type of n-gram. If we want to extract bigram, then depth will be 2. This will help in removing the spelling mistakes and other errors from the document. And the classification and counting TF will be easy.

Second approach which deals with the sentences known as morphological approach. Here the whole sentences are taken into account and meaning is fetched from this. Here more number of comparison has to be made since we have longer sentences and combi-

nation of more than one word. But it represent meaning full words and so classifier built from morphological approach are more efficient than letter n-gram approach.

N-gram letter features have a variety of advantages compared to morpheme features. Implementation of n-gram feature construction is very easy and independent of language. N-gram features automatically perform certain kinds of stemming and they are robust against misspellings. Furthermore, N-gram features automatically capture many kinds of multi word phrases, if one considers n-grams across word-borders (inter-word n-grams).

2.4 Precision and Recall

When Machine tries to classify previously unseen documents, the result may be desired or undesired. These results are put under four categories. True Positive, False Positive, True Negative and False Negative.

Precision can be describe as the ratio of truly identified documents from relevant documents. Recall can be defined as the ratio of truly identified documents and total available documents under that category. We are using this measures to find the effectiveness of the classifiers. In this research the measure used is F1 measure.

$$\text{precision} = \frac{\text{Number of documents retrieved that are relevant}}{\text{Total number of documents that are retrieved}}$$

$$\text{recall} = \frac{\text{Number of documents retrieved that are relevant}}{\text{Total number of documents that are relevant}}$$

$$\text{F-measure} = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}$$

2.5 Naïve Bayes Classifier for document classification

Document classification is a growing interest in the research of text mining. Correctly identifying the documents into particular category is still presenting challenge because of large and vast amount of features in the dataset. In regards to the existing classifying

approaches, Naïve Bayes is potentially good at serving as a document classification model due to its simplicity. Results show that Naïve Bayes is the best classifiers against several common classifiers (such as decision tree, neural network, and support vector machines) in term of accuracy and computational efficiency. The below given tables shows that the Naïve Bayes is better than other classifiers. This is simple simulated WEKA.

Table 2.1: Classification Accuracy of Different Classifiers

Classifiers	TP and TN	FP and FN	Precision	Recall	F-Measure
Naïve Bayes	2717	83	0.970	0.970	0.970
SVM	2712	88	0.969	0.969	0.969
DT	2551	249	0.911	0.911	0.911

Table 2.2: Time Taken to build different Classifiers

Classifiers	Time Taken to Build (seconds)
Naïve Bayes	0.19
SVM	2.69
DT	1.8

Naïve Bayes models allow each attribute to contribute towards the final decision equally and independently from other attributes, in which it is more computational efficient when compared with other text classifiers. Naïve Bayes classifier is the simplest instance of a probabilistic classifier. The output $\Pr(C=d)$ of a probabilistic classifier is the probability that a document d belongs to a class C . Each document contains terms which are given probabilities based on its number of occurrence within that particular documents. With the supervised training, Naïve Bayes can learn the pattern of examining a set of test documents that have been well-categorized and hence comparing the contents in all categories by building a list of words as well as their occurrence. Thus, such list of word occurrence can be used to classify the new documents to their right categories, according to the highest posterior probability.

2.6 Flat Classification and Hierarchical Classification

Naïve Bayes approach is used to deal with the problem of document classification via a deceptively simplistic model: assume all features are independent of one another, and

compute the class of a document based on maximal probability. The Naïve Bayes approach is applied in Flat (linear) and hierarchical manner for improving the efficiency of classification model. It has been found that Hierarchical Classification technique is more effective than Flat classification.

The Naïve Bayes classifier performs its classification tasks starting with analyzing the text document by extracting words which are contained in the document. To perform this analysis, an extraction is used to extract each individual word from the document to generate a list of words. This will be termed as vocabulary. —vocabulary— =the total number of distinct word set found.

Within all the training data This list is helpful when the probabilistic classifier calculates the probability of each word being annotated to each category. The list of words is then used to generate a table, containing the words which are extracted from the input document. The probabilistic classifier is needed to be trained with a set of well-categorized training dataset. Each individual word which will be matched with words contained in vocabulary from all training documents in the same category are extracted and listed in a list of words occurrence for the particular category. Based on the list of word occurrence, the trained probabilistic classifier calculates the posterior probability of the particular word of the new unlabeled document being annotated to particular category by using the formula which is shown as equation(1). The prior probability, $Pr(\text{Category})$ can be computed from equation (2).

2.6.1 Comparison of Flat classification with Hierarchical Classification

Limitation of flat classification is that as the number of possible categories increases the distinction between document classes get blurred. While in hierarchical structure as we go down in hierarchy document types become more specific. In flat classification relationship among documents cannot be identified thus it proves problematic in multi-label classification. While the hierarchical structure identifies the relationship among the classes which allows for efficiency in both learning and representation.

2.7 Finding Frequent Patterns to be more specific in classification

To find frequently occurring items in transactions of a database, Apriori Algorithm is very widely used because of its speed and ease of use. It scans through every transaction adding one item to a frequent item set at every step and builds possible list of n-frequent item sets by pruning those candidates which do not have specified support or lift. It uses the anti-monotone property. When text documents make up the database, we need to convert the word frequencies into binary form. Each sentence of a document or a single document can be considered as a transaction.

Chapter 3

Proposed Approach

3.1 Problem with the existing approach

Naïve Bayes classifier gives the most probable classification tag using the TFIDF value of each term. But only calculating the word frequency and inverse document frequency does not always give correct classification. When we apply Naïve Bayes, it classifies a document only under one class no matter by what fraction posterior probabilities of other classes are smaller. By considering the fraction by which other associated terms are smaller we rank a document more into a specific context but also little less into another context. To overcome this issue we are integrating Apriori algorithm of frequent pattern mining to find out the associated terms from the documents and giving more precise and more concrete labels.

3.2 Proposed Approach: Naïve Bayes with Apriori Algorithm

Huge amount of unstructured data is available in the form of text documents. Classifying these text documents by considering their context will be very useful in information retrieval.

In proposed approach we classify the document first with Naïve Bayes classifier. One of the strengths of the NB classifier is its simplicity and in many instances it performs much better than more complex classification methods such as Support Vector Machines This will give the probability associated with each term and will give the probabilistic output. We find the context of an abstract by looking for associated terms which help

us understand the focus of the abstract and interpret the information beyond simple keywords. The document is represented as a vector of terms and its weights.

$$D = T_1 : W_1, T_2 : W_2, \dots, T_n : W_n \quad (3.1)$$

At first level, all the documents are classified using Naïve Bayes classifier. We then find words that occur together in at least 50% documents of each class by using Apriori algorithm. We use this set of associated words in each class at second level to determine different contexts of a text document.

$$S_{at} = \{\{T_1, T_2\} : C_1, \{T_4, T_5\} : C_2\} \quad (3.2)$$

Above equation states that There can be document in which Term 1 and Term 2 may define one context and Term 4 and Term 5 may define another context. Each test document is scanned to find the associated words determined in Apriori approach. Then the occurrence of words are count in the document which are there in associated terms and they are sorted in decreasing order. Thus it indicates that a document may belong to one context more but also may belong to some other context. When we apply Naïve Bayes, it classifies a document only under one class no matter by what fraction posterior probabilities of other classes are smaller. By considering the fraction by which other associated terms are smaller we rank a document more into a specific context but also little less into another context.

Chapter 4

Implementation

4.1 Tools

MAchine Learning for Language Toolkit is a tool used in this project for simulation. Naïve Bayes classifier is used in this classification task. WEKA is used in this case to prove Naïve Bayes classifier better than the other classifiers e.g. SVM and DT.

4.2 Integration of Apriori with Naïve Bayes

The proposed approach is to find out frequent patterns out of the testing documents and list them in file. These patterns are actually context of the document. Then in testing documents find out the associated terms with the patterns and list them in non-increasing or decreasing order. As soon as the Naïve Bayes classifier classifies the document we apply the proposed approach and will find the associated terms. The frequency of the terms are calculated and according to the weight, the percentage of the context is assigned. The more the percentage of the context, the more the chances of being classified as label.

4.3 Classification through MALLET

MALLET is a Java-based package for statistical natural language processing, document classification, clustering, topic modeling, information extraction, and other machine learning applications to text. MALLET includes sophisticated tools for document classification: efficient routines for converting text to features, a wide variety of algorithms (including Naïve Bayes, Maximum Entropy, and Decision Trees), and code for evaluating classifier performance using several commonly used metrics.

In addition to sophisticated Machine Learning applications, MALLET includes routines for transforming text documents into numerical representations that can then be processed efficiently. This process is implemented through a flexible system of **pipes**, which handle distinct tasks such as tokenizing strings, removing stopwords, and converting sequences into count vectors.

The toolkit is Open Source Software, and is released under the Common Public License. The Naïve Bayes classifier is used from this tool which gives the probability of each class.

4.4 Technique

To classify any document some traditional steps need to be followed every time i.e. pre-processing. which are listed below.

- Remove stopwords from the document e.g.a, an, the, now, is, are, ...
- Use stemmer algorithm for stemming like and accepted, acceptance will be simply replaced with accept
- feature construction from the pre-processed document.

These feature construction is made through many different algorithms. Some of them are Rank Search, CF's Subset Evaluator. This will give the complete vocabulary-no of distinct necessary words from all the documents. WEKA will help us with this to find out the features with Chi-Squared Attribute Evaluation. This is done through finding Term Frequency (TF) and Inverse Document Frequency (IDF). Then the multiplication TFIDF is the normalization of the values.

After following above steps we will apply apriori algorithm on the train documents to find out frequent patterns and will list them in .txt file. The name of the .txt file will be the label. The txt file will contain the frequent patterns in increasing order of their occurrence in the document. This .txt file contains the context of the document. If we look at the apriori algorithm we will find that in order to find frequent patterns, in the first pass it calculates the occurrence of the single terms as well which will give us the TF in the document.

At first level the Naïve Bayes will classify the document using following formula

$$P(C/D) = \frac{P(D/C) * P(C)}{P(D)} \quad (4.1)$$

where $P(D)$ will remain same so we can neglect it and $P(C)$ can be calculated using following equation

$$P(C) = \frac{T_{wc}}{T_{wts}} \quad (4.2)$$

This is the ratio of Total no of words in class to the Total no of words in training set. The Naïve Bayes classifier will give the probability of the class.

At second level we will find the associated terms with the frequent patterns extracted using apriori algorithm. The document will have more than one context. For e.g. consider a fragment from an abstract Load balancing is applied to the development of network-based Intrusion Detection System (NIDS) to fit the performance problem caused by traffic in high bandwidth network. This abstract contains two contexts network load balancing and network intrusion detection system (IDS).

Thus if a document D represents two contexts C1 and C2 , and if C1 is greater than C2 then if C2 is less than some x percent of C1 then D belong to C1 and then C2 in this order.

To use MALLET we need to follow the below given steps.

- Download the MALLET tool from the <http://mallet.cs.umass.edu/index.php>. It is available for Windows and Unix.
- Extract MALLET file and copy it in one directory.
- set the environment variable to the bin path to the MALLET directory.

After setting the environment for MALLET we can start work with it. To train a classifier in MALLET we first need to import the data in .mallet file. There are two ways to import data in .mallet file. One instance per file and one instance per line. In the former one we need to import the whole directory containing training examples. In the later one we need to import one file containing different training datasets. The one instance per line datasets will contain following format.

[InstanceName] [Label] [feature 1] [feature 2] ...

- To import data in mallet file we need to write the command

```
bin/mallet import-file --input sample-data/web/* --output web.mallet
```

```
bin/mallet import-dir --input sample-data/web/*.txt --output web.mallet
```

The above command will load the data in web.mallet file.

- To train a classifier we need to write the command

```
bin/mallet train-classifier --input training.mallet --output-classifier my.classifier
```

```
--trainer NaïveBayes
```

- To classify the unlabeled documents we need to write the command

```
bin/mallet classify-file --input data --output - --classifier classifier
```

```
bin/mallet classify-dir --input datadir --output - --classifier classifier
```

the above commands will write the results on the standard output. The former one will classify the on instance per line files and the later one will classify one instance per file data.

4.5 Naïve Bayes Trainer

In this approach at the time of training the classifier frequent patterns are found and they are stored under one file named the classification tag. Then the simple Naïve Bayes is trained. When Naïve Bayes classifier classifies the document, the algorithm tries to find out the associated terms with the frequent patterns found out in Apriori algorithm. This way the the frequently occurred associated terms are listed under one file in decreasing order. The document is labeled with highest matching context listed under the classification tagged file. The Naïve Bayes algorithm is modified to integrate Apriori within it and named Naïve Bayes trainer in MALLET.

Chapter 5

Simulation and Result Analysis

5.1 Document having multiple contexts

As we have already discussed, using Apriori algorithm we are extracting frequent patterns from the document and identifying the context of the document. But a document may have multiple context in it. This will lead us to labeling a document with highest valued context. Figure 5.1 shows a pictorial representation of the document having multiple contexts.

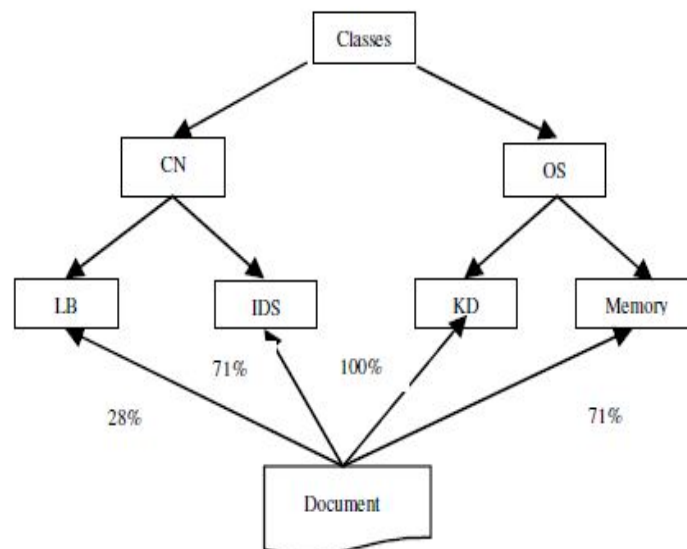
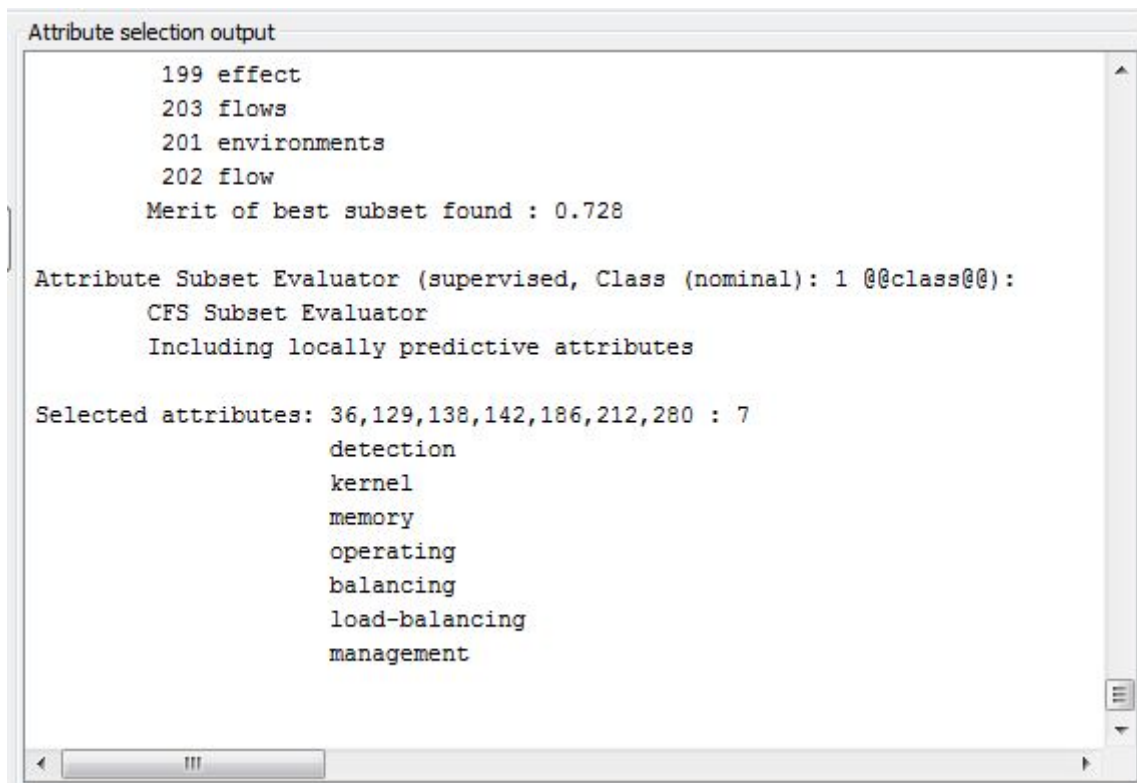


Figure 5.1: Pictorial Representation of Document having multiple context

5.2 Extraction and listing of Associated terms with their Context

To extract features from the document we have used CF's subset feature evaluator. This considers rank search algorithm for selection of attributes. We have shown the selected attributes and most frequent terms are listed below selected attributes. It is shown in below figure.



```
Attribute selection output
199 effect
203 flows
201 environments
202 flow
Merit of best subset found : 0.728

Attribute Subset Evaluator (supervised, Class (nominal): 1 @@class@@):
CFS Subset Evaluator
Including locally predictive attributes

Selected attributes: 36,129,138,142,186,212,280 : 7
detection
kernel
memory
operating
balancing
load-balancing
management
```

Figure 5.2: Associated Attributes using CF's subset Evaluator

In Table 5.1 we have listed the context and associated terms. These are the terms which are found from the document and they are listed in decreasing order of their occurrence.

If the document contains word balance or load many times, according to the highest probability calculated by the Naïve Bayes classifier, its classification turns out to be Computer Networks. Now we will check the .txt file with name ComputerNetwork.txt. And this will contain Load Balancing in it as the frequent pattern extracted through Apriori algorithm. This way we can come one level down in the hierarchy.

Table 5.1: Associated terms with their context

Associated Terms	Context
Balance, load, network, result, scheme, simulate	Computer Network:Load Balancing
Intrusion, Attack, detect, intrusion, network, base, system	Computer Network:Intrusion Detection System
Design, kernel, operating, system	Operating System:Kernel Design
Hardware, memory, management, operating, perform, system	Operating System:Memory Management

5.3 Results

We used 60 abstracts of IEEE explorer in different subjects like Computer Networks:Load Balancing and Intrusion Detecion system, operating System:Kernel design and Memory Management to train the classifier.

We have the results of showing how Naïve Bayes classifier is easy to train because of its simplicity. It takes very less time to get trained. This is shown in below Figure 5.3 and Figure 5.4

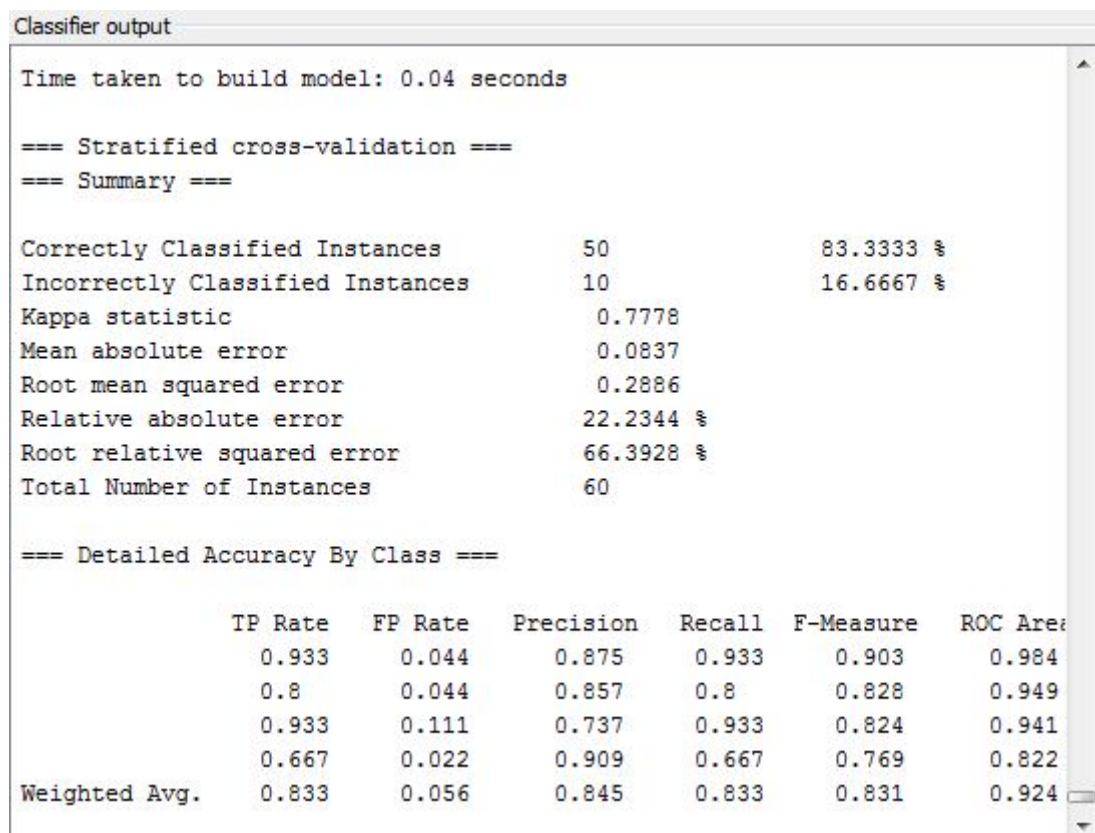


Figure 5.3: Naïve Bayes classifier with training time 0.04 seconds

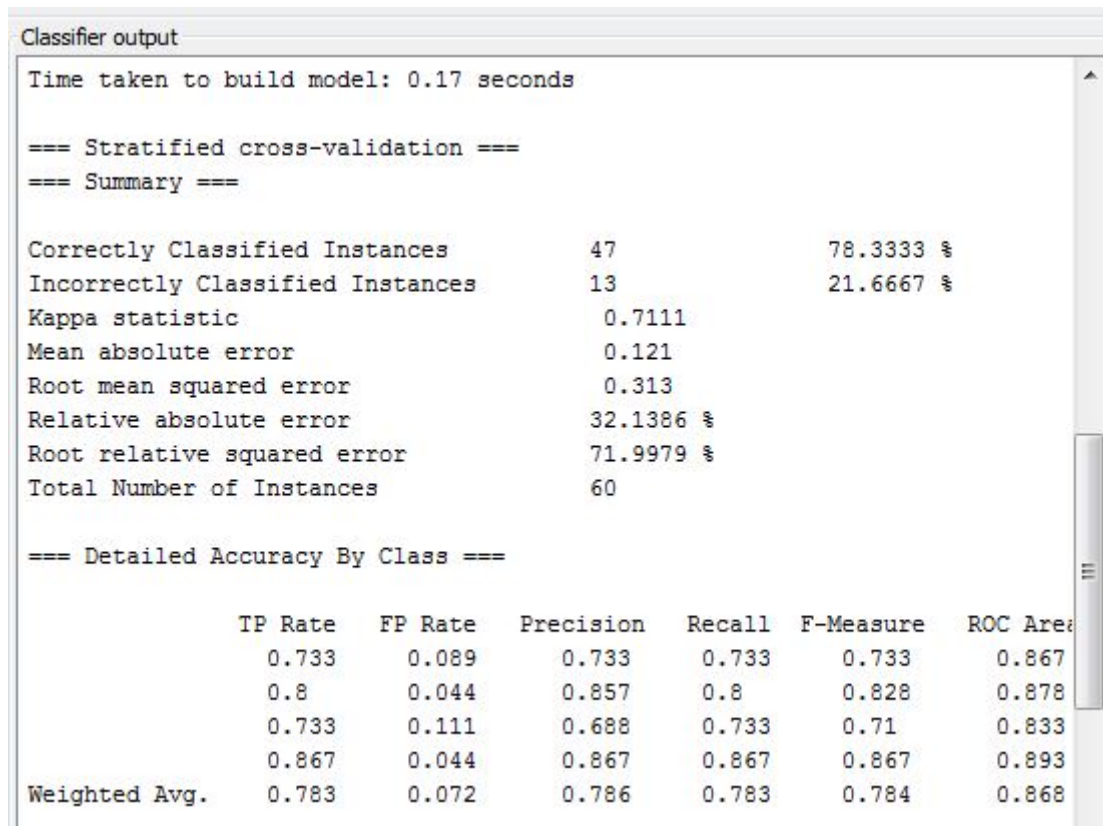


Figure 5.4: DT Classifier with training time 0.17 seconds

We have shown the comparison of three classifiers with their correctly classifier examples (True Positive). The chart given below is generated after working on 60 IEEE paper abstracts on 4 different topics under two different categories Computer Networks and Operating Systems.

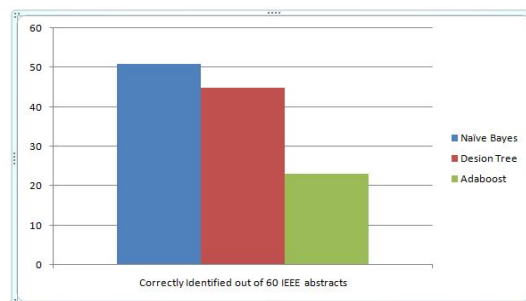


Figure 5.5: True positive documents out of 60 documents

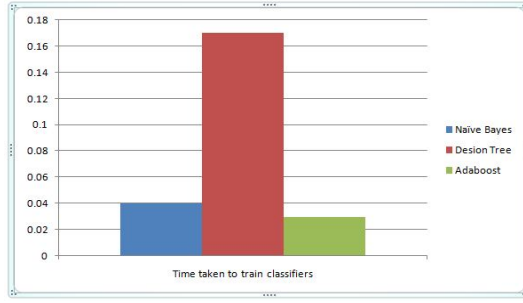


Figure 5.6: Time taken to train three classifiers

As we can see the time taken to train the classifier with our algorithm, Adaboost takes the less amount of time than other two classifiers. But figure 5.5 clearly shows that the truly identified examples are very less in case of adaboost algorithm.

The Table 5.2 shows how a document is representing multiple context and how it is categorized in proper classification. For giving example here we have chosen 4 document which contain more than one context. These way other documents may contain multiple context and context with more association will be chosen as classification.

Table 5.2: Results of some test documents

Test Document	Classification	CN:LB	CN:ID	OS:KD	OS:MM
T1	CN	90%	60%	3%	7%
T2	CN	30%	96%	0%	1%
T3	OS	10%	1%	80%	99%
T4	OS	2%	1%	96%	70%

As we can see that some test document T1 has 90% of the content associated with CN, Load Balancing. And 60% of the context is associated with CN, Intrusion Detection. Thus, we can say that the document classified with Load Balancing.

Simple Naïve Bayes classifier will classify the document T1 under CN category. But Our modified algorithm will classify it under Load Balancing. We can figure out for the other 3 random documents shown in Table 2 under T2, T3 and T4.

Chapter 6

Conclusion and Future Work

The vast no of documents are being created and queried every day. To make the retrieval effective the documents must be classified or labeled. Different classifiers classify the documents with different accuracy. In this report we shown that Naïve Bayes is better classifier than others but, it gives just one classification with highest probability and does not concern with other classification if it is just 0.1% less than the former one. Here after integrating Apriori algorithm with simple Naïve Bayes we have successfully extracted context and one level down hierarchy.

This implementation does not create n-level down the hierarchy and classify the document with more specific classification. This need to be taken care in future work. The proposed approach is implemented, with one level down the hierarchy in mind. To go n-level the implementation need to be modified.

Bibliography

- [1] Goller Christoph, Joachim Lning, Thilo Will, Werner Wolff, "Automatic Document Classification: A thorough Evaluation of various Methods"
- [2] Jonathan McElroy, "AUTOMATIC DOCUMENT CLASSIFICATION IN SMALL ENVIRONMENTS", January 2012.
- [3] Manevitz, Larry, Malik Yousef , "Document Classification on Neural Networks Using Only Positive Examples", 2000.
- [4] Cavnar,William, John, Trenkle, N-Gram-Based Text Categorization"
- [5] Tokunaga, Takenobu, Iwayama, Makoto, "Text categorization based on weighted inverse document frequency", March 1994.
- [6] Hoda, Mohmed, "Automatic Document Classification", 2007.
- [7] Joshi Shweta, Bhavna, Nigam, "Categorizing the Document using Multi Class Classification in Data Mining", 2011.
- [8] M. IKONOMAKIS, S. KOTSIANTIS, V. TAMPAKAS, "Text Classification Using Machine Learning Techniques", August 2005.
- [9] XIU-LI PANG, YU-QIANG, FENG, WEI, JIANG, "An Impoved Document Classification Approach With Maximum Entropy And Entropy Feature Slection", August 2007.
- [10] McCallum, Andrew Kachites, "MALLET: A Machine Learning for Language Toolkit" <http://mallet.cs.umass.edu> 2002.
- [11] Frank Klawonn, Plamen Angelov, "Evolving Extended Naïve Bayes Classifiers", 2006.

- [12] Soumen Chakrabarti, Shourya Roy, Mahesh V. Soundalgekar, "Fast and accurate text classification via multiple linear discriminant projections", 2002.
- [13] S.L. Ting, W.H. Ip, Albert H.C. Tsang, "Is Naive Bayes a Good Classifier for Document Classification", July 2011.
- [14] Razvan Stefan Bot, Yi-fang Brook Wu, Xin Chen, Quanzhi Li, "Generating Better Concept Hierarchies Using Automatic Document Classification", November 2005.