Spam Computation Algorithm for Final Year Project

Jalal Uddin U1366889

April 2016

Contents

List of Figures

1 Acknowledgement

I would like to thank Dr David Wilson for his insightful opinions. always encouraging me and moving me in the right direction with this rather. At times bizarre idea that I happened to have when in conversation with him. This is the result of that idea. I hope I do not disappoint.

2 Summary

Text classification for spam has been a long and arduous battle of developing systems that are able to combat the plague of spam emails. Which have been growing exponentially now with the faster speeds of internet access.
The terms spam denotes the sending of unsolicited electronic mail via the internet to anyone and everyone as described in the paper by (Mendoza, 2011). The protest of receiving spam which in most cases can be troublesome as well as complicating the removal of such mail as they tend to arrive en- masse and therefore take a singular amount of time to delete.
Spam or spamming has now been combated by using sophisticated systems that are able to use statistics and probability theory to discern how much a document is spam or non-spam. Therefore based on mathematics a choice can be made which can deliver an amount of automation, to deal with spam emails that are sent out in their billions on a daily basis around the world.
The questions arise as with the exponential increase in spam how can it be combated to prevent it from spreading so rapidly. Techniques and methods have been developed form the first spam filter developed using the Naive Bayes formula by (Jason Rennie,1996).
To the far more advanced version being built using Artificial Intelligence by Google and Microsoft. The larger global companies have technologies undoubtedly which far in advance of anything that is currently on the market. This ultimately leaves the consumer having to decide on the best generic spam filters that are currently on the market with some having questionable performance at times.

3 Introduction

Any custom spam filter will most times utilise the Naive Bayes theorem to allow for the reasoning and learning of the model. It will be figuratively developed for that sole purpose to see if the application works with modified Bayes algorithms adapted for this current applications process.

The essence of this application has been to make sure that the application has the necessary training exercised to it to make it work correctly. The model will be based on utilising the WEKA tool to allow for the Machine Learning part of the application.

The Machine Learning is essential in that it produces the classifications that are to make the probabilities of the application work by creating a frequency matrix. This allows the Naive Bayes algorithm to use that training or historical data that the WEKA tools model has learned from. Then allowing the Bayes algorithm to utilise the probabilities of certain outcomes like for instance terms or words that we program into the WEKA tool.

Using these parameters the algorithm now has the foundation to make decisions, based on the probability that a certain term that has arrived as an email has the likelihood of being spam or not spam. This discernment is made possible by applying the machine learning to the initial training data which is used to train the model to know what a probable spam is and not spam.

With the use of WEKA it was decided to actually look into what that model does to the data, as the values of probability based upon the variability of terms that will appear as a function of spam. This effectively means that spam email will be the foundation of and further the vector matrix development which will use the Naive Bayes algorithm to decide the frequency of or the probability of two mutually exclusive variables.

The language that would be used to code this will be based on Python as well as PHP, the initial component will be coded in Python to code the probability of terms that will be used the second part will code in PHP the Naive Bayes algorithm that will be used in the server to mine the terms data. Finally there will be a UI coded in PHP as well which will allow a user to load and test emails to check if it is spam or non-spam.

4 Current Research

The Dataset that were used have been in circulation for many years and been a form of excellent spam email corpus. The Enron spam datasets are a culmination of emails form 150 individuals in the Enron Corp during the Federal investigation into the company. The emails were made public and have been used as a testing spam training systems from then onwards.

The Dataset itself has had a paper written about it introducing the Enron Corpus (Klimt, Ying 2004). In this report was highlighted the extent of the Dataset and the correlations of the data and users. Where the x-axis being the number of users and the y-axis denoting the number of message sin the log scale. In the report per user emails were said to have 757 messages with a total message count of 200, 399 a substantial amount of messages for a short period of time. It has been decided in the final sections of the report that the Enron Dataset is useful for email classification.

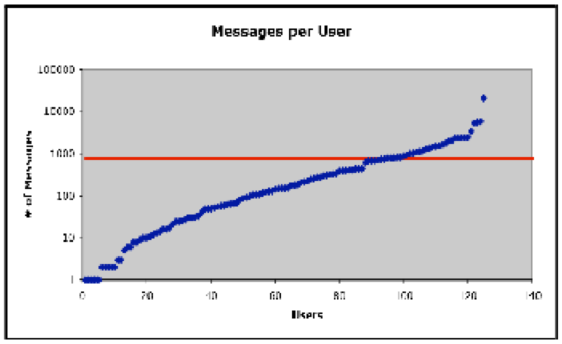

Fig. 1.

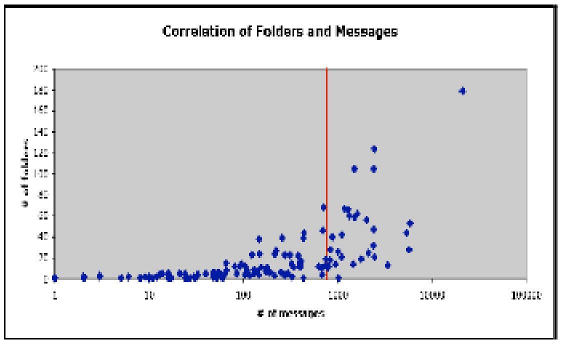

Fig. 2.

Figure 1: Enron Dataset corpus analysis

Therefore is was a case of choosing to utilise readily available data sets from educational sites that store spam emails in sample format to be used for training a spam filter for that express purpose. In using this type of dataset and by training the spam filter using this new spam mail it was possible to test the postulation of actually developing a spam filter by utilising basic Python and PHP.

Using sample spam email has been useful to train the filter to recognise spam and be able to detect it more readily. The implementation of this assisted in the machine learning part of the training regime can one train a filter to detect spam as readily as it would do so if used on a daily basis by any user. Similarly it has to be noted that the datasets that have been acquired have been tested before by other academic users and have found them to be of a good quality for use as a baseline spam dataset

5 WEKA

WEKA [1] has many functional uses as it has a suite of pre-process stages as well as classification steps that allow for the user to have complete control of their data using linear or regression algorithms.
The research will be based on the Naive Bayesian model that has been at work in the WEKA application. However it required a fundamentally large amount of Datasets of working spam emails or Datasets that would make the application viable as a testbed for the implementation of the Spam filter that has been devised.
After looking at WEKA in great detail it has been associated with a great many functions that require the coding of or the breakdown and analysis of. The main aim has been to develop a functional facsimile of WEKA without the complexity of the WEKA system. Therefore the main reason for using WEKA was to gain a quick prototyping of the application using PHP and MySQL. This type of prototyping would allow for the main features to be tested and optimised using a light weight framework without implementing a full and application.

5.1 WEKA Further Data

- Pre processing
- Classifying
- Prediction (nearest neighbour)
- Modelling

With these features WEKA has shown itself to be very scalable and being open source it has the added benefit of being improved upon by its creators. WEKA is coded in Java and is one of many platforms that data mine that are available today. For as much as it is free WEKA has shown a resilience in that it has acquired a very substantial following since its inception in 2000. When it was first released by the University of Waikato in New Zealand.

6 Dataset Creation

The initial stage of the dataset creation process is to read on the raw data form the Enron files and process them to a more readable format for WEKA. This is done by taking the raw data and running it through a Python script to reduce it down by eliminating delimiters and stop words.

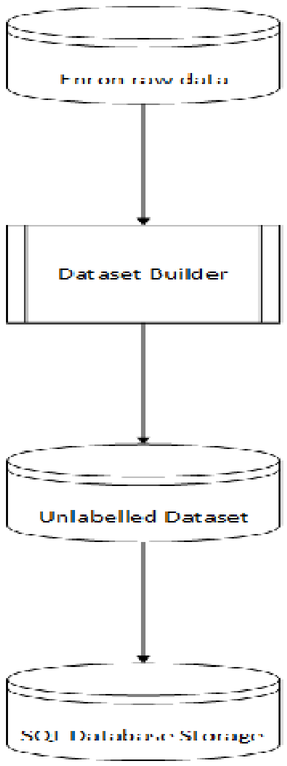

Figure 2: SDL Diagram for dataset creation

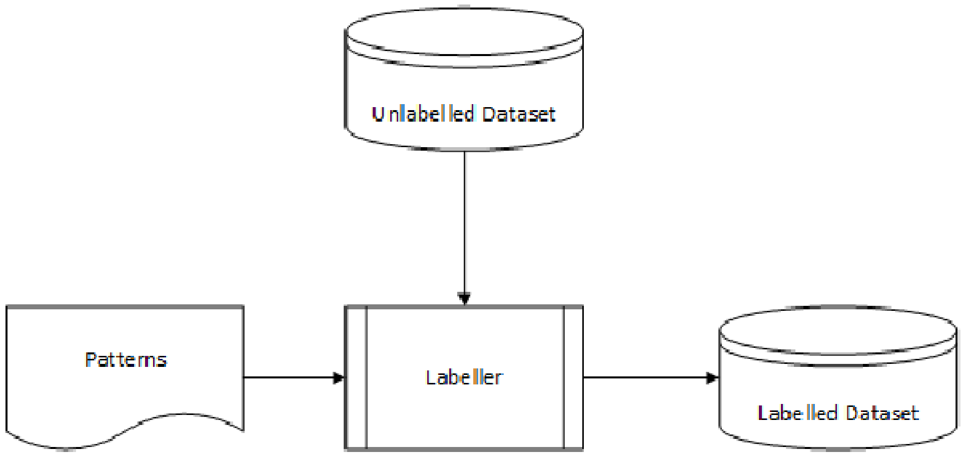

Figure 3: SDL Dataset labeler diagram

The data is unlabeled at first and will have to be labeled before it is uploaded to the SQL database. The SDL level schema is available to view .The stage now is to take the unlabeled data set and then to pre-process it via WEKA which will attach the probability to each term after it has engaged the pre-processing final stage.

However the pre-processing stage in WEKA will enable the feature extraction to occur and the model to be created that will be used to determine spam emails when compared against it.

7 Machine Learning Techniques

The question of Machine Learning is based on a very large subject area that has grown over the decades since it was first postulated and confirmed by many researchers looking in to neural networks like (Rumelhart &— McClelland, 1986). Also decision tree learners by (Quinlan, 1979) and CART (Breiman et al., 1984). They are few of the many researches that have been done over a few decades leading one to this massive subject are which is called Machine Learning. In essence it is the way algorithms extract data from certain types of resources be it like the web, online books, computer files and more.

By utilising in our case statistical analysis machine learning can be used to create classifiers or improve upon existing models that allow the process of machine learning to grow and continue as long as there is data there that required machine learning. The algorithms will continue to assimilate that data and in a way it can be used to provide accurate types of outputs when required by users or researchers.
As is described below in mathematical form for the Naive Bayes formula that have been used in the development of the classifier and thus the algorithm as well. That will feed off it and produce that accurate data extraction that will be required for the spam detector to function within the parameters that I have set out for it.

The essential components of Machine Learning are that outcomes or events can be readily identified by using terms like Predictors or variables, similarly in this application as is being doing it using Naive Bayes whereby by finding patterns in the data we can utilise this knowledge to classify other test datasets that can be matched against it. This is the feature of Machine Learning that using statistical modelling we can create a template or usable probability matrix that can be compared against other test documents.
To primarily isolate any patterns in the document and thus identify its class that in this case is either not spam or not spam. The essential tenet that has to be used is the Bayes documents classification problem. It lies with the Bayes formula which itself is probability equation on two probable outcomes that are independent of each other yet are intrinsically linked via a probability that if one event occurs then it would and might lead to the other outcome occurring as well.

7.1 Naive Bayes Theorem

With this in mind the Naive Byes is a simpler formula based on the premise that if a class which is either spam S or not S. Then it is right to assume it can be put into a Bayes type formula with probable outcomes dependent on the sum of probabilities,of the total number of classification that would be demanded in the form of.

$$P(D|S) = \prod_i P(w_i|S)$$

Now here we have the above formula consist of the S which states that the email is not spam. Now mathematically can it be possible to discern if the relative equation above will have the required field in it to define the not spam.

$$P(D|\neg S) = \prod_i P(w_i|\neg S)$$

This final formula above is the decision based on the variables that have been encountered in using the log likelihood ratio.

$$lnP(S|D)/P(\neg S|D) > 0$$

Defined as:

$$P(S|D) > P(\neg S|D)$$

Where if it is spam then then the above formula is valid, otherwise it is not valid and therefore is not spam. The main issues here are questions which arise based on the probability of the document having the benefit of being in either class S or S. The question has been to ask if there are two mutually exclusive classes for any given document then what will it mean for the main application.

The text mining application will be developed on the basis that it will be able to make a decision based upon probability factors the likelihood of a term that is part of a email or terms to be spam. It is therefore the a primary order of functionality that it would be coded in such a way as to show the possibility of how the Nave Bayes formula works in real life, It has been decided to use spam data that will be run through a UI to test the Bayes algorithm and its related functionality.

With the creation of a probability matrix that will hold the measure of the values of the outcomes that will define the nature of the probable outcomes. Once that matrix has been created it will be then up to the algorithm integration that will determine the values of the calculations using the Bayes algorithm. This determination will be based on the application of making the probability as easy as possible for the calculations to be made. However this determination only allows for a finite probability outcome and not multiple outcomes. This flaw will of course have to be dealt with in later chapters, as we look further into how we can take the calculations of the probabilities and seek to discern outcomes dependent on the values that they come by. Once this has been done it will be easy to make a linear calculation for the outcomes and not be too dependent on how they are worked out.

Now we can look in more detail of the Bayes theorem and its constituent parts and how it is made up and furthermore its interaction with the probability matrix development. The Bayes Algorithm is broken down into PHP and taking this is takes a value independently as it calculates the Naive Bayes system of probability matrix development. The algorithm has to be able to function with the parameters that are allotted to it, and be able to make the necessary calculations that will allow for the development of the matrix.

This custom spam filter will utilise the Naive Bayes theorem to allow for the reasoning and learning of the model. It has been developed for that sole purpose to see if the application worked with modified Bayes algorithms adapted for this express application processing. The essence of this application has been to make sure that it has the necessary training given to it to make it work correctly. The model will be based on utilising the WEKA tool to allow for the Machine Learning part of the application.

8 Spam Filter Applications

The current spam filters on the market are based on the Naive Bayes formula all apply the same functionality yet hold true to the equation that it is all based on. The main proponents are Google who have recently virtually guaranteed all Gmail users that they will be spam free. Which is questionable as of now as some spam still manages to get through their filters.
Google has invested heavily in the development of software in the fight against spam. A few years ago they acquired the UK based company called Deep Mind which was doing cutting edge research in artificial intelligence. It became apparent that the aims of the company paralleled their own and therefore they bought the company.
This was the first foray in the AI sector for Google and it has been noted in the press that this lead concurrently to the development of Google's spam application development. With the application that is being postulated in this report it is envisaged that it will be quickly prototyped using the Naive Bayes networks and WEKA as a rapid prototyping system to test the application functions before any real development is commenced.

9 Delimiters

In all the research papers there was a mention of stop word elimination There seemed to be an agglomeration of script that would eliminate stop words and replaces with white space or nothing at all. Stop word elimination was implemented to allow the parser to be able to read each term data with the interference of commas, full sops, or inverted comma or similar. This over complication could inherently led to a great many delays in the development to the application a script would need to be implemented to do the above.

9.1 Delimiter Solution

The alternative a new solution not seen in any of the papers researched was to use (hashtag) as a replacement to eliminate the stop words and commas. When scripting in PHP there was a function called str -replace which would allow the passage of a parameter in lieu of certain types of delimiters. This was passed in another parameter value as well and the replacement key value was actioned to replace those delimiters. This was a far more efficient way of resolving any Delimiter. Issues rather than lengthy code over complication. Adding the hashtag allowed for parser to ignore it and read the term data rather than any delimiters.

10 Project Aims & Objectives

10.1 Research Aims

With this in mind the problem statement was to design an algorithm to output the same response as the Naive Bayes formula. The proposed formula would be a summation rather than the product which is the classic Naive Bayes argument.

There will be one singular aim and that is to prove that the summation argument will produce the same filtering capabilities as the classical Vmap formula.

This custom spam filter will utilise the Naive Bayes theorem to allow for the reasoning and learning of the model.

The model will be based on utilising the WEKA tool to allow for the Machine Learning part of the application. This will allow one to carefully analyse the tool and to see how it functions and then model its functions as a spam filter and make an attempt to replicate its core functions. Without its more complex functionality but keeping its core functionality and adhering to the Naive Bayes theorem and equation in spam detection using conditional probabilities.

The Machine Learning is essential in that it produces the essential classifications that are to make the probabilities of the application work by creating a frequency matrix. This allows the Naive Bayes algorithm to use that training or historical data, that the WEKA tools model has learned from and allowing the Bayes algorithm to utilise the probabilities. Of certain outcomes like for instance terms or words that we program into the WEKA tool.
Based on these parameters the algorithm now has the foundation to make decisions on the probability that a certain term that has arrived as an email, it therefore has the likelihood of being spam or not spam. This discernment is made possible by applying the machine learning to the initial training data, which is used to train the model to know what a probable spam is and not spam.

With the use of WEKA it has been decided to actually code what that model does to the data and its prescribed values. The probability is based upon the variability of terms that will appear as a function of spam. This effectively means that spam email will be the foundation of and further the vector matrix development which will use the Naive Bayes algorithm. To decide the frequency of or the probability of two mutually exclusive variables e.g. like A or B in the Bayes formula.
The language that would be used to code this will be based on Python as well as PHP, the initial component will be coded in Python to code the probability of terms that will be used. The second part will code in PHP the Nave Bayes algorithm that will be used in the server to mine the terms data. Finally there will be a UI coded in PHP as well which will allow a user to load and test emails to check if it is spam or non-spam.

10.2 Framework Analysis

The framework is to provide the ability to:

1) The main aim of the Framework analysis is to take the main features of WEKA and learn from it after much experimentation. Analysing the main components of WEKA and how they are put together and also how they work as a whole. This will involve some time consuming work and a thorough understanding of the application and how it functions or how its higher functions work. In concert with deriving output data that is relevant as well as well as stable clearly indicating that the data mining has been a success.

2) After that analysis has been conducted then a requirements will need to be conducted that will clearly indicate the main components that need to utilise in the development of this own application. Then plan a schedule and a further plan on how to go about achieving that aim in terms of code level implementation. It has already been indicated that the main application will have the main core components that make up the WEKA these are the pre-processing stage as well as the classification stage. After that the derived output of can be output via a browser indicating the level of spam in the sample datasets.

3) Prior to this is the process of machine learning the rather subjective terms that has been applied to the process of whereby by virtue of the usage of the Naive Bayes formula. The training datasets that are used will train the spam filter and build the necessary classifier that will be used ultimately in the detection of spam. In the sample dataset that will be applied after the initial training run has been completed.

4) Once the process of training has been completed and the classifier has shown to be stable the sample data will be applied and then tested and its performance will be measured in the form of ROC curves. Which will indicate how the algorithm is faring with processing this new sample data.

5) Moreover it is known that when implementing any application one must have not only the baseline research but also have a comparator that will compare the results and expectations in comparison to an algorithm. That has been known to perform better due in part to the utilisation of regression algorithms. It will be a more linear implementation of the Nave Byes algorithm. Based on the conditional probabilities of the respective outputs using statistical analysis, it can thus derive a performance ROC curve of both algorithms and can then see how efficient they are.

10.3 Research aims using Bag of Words Model (BoW)

The bag of words or (BOW) system is a system of simplifying sentences into a multiset format whereby the order of the terms or words is not as important as the term or words itself.
What this means is that in an unsupervised learning method unstructured data is brought to the fore and hidden meanings are evaluated from its inference, processing the original raw data to a pre- processed format therefore when feature extraction is used.
It will ultimately use the Bag of Words model (BOW) to get the classifying model that will be used as the classifier. It has been surmised that word ordering will have a detrimental effect on spam classifiers as described in a paper by-Text Classification using Multi-word Features(Zhang, Yoshida, Tang 2007) . However this will not form the basis of the experimentation for this application development and will therefore not be looked into in more detail.

10.4 Training data experiments

The Training data will be compiled by extracting a set amount of terms in the region of 5000 terms from the spam Corpus. This will be done whilst using WEKA to pre-process the raw data into strings that can be easily read by any program taking away any and all forms of delimiters from the documents. This training data when extracted will form the basis of the classifying model and will be uploaded to the SQL database in its entirety.

10.5 PHP Function experiments

The search function will play a major role in the computation of the probability of spam in a dataset. Within PHP there are numerous functions that are readily available like str-char which calculates the accuracy of String matches.
This will find the last known position of a string and will return that position to the end of the string It takes a parameter value of string to search for and character to find This is followed by the other function similar- text which is passed on calculating the percentages of the similarity to get the best possible result for the return value this is the function chosen to match term data.

Similar text works on calculating the percentage of similar words and using a third argument will work out the accuracy of the match in percentage. The implementation of a PHP search function will be a required implementation as it will work alongside the computation of the terms in the dataset. In essence it will supplement the already theoretical processing of the algorithm and make it that much faster.

11 Framework

11.1 Requirements

The requirements analysis undertaken for this application are detailed below in exhaustive detail. This application specification will work on the understanding that current systems in pace are based on a premise of Naive Bayes being applied to a construct from a vector list of words to a matrix of words. Then their corresponding probabilities appended to them, this is what is called the classifier which computes the individual independent values for each word in the dataset.

The issue at hand is that there is no known system that can be compared to what has been proposed. As most Nave Bayes classifier are purely based on the findings in the research papers only allow for representing data in text format. This is a malleable format to display a classifier and can easily be read form with the right scripting and the data extracted and compared to model if a test email is spam or not spam. The core requirements will entail the following functionalities initially it will entail the training datasets to be used and the data extracted via the implementation and then stored in a database, the process flow will be as below:

- Using Python and PHP to code the mathematical formulas and other core algorithms
- Using MySQL Database to store the classifier data in from there the data can be accessed and modified and trained allowing for the machine learning aspects of the application.
- Using HTML to output the data as a UI in the browser

This simple process done for research purposes are all done in this way to fulfil the research criteria or question posed to it. However it has to be noted that requirements of the system will be that it is a spam detector and can filter and identify spam emails when tested.

Finally the application will be about how this algorithm functions and how it will allow for spam detection. The principle factor in this is speed and where both will be inversely proportional as speed increases, processing resource requirements will radically decrease as postulated or otherwise accuracy will stay stable in comparison. This is essentially the theory without the application in the classical system of looking for accuracy and precision in detecting spam emails, this application will make use of a string comparator in a summation algorithm. Instead of using complicated calculations for each word as described in the process flow below it is done once in creating the probability model.

With this you can now compare with the test email by comparing each word matching from the model and the term data from the test sample. The word is then appended to the map as well as the probability of spam and not being spam. This is then computed at the very end to sum up all the values and as we have postulated that spam will be of a higher posterior value then non spam.

Process Flow:

The most fundamental issue at this point is to highlight the classical nature of using a Naive Bayes classification system for spam detection. In this the process it involves creating a set of:

1) Spam and non spam emails Email Pre-processing - using the main bodies not header information Generating word lists using delimiters to generate word the word lists Training the spam email detector Involves computing the probabilities with the equation.

2) After that it is a case of saving the model to a Text File which will be compared later at the testing stage. The testing stage involves using the term data values which are used to determine the classification accuracy of the text classifier. This process is done by comparing the label value assigned by the classifier with the pre classified label value.

Testing Stage:

3) This process involves the pre classified and bundled together Generating Word lists as before the main bodies of the text are used and anything else is discarded Loading of the models this is the step where the classifier is now compared to the entry email. Generating word maps - when the comparison is run of the email to the classifier a map is built of the elements in the classifier against the testing data.

11.2 System Design

11.2.1 Programming Language

- There will be three core languages that will be used to code the application and will be Python, PHP, and HTML which will finally allow for a user requested output via the browser. Python is a powerful language and has the advantages of being very simple to implement especially mathematical based algorithms and machine learning frameworks.
- Then there will be the implementation of the PHP which will work on the basis or server construction or database construction which will house the output classifier and store in a database rather than a local one .Which is easily manageable and also scalable once the algorithms have been coded we now need a place to extract the feature data from the classifier and

store the relevant data in a location that will be easily accessible as well as secure.

- Finally there will be the HTML code which will work on extracting the classifier from the database and displaying it in readable form for the user.

11.3 Data Storage Medium

MySQL is a quite favourable storage system for storing the raw data of the classifier but when it comes down to flexibility and working on the dataset, it could never provide those facilities that a Hadoop server can provide via its capacity to handle big data[1]. Furthermore the viability of Hadoop is even more advantageous in that MySQL likes to have structured that it can deal with whereby data types are not pre-defined with Hadoop no such issues arise. Yet for the purpose of prototyping a SQL database will be used instead of the more desired Hadoop server.

12 Algorithm design

The algorithm will be a form of string search algorithm but will have the added advantage of being able to control and measure the values of the probabilities. Simply using a summation function to identify the likelihood of that test document being spam or not spam.

Moreover this algorithm will have to be functionally compliant with the current framework that has been put into place .The main aim has been to create a structure of the algorithm and then to script it to complete the task at hand. Furthermore it will have the benefit of being able to process large amounts of processing in succession.

Moreover coupled with the PHP word search function it will have the added processing power and speed to compute large datasets. The ideas that has been proposed in the introduction entails the determination of spam or the computation of spam using a summation method rather than a product method that has been used before. The concept has been the culmination of an intensive research to define the ideas that have evolved and to take the best of them and implement a workable program. That will take the computation and make the determination of the probability of spam or non-spam by using simple summation.

The code below details the construct and how it will work at the code level, ***as this has never been tried before it is not known if this will work to the efficiency required***. The PHP code depicts the exact form for the computation of the summation of the probability of individual terms in the model appearing in the test document. It will extract the full model form the SQL database and then run a further script which based on the function similar text will match each term in the test document to that of the model

Utilising a for loop to iterate through the array of terms. If a match is identified it will take that terms and push it to a table and also increment the value to a new variable which are either (sh) for non spam or (ss) for spam. Finally of course a comparison has to be made based on the Vmap equation that as spam is determined to be greater than non spam we can easily make a decision like this using the code below.

12.1 Algorithm design for Bench marking

Further to the development of the above algorithm research was carried out to identify the scope of bench marking the application overall. How to do that would clearly indicate the success or failure of this complicated project. Much theoretical research was carried out to this end and finally a determination was made that based on current findings in papers and research concluded was that the papers dealt with output of their code and trends of the applications.

There were some substantial papers on the code construct but they dealt with scripting in a framework, or doing it in R. Moreover a few papers provided pseudo-code to a certain degree but was inconclusive when researched further. In the end it was decided that it was better to build the bench marking application.

Essentially it would follow the Vmap equation which basically takes the product of all unique matched probabilities of their respective terms. It works by initially creating two arrays one which will take the input parameter of matched terms (mw). And the second array called (mwv) which will house and hold the comparator values. At this point the second stage filter will eliminate duplication by using a PHP function called in-array, which we will define as !in-array which basically means that it will check on the existence in any, of elements in that specific array.

Now will return false and then iterate over the array incrementing by +1 every time (mwv) which will contain the matched terms from the comparison of the test document. Of course if they are equal in the number of elements in each array, (mw) will then it will pass that data to the new matched array called (mwv). This is where it will finally be put through the array splice function which will if the two variables do not match will then remove any duplicate elements in that array.

The bench marking is scripted this way to ensure that only elements or terms that are uniquely matched with the test data will be isolated and extracted to enable the computation for the product to be made. Where it will take all probability values for each matched term and find the product. The largest total product value will then of course be compared to that of the non spam value to ascertain if the test document is actually spam or non spam. Essentially this is not a fully perfect system and there is variance in the computation when large datasets are applied to it. Resulting in smaller then normal exponential values being generated. This is of course beyond the remit of this project proposal and can be added as future work.

12.2 Data Flow Summary

The core requirements will entail the following functionalities initially it will entail the training datasets to be used to and the data extracted via the implementation and then stored in a database, the process flow will be as below.

- Using Python and PHP to code the mathematical formulas and other core algorithms.
- Using MySQL Database to store the classifier data in a SQL Database from there the data can be accessed and modified and trained allowing for the machine learning aspects of the application.
- Using HTML to output the data as a UI in the browser

12.3 Training data model

The training data model will be based on the 5000 key terms being rendered into the WEKA application for processing. This limit is arbitrary and there for open to increase if necessary. However it is assumed that this will provide a good source of training data for the classifier model.

13 Approach

13.1 Methodology

As can be seem form the equations the formalism involved in classifying any document (Zhang, Li -2007) in this paper the authors discuss the many approaches to calculating the using the chain rule the Naive Bayes classification solution as spam or non-spam. It is dependent on may factors once the vector has been created it is then a case of working out the conditional probabilities of the email.The spam detection we use then will attempt to calculate the spam or not spam by summations of each Class . Determining in which the email belongs to whereby by using Cj it will determine the maximum probability value of.

14 Implementation for Classical Vmap

Now we come to the implementation for the Nave Bayes document classification see (Zhang, Li 2007). In which they describe has many facets and many ways to develop the code or application based this type of algorithm. One proposal has been to allocate Python as the main computational tool as it facilitates mathematical based algorithms into code much more efficiently. It could use this language as a precursor to developing the first stage of the application.

1. The first stage will involve the dataset being made ready for pre-processing

2. Pre-processing begins with the main bodies being pre-processed using the delimiters as word separators

3. A word tokenizer is used to create a word list

4. The training of the spam detector is done by taking the Bayes Formula and applying it to selection of words - as cases of the words in the word list which will compute the total number of words in a corpus of documents. The word list and then this will take the as an example 5000 of the most prominent words from the word list and this will be computed using the formula.

5. Once this has been accomplished the training of the classifier which is in many aspects a form of Machine Learning this will form the foundation for your classifier that will be used for the classification of all test data applied to it.

Generally want the most probable hypothesis given the training data

15 Main Approaches for Classical Vmap formulation

As can be seem from the equations the formalism involved in classifying any document (Zhang, Li -2007) in this paper the authors discuss the many approaches to calculating the using the chain rule the Nave Bayes classification solution as spam or non-spam. It is dependent on many factors once the vector has been created it is then a case of working out the conditional probabilities. The email spam detection we used can then be utilised to calculate the category which an email belongs to. Whereby by using Cj which will determine the maximum probability value of.

$$P(C_j, C1, ..C_n)$$

From this we are then able to derive the decision rule that will be applied to the classifier to generate an output (Lee, Isa 2010) in this paper they discuss the many states of Naive Bayes computation. In which they proposed a novel solution to the classification problem, by looking at the density of the documents to be classified for each category. As a means to create a weighting system that can be applied automatically during computation. Furthermore the proposal also details another aspect of the text classification. That will compare and measure the terms in the matrix in a database by using a string search algorithm that will be devised for this purpose alone.

15.1 Mathematical Proofing for Classical Vmap formula

The Naive Bayes classification is based on Bayes theorem which was developed by Rev Thomas Bayes (1704 -1761).

The theory states that in text classification it will work on the basis of conditional independence whereby it will assume that each component of the theory is independent of each other.

$$P(w_i|D) = \frac{P(D|w_i)P(w_i)}{P(D)}$$

Bayes formula

$$P(w_j, C1, ..w_n)$$

For any word in a vector

$$P(,C,w_j,..w_n) = \frac{1}{(ZP(C)\prod_i P(W|C))}$$

Figure 4: Resolving join probability formulas

The way Naive Bayes functions is by assuming all variables are independent of each other given the category they are in.

The conditional probability of the theory is what makes Naive Bayes the simplest and most used text classification tool which in most cases is faster and more efficient than spanning trees or other forms classifications. In this paper a discussion be instigated that will take in the main sources of and works on email classification in use today and also the main principles of Naive Bayes algorithms. After that we will move to discuss the main principles of Naive Bayes by detailing the implementation of a classifier for spam detection. The main aim has been to base the formula on.

$$P(w_i|C,w) = P(w_i|C)$$

Now the joint probability can be written rewritten with limits.

$$P(w_i|C,w) = P(C)\prod_i(w_i|C)$$

Resolving join probability formulas

Detect in the initial proposal if there was a marked difference in the speed of computation for the normal Naive Bayes compared to that say the inverse term weighted algorithm. With this in mind the proposal has been in essence formulated to try those two main algorithms and see if certain differences arise due in part to the way the algorithms have been ***constructed***. This will essentially lead to the formulation of the final summation formula based on this research in differing algorithms.

Which can be reduced down further:

$$Posterior = \frac{Prior \times liklihood}{Evidence}$$

Resolving joint probability formula.

The above formula can then can be resolved by applying a constant.

Whereby :

$$V_{MAP} = \arg\max_{h \in H} \prod P(C_j)P(W|C_j)$$

Figure 5: Final formula for maximum likelihood

$$Z = P(w_j, ..w_n)$$

Now in Naive Bayesian algorithms to categorise an email in a given category CJ which will now attempt to get the maximum probability value of:

$$P(C, w_j, ..w_n)$$

Any given category for a given variable

Therefore in essence we can now call that value the Most probable target value see (Zhang, Li- 2007) as described in their paper as being denoted by (Vmap) which will map the maximum probability of any given variable being in a given category now denoted by (Vnb) or a reference to Nave Bayes classification algorithm.

15.2 Document Classification Model

Now we can look in more detail at another method that is useful in classifying documents in text classification. It involves the use Naive Bayes again as seen below. We assume a document will contain words and certain words will appear in no rigid order and the probability of an independent word will appear as:

$$P(w_j|C)$$

For any words in a given document probability

Now we can take this a step further and state the probability of a given document containing all the words in the above function will take the form:

$$P(D|C) = \prod P(w_i|C)$$

This equation can be further reduced to conclude when Naive Bayes is applied to give the output:

$$lnP(S|D)/P(\neg S|D) > 0$$

$$P(S|D) > P(\neg S|D)$$

Figure 6: It is spam if the above condition hold true and not spam if not

This forms the basis of the final document classification formula which will be used later on in the development of summation concepts. For spam computation also leading to more ideas into deriving an expression with different conditions than ascribed above.

15.3 The Summation Algorithm Proposal

The main impetus has been the determination of the above variance which in itself is very interesting yet has the added value of being already coded and applications constructed that have determined this already.
After much analysis by reading various papers on this subject it is apparent that a much better determination has to be accomplished that has the benefit of being original as well as simple in its scope. The intention has been to acquire initially a variance analysis of two differing algorithms, that have been researched with implementations looked at on a purely theoretical level.
It clearly indicated that the browser side was using the Naive Bayes document classification algorithm to take the maximum likelihood values. Then running a calculation based on that data.

It was envisaged a simple solution which will be described below it will not have to utilise the Naive Bayes algorithm on the testing side as well as the training side to compute accurate spam filtering. Which was used in the classical system and mentioned in the paper by (Zhang, Li- 2007) .
However in this proposal will use a simple character search algorithm which will be devised by the report author, the supplementary mathematical nature of this algorithm will take the form.It will just run comparisons of labelled emails against the classifier thus reducing the level of computation required at the testing stage.

15.3.1 Solution Overview

The proposal therefore has in hindsight been based on the numerous papers that have been written in the subject. It will be a culmination of taking those ideas to create a matching algorithm to match the integer values / long values with Strings/Character values. This data will then be saved to an array which will sum the totals and an inequality will be used to determine the document classification - either being spam or non-spam.

15.4 Methodology & Mathematical proof

The methodology will be based on the principle of how do we classify document content into spam or non-spam. If we take the premise that documents will be taken or be part of a myriad of classes. Therefore the nth terms of the document will be the probability of a words appearing in a random document from a certain class.

$$P(D|C) = \prod P(w_i|C)$$

Classical Naive Bayes document classification formula

The fundamental question that also needs to be asked here is how, to derive an answer to the question of working out the probability a given document belonging to a given class C. What has been neglected to be asked in addition to the above question is: Based on the probability of words appearing in any type of document, what is the total probabilities of any given word(s) matching the classifier being in those document(s)".

Since the assumption is that the training data is based on spam emails and non spam emails being classified and a matrix being created for it .Now we need to know quickly how to compute this from first principles we can add to the above function, by making asking the question but breaking it down into logical steps:

1. Based on the probability of words appearing in any type of document in any category.

$$P(D|C) = \prod P(w_i|C)$$

2.What are the total probabilities of any words matching the classifier being in those documents in any of those categories?

3.Where Spam = S, Non- Spam = not S, Document = D, C= Class either spam or not spam

With this in mind we modify the former function to this equation in Fig.7 quite simply and finally to incorporate the new premise. That former previous

$$P(S|D) = \sum_{i}^{n} P(w_i|S)$$

Figure 7: Summation by category of spam formula

equation will hold and will also hold for the new premise of the equation in Fig.7 as similar to the former equation. This has been done by substituting (C) for (S) where (S=spam) for now and defining two mutually exclusive classes which are spam and not spam. For this proofing we will assume ***ceteris paribas*** that spam is the desired output required for now. This substitution can now bring to light the formula in Fig.7 to a much clearer perspective.

16 Results

The data was gathered in the form of confusion matrices which will hold the required data for the occurrences of spam or not spam in a logical format. The aim was to generate an adequate number of confusion matrices to provide a good and through sample of the effects of the summation formula in comparison to the classical product formula.

A total of 1400 plus emails were manually tested against the summation formula and a further set of emails were tested for the bench marking which will equate to another 240 plus emails for that form of testing as well. The testing was done on the varying degrees of the similar-text percentages to gauge how the algorithm now that it had been implemented will function.
Furthermore an analysis will be undertaken to ascertain the levels of accuracy of the data gathered and how accurate is the algorithm as a whole when all the data is collated. As defined in Fig.5 the comparisons show the level for accuracy of each percentage value for similar-text, also misclassfication how often is it wrong and more.

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	112	156	268	False-Positive= 0.58
Actual((Spam)	79	36	115	True-Positive = 0.31

Table 1: Confusion matrix for PHP function at $\leq 50\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	39	182	221	False-Positive= 0.82
Actual((Spam)	165	26	191	True-Positive = 0.13

Table 2: Confusion matrix for PHP function at $\leq 70\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	27	161	188	False-Positive= 0.85
Actual((Spam)	176	35	211	True-Positive = 0.16

Table 3: Confusion matrix for PHP function at $\leq 90\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	95	6	101	False-Positive= 0.05
Actual((Spam)	5	94	99	True-Positive = 0.94

Table 4: Confusion matrix for PHP function at $\leq 100\%$

Confusion Matrix	**PHP Function Efficiency**			
	$\leq$ 50%	**$\leq$ 70%**	**$\leq$ 90%**	**$\leq$ 100%**
Accuracy	0.38	0.15	0.15	0.94
Prevalence	0.30	0.46	0.52	0.52
Misclassification	0.61	0.82	0.84	0.05
Specificity	0.41	0.17	0.14	0.94

Table 5: Confusion matrix

16.1 Help Section for Rate Calculation

This is a list of rates that are always computed form a confusion matrix they are essentially the key terms that are always searched for in computing the efficiency of any spam detector.

Accuracy: Overall, how often is the classifier correct?(TP+TN)/total.

Misclassification Rate: Overall, how often is it wrong?(FP+FN)/total equivalent to 1 minus Accuracy also known as "Error Rate"

True Positive Rate: When it's actually yes, how often does it predict yes? TP/actual yes also known as "Sensitivity" or "Recall"

False Positive Rate: When it's actually no, how often does it predict yes? FP/actual no

Specificity: When it's actually no, how often does it predict no? TN/actual no equivalent to 1 minus False Positive Rate

Precision: When it predicts yes, how often is it correct? TP/predicted yes

Prevalence: How often does the yes condition actually occur in our sample? actual yes/total

17 Review

All the values have been compiled data generated with the first instance of the PHP function being set at $\leq 50\%$.It is evident that this percentage is not really of any real efficiency in that regards the tables were a measure to test the accuracy of how the function would react to the percentage being changed to lower and higher values.
It is apparent that the rate set at over 90 percent will achieve the best result based on the current findings. This is of course against the labelled emails and therefore we are assured of its efficiency as the data set has all emails labelled to either spam or not spam.
Finally it has to also be noted that the higher the percentage the greater the efficiency of the string matching function. At this juncture it is possible to make an assertion that the highest percentage will yield the best results. However as the data clearly indicates the concept of the summation algorithm works in isolating spam in a large dataset.
Therefore based on the collated data it is a best fit to have the similar-text working at peak efficiency and by being set to over 90%. Looking at Fig.9 in the Accuracy graph the efficiency dips after 50% to plateau until 90% then increase substantially after that. This cannot be explained and is beyond the scope of this project However brings to light some interesting data on the similar-text outputs that can be looked into in more detail at a later stage.

17.1 ROC Analysis

The current data can be passed to ROC Curve analysis to ascertain the efficiency again of the spam computation accuracy. The data was compiled and analysed and graphs generated which correspond to the initial findings in that the efficiency of the more lower percentages are varied compared to the function set at 100% efficiency. This generates a very high accuracy for the spam filter at this point. In comparison all the other graphs were below the threshold which in essence means that their efficiency was substandard and below the normal threshold levels.

As can be extrapolated form the Accuracy graph in Fig.10 when the efficiency is set to 50% there is an overall decrease in efficiency as the higher the value is increased. There is in essence an inverse proportionality to the graph. This is most notable at 90% when it is at its lowest. Then after 100% it rapidly increases in efficiency as the string matches become much more accurate.

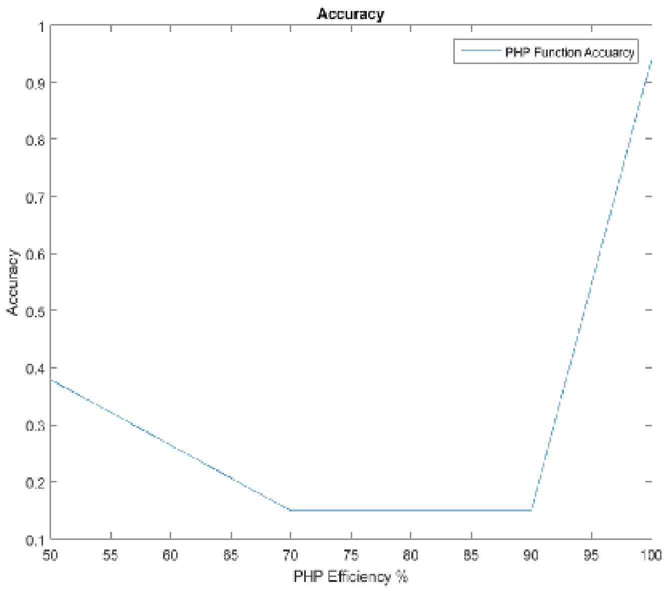

Figure 8: General Accuracy of PHP Function

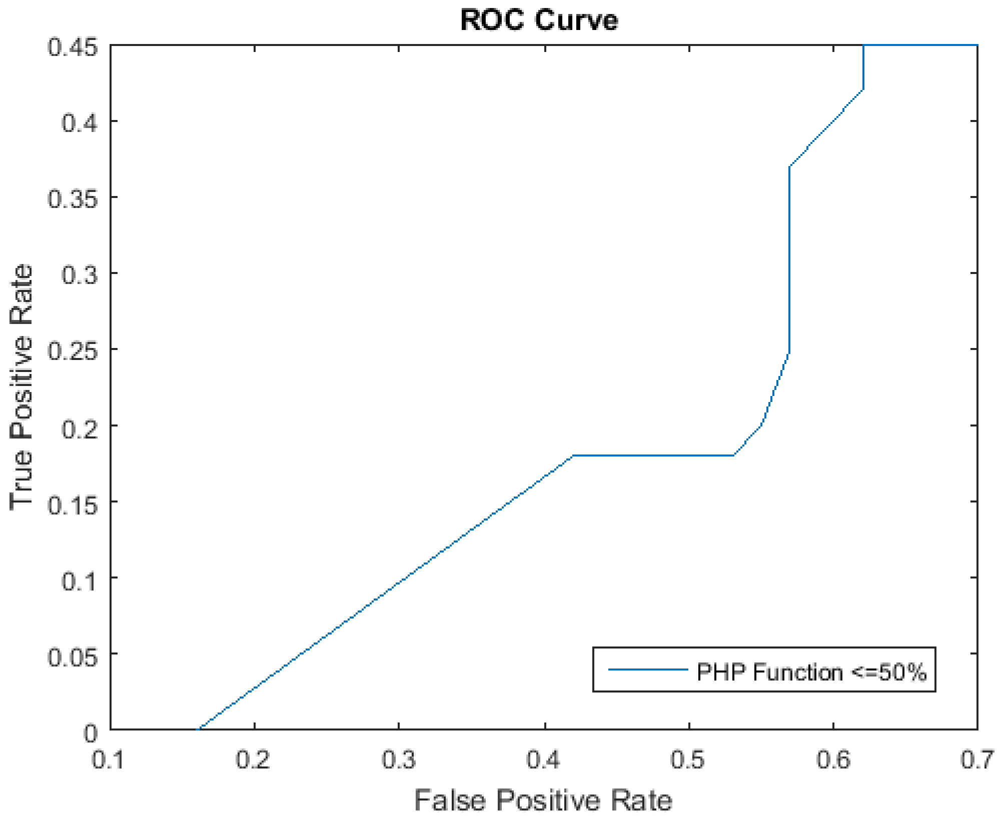

Figure 9: General Accuracy of PHP Function at $\leq 50\%$

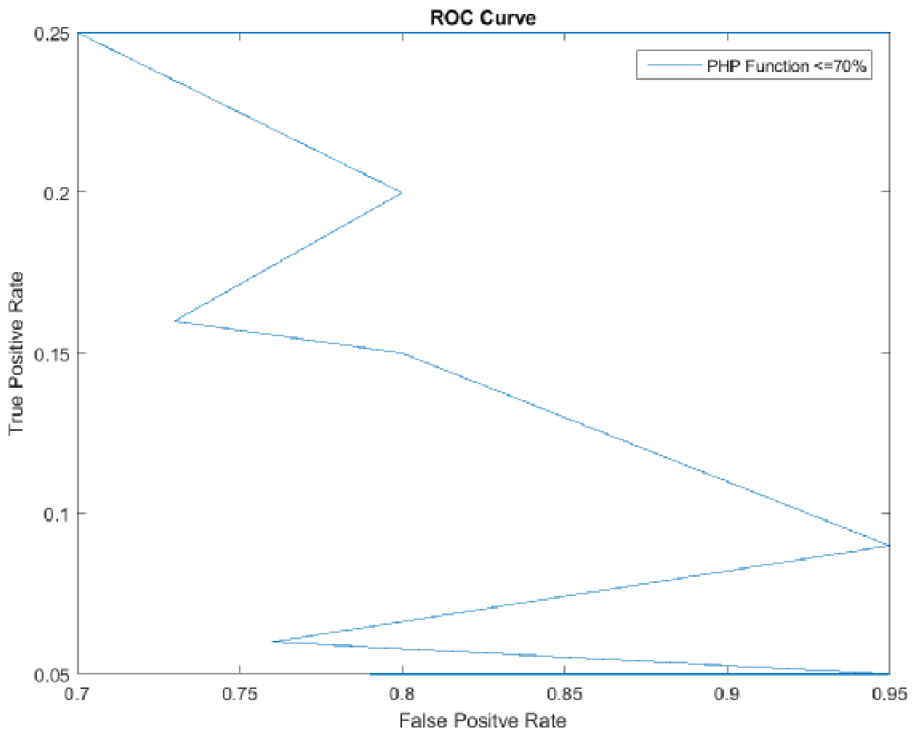

Figure 10: General Accuracy of PHP Function ≤ 70%

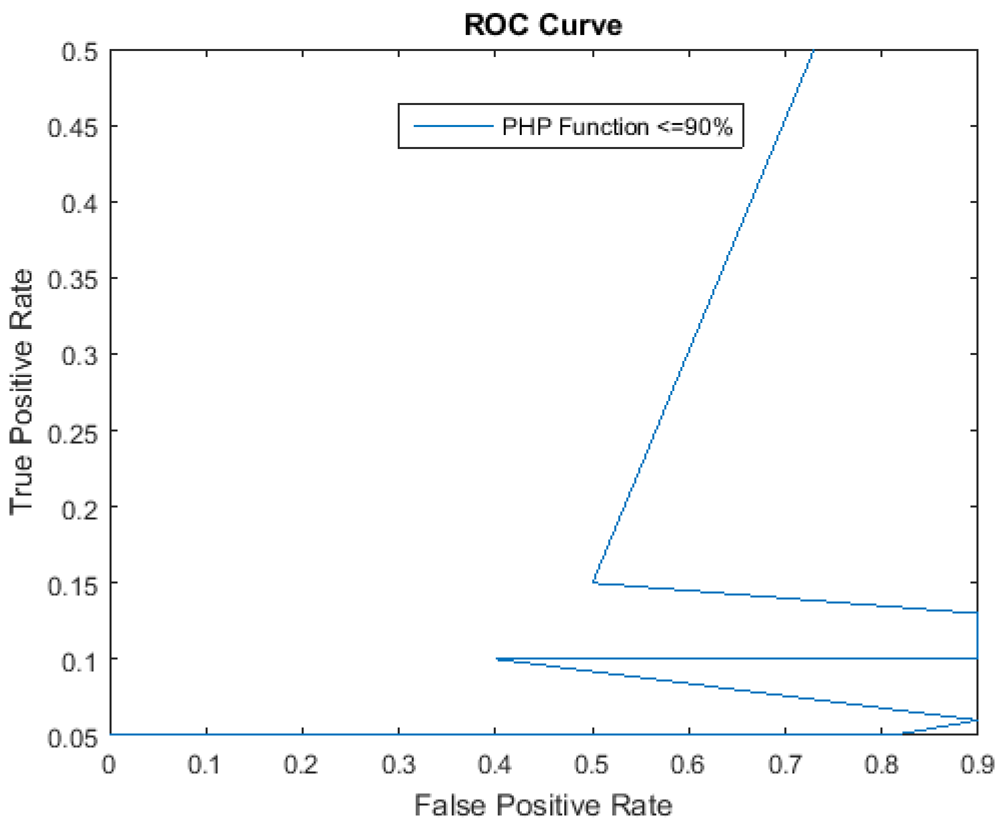

Figure 11: General Accuracy of PHP Function $\leq 90\%$

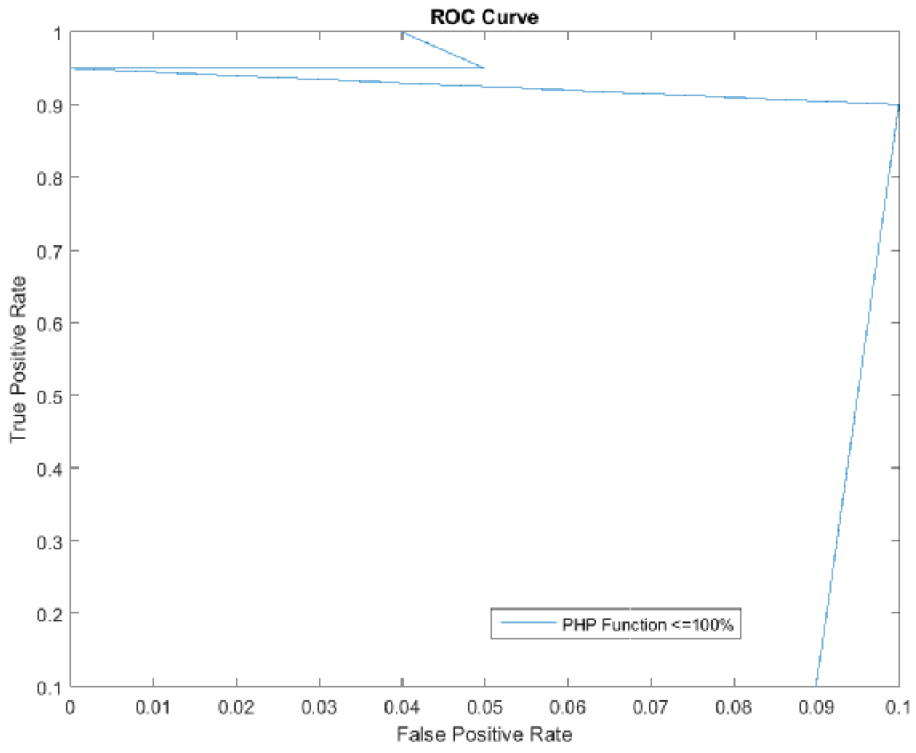

Figure 12: General Accuracy of PHP Function $\leq$ 100%

18 Bench Marking Analysis

The bench marking testing was to use the same level of spam datasets against it to check for accuracy and any inconsistencies.This was relative as whole analysis is based on the comparison of the summation argument in comparison to the product base argument. Which is more accurate and which is more viable. The same dataset was tested against this spam filter as well with the result below.

With the bench marking as a whole there were sudden anomalies in the computation at times as the smaller the values became through repeated multiplication. This resulted in errors of no answer at all as the computer was unable to define the value. Also it would give an accurate spam response but no numbered value. Yet again these are very interesting findings and ah=gain are beyond the scope of this project and perhaps with more time they can be looked into.

Moreover as with the analysis done for the summation the accuracy as well as misclassification rates were all calculated for the bench marking as well and can be viewed in Table.7.

Banch Mark Matrices	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	112	11	123	False-Positive= 0.08
Actual((Spam)	9	109	115	True-Positive = 0.92

Table 6: Confusion matrix for Bench Marking PHP function

Product Matrix	**BenchMarking**
	$\geq$ **90%**
Accuracy	0.91
Prevalence	0.48
Misclassification	0.08
Specificity	0.91

Table 7: Confusion matrix

The analysis gauged form the bench marking shows an indication of a highly accurate and efficient spam filter. It can be garnered from the data that the accuracy is very high indeed in comparison to the Summation formula it is of a similar vein. There is also a fundamental degree of similarity to the summation formula in that the rates of Misclassification or more simply how often the model is wrong is at a rate of almost 8%.
furthermore the rates of Prevalence which means how often that spam is located is at a rate of 48% which again follows the same lines and similarity to the classical product based formula. This comparison table clearly indicates the level of similarity in both formulas for the summation algorithm as well as the Product based algorithm as well.

Product Matrix	**Bench marking**	
	Summation Formula	**Product Formula**
Accuracy	0.91	0.91
Prevalence	0.48	0.52
Misclassification	0.08	0.05
Specificity	0.91	0.94

Table 8: Bench Marking comparison of Summation & Product formulas

19 Discussion

With retrospect to the project aims the overall aim has been to define and create a functional spam filter algorithm. That is on a similar par to the Classical product based algorithms that is used in a cross section of spam filters.
The code as well as the application clearly indicates a degree of accuracy with the product based algorithm in many respects. On a more mathematical note in any product based system where you are to multiply elements to such a degree - where the values are at time less than 1.
This causes issues with the sizes of the values gained when the computation has run its cycle, as the output value is always getting smaller and smaller. In contrast to the summation algorithm it can increase by summation but will not have the Floating point numbering limits. Which at that point the computer will find it much more difficult to derive a probable output representative of the inputs. This is another out of scope issue that can be further looked into at a later stage.

20 Future Work

Due to both the project and time constraints and the projects not so trivial problem statement. There were indeed many areas of future work that can be looked into. Areas such as Nano scale computing where if values becoming smaller and smaller what happens at that point.
Furthermore the anomalies garnered in the earlier findings can also be looked into as well - as to why the percentage settings for the similar-text function when set to a lower percentage value would have a higher accuracy for terms data matches than higher percentages.

These are areas more commonly that are very abstract as well as esoteric. At times when we discuss issues of fuzzy logic or Fractal based computing systems. There are also areas of comparison that can be looked into in regards to the Summation Algorithm as a whole. Furthermore it would need to be tested much more vigorously than what has transpired in this project.
Moreover now that a mathematical proofing has been clearly defined for this project. More research can indeed be undertaken to work out how this will impact spam filtering, and is it applicable in commercial terms and environments. This project has and always been a postulation or more formally a theoretical postulation on the intrinsic nature mathematics plays in computing. This has been the driving force to push the boundaries of certain ideas and concepts and to generate newer ones.

21 Conclusion

This project deployed a new type of algorithm defined as a summation algorithm for computing spam. After much analysis and mathematical proofing the formula was generated and comparisons made with respects to the classical based and product based algorithm. There were indeed very great similarities in the findings as well as in the levels of accuracy of the system as whole.

The overall data generated will go to prove that to a certain degree the new algorithm has some validity as well as rationale to be taken seriously. The similarity of operation also highlights what has been discussed in the above discussion area. Where the data was compared and similarities arose for each algorithms output data.
However the discussion about how mathematically as very small probabilities are computed or their products computed their final values inherently get smaller. Therefore the argument is whether summation is better on a mathematical basis to have some viability of use in the future.
Further on form this the research done on this has generated more questions than answers at certain stages and this is quite unprecedented as well. Certainly on the usage of the similar-text anomalies which were discussed before as well as the further anomalies in the bench marking stage. These are all very curious and often times intriguing in the developments but can be looked into as further study.

22 Appendices

22.0.1 Confusion Matrices for PHP function at $\leq$ 50%

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	20	15	35	False-Positive= 0.42
Actual((Spam)	6	5	11	True-Positive = 0.45

Table 9: Confusion matrix for PHP function at $\leq$ 50%

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	16	22	38	False-Positive= 0.57
Actual((Spam)	6	2	8	True-Positive = 0.25

Table 10: Confusion matrix for PHP function at $\leq$ 50%

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	15	17	32	False-Positive= 0.53
Actual((Spam)	6	5	11	True-Positive = 0.45

Table 11: Confusion matrix for PHP function at $\leq$ 50%

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	12	20	32	False-Positive= 0.62
Actual((Spam)	8	0	8	True-Positive = 0

Table 12: Confusion matrix for PHP function at $\leq$ 50%

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	12	15	29	False-Positive= 0.55
Actual((Spam)	8	5	13	True-Positive = 0.38

Table 13: Confusion matrix for PHP function at $\leq$ 50%

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	7	17	24	False-Positive= 0.70
Actual((Spam)	13	3	16	True-Positive = 0.18

Table 14: Confusion matrix for PHP function at $\leq 50\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	11	18	29	False-Positive= 0.62
Actual((Spam)	9	2	11	True-Positive = 0.18

Table 15: Confusion matrix for PHP function at $\leq 50\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	10	14	24	False-Positive= 0.16
Actual((Spam)	10	6	16	True-Positive = 0.37

Table 16: Confusion matrix for PHP function at $\leq 50\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	11	15	26	False-Positive= 0.57
Actual((Spam)	9	5	14	True-Positive = 0.20

Table 17: Confusion matrix for PHP function at $\leq 50\%$

22.1 Confusion matrix for PHP function at $\leq$ 70%

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	1	19	20	False-Positive= 0.95
Actual((Spam)	19	5	20	True-Positive = 0.05

Table 18: Confusion matrix for PHP function at $\leq 70\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	1	19	20	False-Positive= 0.95
Actual((Spam)	15	15	20	True-Positive = 0.25

Table 19: Confusion matrix for PHP function at $\leq 70\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	6	14	20	False-Positive= 0.7
Actual((Spam)	14	6	20	True-Positive = 0.25

Table 20: Confusion matrix for PHP function at $\leq 70\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	4	16	20	False-Positive= 0.8
Actual((Spam)	16	3	20	True-Positive = 0.15

Table 21: Confusion matrix for PHP function at $\leq 70\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	5	16	20	False-Positive= 0.8
Actual((Spam)	15	4	0.2	True-Positive = 0.38

Table 22: Confusion matrix for PHP function at $\leq 70\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	6	19	25	False-Positive= 0.76
Actual((Spam)	14	1	0.06	True-Positive = 0.18

Table 23: Confusion matrix for PHP function at $\leq 70\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	5	21	26	False-Positive= 0.80
Actual((Spam)	19	1	20	True-Positive = 0.05

Table 24: Confusion matrix for PHP function at $\leq 70\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	1	21	22	False-Positive= 0.95
Actual((Spam)	19	2	21	True-Positive = 0.09

Table 25: Confusion matrix for PHP function at $\leq 70\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	5	17	23	False-Positive= 0.73
Actual((Spam)	15	3	18	True-Positive = 0.16

Table 26: Confusion matrix for PHP function at $\leq 70\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	5	19	24	False-Positive= 0.79
Actual((Spam)	16	1	17	True-Positive = 0.05

Table 27: Confusion matrix for PHP function at $\leq 70\%$

22.2 Confusion Matrix for PHP Function at $\leq 90\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	4	18	22	False-Positive= 0.81
Actual((Spam)	16	2	18	True-Positive = 0.11

Table 28: Confusion matrix for PHP function at $\leq 90\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	5	19	24	False-Positive= 0.79
Actual((Spam)	15	1	16	True-Positive = 0.06

Table 29: Confusion matrix for PHP function at $\leq 90\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	2	18	20	False-Positive= 0.4
Actual((Spam)	18	2	20	True-Positive = 0.1

Table 30: Confusion matrix for PHP function at $\leq 90\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	2	2	4	False-Positive= 0.5
Actual((Spam)	18	18	36	True-Positive = 0.5

Table 31: Confusion matrix for PHP function at $\leq 90\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	0	17	17	False-Positive= 0.0
Actual((Spam)	20	3	23	True-Positive = 0.13

Table 32: Confusion matrix for PHP function at $\leq 90\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	2	19	21	False-Positive= 0.90
Actual((Spam)	19	1	20	True-Positive = 0.05

Table 33: Confusion matrix for PHP function at $\leq$ 90%

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	2	18	20	False-Positive= 0.9
Actual((Spam)	18	2	20	True-Positive = 0.1

Table 34: Confusion matrix for PHP function at $\leq$ 90%

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	4	16	20	False-Positive= 0.9
Actual((Spam)	18	2	20	True-Positive = 0.1

Table 35: Confusion matrix for PHP function at $\leq$ 90%

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	3	17	20	False-Positive= 0.73
Actual((Spam)	17	3	20	True-Positive = 0.15

Table 36: Confusion matrix for PHP function at $\leq$ 90%

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	3	19	22	False-Positive= 0.86
Actual((Spam)	17	1	18	True-Positive = 0.05

Table 37: Confusion matrix for PHP function at $\leq$ 90%

22.3 Confusion Matrix for PHP Function at $\leq$ 100%

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	19	1	20	False-Positive= 0.05
Actual((Spam)	1	19	20	True-Positive = 0.95

Table 38: Confusion matrix for PHP function at ≤ 100%

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	18	2	20	False-Positive= 0.1
Actual((Spam)	2	18	20	True-Positive = 0.9

Table 39: Confusion matrix for PHP function at ≤ 100%

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	2	18	20	False-Positive= 0.4
Actual((Spam)	18	2	20	True-Positive = 0.1

Table 40: Confusion matrix for PHP function at ≤ 100%

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	19	0	19	False-Positive= 0.0
Actual((Spam)	11	20	21	True-Positive = 0.95

Table 41: Confusion matrix for PHP function at ≤ 100%

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	20	1	21	False-Positive= 0.04
Actual((Spam)	0	19	19	True-Positive = 1

Table 42: Confusion matrix for PHP function at ≤ 100%

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	19	2	21	False-Positive= 0.09
Actual((Spam)	1	18	19	True-Positive = 0.10

Table 43: Confusion matrix for PHP function at ≤ 100%

22.4 Confusion Matrix for PHP Function for Product computation at $> 90\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	19	1	20	False-Positive= 0.05
Actual((Spam)	1	19	20	True-Positive = 1

Table 44: Confusion matrix for PHP function at $\leq 90\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	18	3	20	False-Positive= 0.15
Actual((Spam)	2	18	17	True-Positive = 0.89

Table 45: Confusion matrix for PHP function at $\leq 90\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	19	1	20	False-Positive= 0.05
Actual((Spam)	1	19	20	True-Positive = 0.95

Table 46: Confusion matrix for PHP function at $\leq 90\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	19	1	20	False-Positive= 0.05
Actual((Spam)	1	19	20	True-Positive = 0.95

Table 47: Confusion matrix for PHP function at $\leq 90\%$

Confusion Matrix	**Predicted**			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	18	1	20	False-Positive= 0.05
Actual((Spam)	2	19	21	True-Positive = 0.90

Table 48: Confusion matrix for PHP function at $> 90\%$

Confusion Matrix	Predicted			
	Non-Spam	**Spam**	**Total**	**Rates**
Actual(Non-spam)	19	4	23	False-Positive= 0.17
Actual((Spam)	1	16	17	True-Positive = 0.94

Table 49: Confusion matrix for PHP function at $\leq$ 90%

22.4.1 Meetings and Milestones

Meetings

November 3 meetings initial concepts and declaration of intent. Discussions ranged from initial spam concepts base don PHP to algorithm comparisons.

December 3 meetings planning started and requirements drawn up with regards to the key ideas of the spam filter.Preparation of Literature Review was discussed in detail and its implementation.

January 2 meetings Literature review discussed as well as key ideas on how to go about coding application. By this time pivot to develop an algorithm was declared.

February 1 meeting coding started on application and resultant feedback sent to supervisor.By this time a full suite of requirements and plan had been drawn.

March no meetings by this time the latter stages of the code development was taking place and resulting in a proposed demo in April.

April 1 meeting demo done but still had bugs in the system these were isolated and fixed.

23 References

1. Mendoza, M. (2012).A new term weighted Scheme for Nave Bayes text Categorization, International Journal of Web Information Systems, 2012: Emerald Insight.

2. Harish BS, Hedge M, Meghana M. (2012).An empirical study on various text classifiers, Mysore, 2012. Dept. of Computing Science: SJ college of Engineering.

3. Satake,E, Murray, A. (2015).an alternative teaching method of conditional probabilities and Bayes rule : An application of Bayes Rule, Emerson College, Boston MA USA, 2015: Teaching Statistics

4. Burduk, R. (2011).Imprecise information in Bayes classifier, Dept. of Computer Networks, 2011.Poland: Springer.

5. Isa, D, and Lee, H. (2010) .Expert systems with Application: Automatically computed document weighting facility for nave Bayes classification, Malaysia, 2010.Elsevier.

6. Nguyen T, Chang K, Hui S. (2012).Supervised term weighting centroid-based classifiers for text categorization, Singapore, 2012. School of Computer Engineering: Springer.

7. Last name, FM. (Year published). Title of Paper or Proceedings, Title of Conference, Location, Date. Place of publication: Publisher.

8. Song Y, Kolcz A, Giles L. (2009).Software Practice and Experience: Better nave Bayes classification for high precision spam detection, 2009. Dept. of Computer Science Pennsylvania State University USA: Wile & Sons.

9. Qiang G,(2010).An effective Algorithm for improving performance of Nave Bayes for text classification, Second international Conference on Computer Research and Development, Higher vocational College, Shanghai, China, 2010. : IEEE the Computer Society.

10. Sathyadevan S, Athira U Sarath PR Anjana V. (2014). Improved Document classification through enhanced Nave Bayes Algorithm, 2014 International Conference on Data Science & Engineering (ICDSE), 2014.Amrita cyber Security Kollam India.

11. Gao F, Zhang F. (2011).An improvement to Nave Bayes for text Classification, China, 2011.Moe Klinns Lab Xi a Jiaotong University: Elsevier.

12. Puerta J, Jose P. (2010).Improving the performance of Nave Bayes multinomial in email foldering by improving distribution based balance of datasets, Expert Systems with Applications, Spain, 2010.Universidad de Castilla- La Mancha: Elsevier.

13. Harris Robert F , Deshpande V. (2007).An Evaluation of Nave Bayesian Anti-Spam Filtering Techniques, Proceedings of the 2007 IEEE Workshop on Information Assurance United States Military Academy West Point, USA, 2007.

14. Mcallum A and Nigam K. (1998). A comparison of event models for Nave Bayes text classification, Proceedings of the international conference on Machine Learning. Workshop on learning for Text Categorization, Madison WI, USA.

15. Lewis D and Ringuette M. (1994). A comparison oftwo learning Algorithms for Text Categorization, Proceedings of the international Annual Symposium on Document Analysis and information Retrieval Las Vegas NV USA .

16. Vikas P and Robert Erbacher F, Chris Harris (2007). An evaluation of Nave Bayesian spam filtering Techniques, Proceedings of the international IEEE Workshop on Information alliance USA.

17. Wei Zhang, Feng Gao J (2011). An improvement to the Nave Bayes for Text Classification. 2011 SciVerse ScienceDirect.

18.Cmuedu. (2016). Cmuedu. Retrieved 1 May, 2016, from https://www.cs.cmu.edu/ . /enron/In-text citation: (Cmuedu, 2016).

19.Dataschoolio. (2014). Data School. Retrieved 1 May, 2016, from http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/In-text citation: (Dataschoolio, 2014).