Tweaking Naïve Bayes classifier for intelligent spam detection

Ankita Raturi[1] and Sunil Pranit Lal[2]

[1] University of California, Irvine, CA 92697, USA.
araturi@uci.edu
[2] School of Computing, Information and Mathematical Sciences,
University of the South Pacific, Laucala Campus, Suva, Fiji
lal_s@usp.ac.fj

Abstract. Spam classification is a text classification task that is commonly implemented using Bayesian learning. These classification methods are often modified in order to improve the accuracy and minimize false positives. This paper describes a Naïve Bayes (NB) classifier for basic spam classification. This is then augmented with a cascaded filter that uses a Weighted-Radial Bias Function (W-RBF) for similarity measure. It is expected that the NB classifier will perform the basic classification with the W-RBF acting as a secondary filter, thus improving the performance of the spam classifier. It was found that the NB portion of the cascade was the initial spam filter with the W-RBF filter acting as a False Positive filter.

1 Introduction

With the advent of the Internet, email has become one of the popular applications for communication. However there is growing problem of spam email which unnecessarily clogs the global network infrastructure and carries profound economic implications. Spam generally appears in the form of bulk unsolicited email in the user's inbox, that can be phishing attempts, scam letters, or even just annoying, unsolicited advertisements of goods and services. Spam filtering is an attempt at sheltering the user from the murky world of unwanted emails selling and scamming drugs and love and rock and roll, while at the same time ensuring that the user enjoys the benefits of electronic communication with other users that they wish to be in contact with. When a user accesses their email, they would ideally like to only receive legitimate mail, good emails that are solicited. The variety of spam, however, is great and ever changing and so methods to combat spam also need to continuously evolve [1, 2].

Spam filtering is a staged process, whereby first emails undergo feature extraction, where the words of interest are extracted from the content of the email. These are the tokenized and represented in a certain manner, to be presented to a classifier [3]. A dataset of tokenized emails is then collated and presented to the classifier for training and testing purposes. Spam classification is a text classification (TC) task [4–6]. Based upon the content of an email, the classification is then made of whether the email is spam or a good email.

2 Design Inspirations

2.1 Paul Graham's Plan for Spam

In August 2002, Paul Graham posted an essay on his website titled A Plan for Spam that detailed the use of a Naïve Bayes classifier for spam classification [1]. His piece popularized the use of Bayes theorem in spam filtering and set the baseline for design, implementation and expectations for spam filters. This essay was particularly influential in approaches to spam classifiers as it gives the reader an immense appreciation of the entire process involved in spam filtering as well as the potential for NB classifiers. Graham's essay succintly explains how a seemingly simplistic statistical approach can be applied effectively to classification.

2.2 Better Naïve Bayes Classification

Giles et. al [7] describe several modifications on a Naïve Bayes classifier. They propose three major differences: the use of a correlation measure, normalization of the emails during classification and a cascading of NB classifiers. The correlation measure they implement, the Absolute Correlation Ratio, introduces a supervised term weight on the probabilities. Inspiration was drawn from the cascaded classifier method, which has been incorporated in our design as two stage filter. In the cascaded filter the first stage weeds out initial spam, with the focus of the second stage on improving performance of the classifier in other aspects such as reduction of false positives.

2.3 Weighted Radial Bias Function

Sharma and Paliwal [8] introduce the idea of applying a weighted radial basis function (W-RBF) to NB classifiers for the purpose of masquerade detection. It essentially weights the NB probabilities with a similarity measure that is calculated by comparing the testing instances with previously trained instances. We therefore draw on the notion of incorporating W-RBF within our NB Classifier for improving the accuracy of spam classification.

3 Naïve Bayes Theorem

Classification tasks can be performed by various learning algorithms, however the Naïve Bayes (NB) classifier has been immensely popular. This is primarily due to it's low computational complexity, ease of implementation and good performance [9]

The NB Classifier is a basic probabilistic classifier based on Bayes Theorem. It assumes class conditional variable independence and has a linear complexity [10]. Bayes Theorem is as shown in (1).

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} \tag{1}$$

There are two concepts that are in play: H which is the hypothesis, and E, the evidence, or data. The aim is to calculate the probability, given certain evidence, that a particular hypothesis is true. Therefore in a spam classifier, it can be said Bayes theorem is used to find the likelihood that a particular email is spam, given the contents of the email. If the evidence presented to the classifier is in vector form and there are multiple hypothesis possible, (1) can be written as (2) below. The product of the individual probabilities allows for conditional independence among the individual attributes. In addition, as the denominator of (1) would be the same when calculating the probability of all the hypothesis for a particular evidence, it can be discarded.

$$P(H_j|\boldsymbol{E}) = P(H_j)\prod P(E_i|H_j) \tag{2}$$

Now in order to classify emails, the posterior class membership must be calculated for all classes; so both the probability that an email is good and spam. In order to make the final classification, NB classifiers utilize a Decision Rule [7]. A probability ratio decision rule such as the following:

$$\frac{P(spam|email)}{P(good|email)} < \theta \tag{3}$$

This ratio can be referred to as the Bayes Factor [10], and used in comparison with a threshold θ for classification decision purposes.

4 Weighted Radial Bias Function

The Weighted Radial Basis Function approach [8] has a weighted component, W in conjunction with the RBF function which is a similarity measure between the training and testing vectors in question. W-RBF is comprised of these two concepts and can be summarized in (4) below:

$$\lambda(\nu, a_j) = \mu(\nu b, ab_j)\kappa(\nu, a_j) \tag{4}$$

where, λ acts upon ν, the training vector to be compared with, a_j, the testing vector. μ is the binary weight associated with the vectors and κ is the Gaussian RBF applied to the two vectors, that is influenced by the frequency of the vector components. Here νb and ab_j refer to the binary representation vectors of the training and testing email vectors.

4.1 The Binary Similarity Weight: μ

In order to calculate μ, a binary representation of the frequency vector of all emails has to be created. That is, if a frequency of a particular word in that email vector is greater than 0 (that is, the word exists), it's respective binary value would be 1, and similarly, if the word value is 0, the binary value is 0. Using these binary representations, the binary similarity measure is calculated using the formula below.

$$\mu(\nu b, ab_j) = \frac{\sum_d and(\nu b, ab_j)}{\sum_d xor(\nu b, ab_j)} \tag{5}$$

4.2 The Gaussian RBF: κ

Sharma and Paliwal [8] discuss both a symmetrical and an asymmetrical similarity measure. They define the symmetrical similarity measure as:

$$\kappa(\nu, a_j) = exp\Big(-\frac{1}{2}\Big[\frac{||\nu - a_j||^2}{||a_j||^2} + \frac{||\nu - a_j||^2}{||\nu||^2}\Big]\Big) \tag{6}$$

It analyzes the similarity between the training vector ν and testing vector a_j. In their work, they use the Gaussian RBF to compare a single training vector (as an example of a masquerade block) to all the test vectors. So there would only be a single final λ value for each test vector.

5 Experimental Dataset

The testing and training data are obtained from the UCI Machine Learning Repository [11], the "Spambase DataSet". It is a collection of 4601 emails, of which 1813 are spam and 2788 are good emails. Each email vector is as follows:

- 48 word frequencies – These are in the form of term frequencies within each email. They are normalized values, so for the frequency of a particular word in a particular email is divided by the total number of words in that email.
- 6 character frequencies – the frequencies for these characters are similarly normalized.
- 3 run values that keep track of the longest, average and total capital letter runs. These values were discarded during the experiments.
- 1 classification value – as previously decided during creation of the dataset. This allows for supervised learning and error checking. The entire dataset was split using a 70:30, training to testing corpus ratio. Therefore the Training Corpus consisted of 1269 spam emails and 1951 good emails. The Testing Corpus consisted of 544 spam emails, 837 good emails.

6 Classifier Designs

6.1 Simple Naïve Bayes Classifier (NBC)

In the case of spam classification, the evidence being presented to the NB classifier is a frequency vector of the 48 words and 6 characters. There are two hypothesis available for classification, the email is either spam, or the email is a good email. In this case calculation of the posterior class membership [7] is as follows:

54 components in the email, so $i = 54$

Let x_i^d be the frequency of each word in email x^d such as $< x_1^d, x_2^d, \dots x_i^d >$

2 classes: spam, good. So c = 2

Let θ_j be all the classes available

$$P(\theta_j|\boldsymbol{x^d}) = P(\theta_j)\prod_{i=0}^{54} P(x_i^d|\theta_j) \tag{7}$$

Thus, for example, to calculate the posterior class membership of an email of the class spam, it would be as follows:

$$P(spam|\boldsymbol{email}) = P(spam)\prod P(word|spam) \tag{8}$$

For each email, $P(spam|\boldsymbol{email})$ and $P(good|\boldsymbol{email})$ would need to be calculated.

A straightforward NB Classifier was designed and implemented in Java. The procedure involves preprocessing (Fig. 1) in preparation for training (Fig. 2), which is followed by testing (Fig. 3) described as follows:

1. Training:
   - Total frequency for each word is calculated, in addition to other basic corpus statistics such as total spam and good email counts. These are required for the calculation of later probabilities.
   - Calculate posterior class membership using (8) for each email vector, for class Spam and class Good.
   - Use a decision rule, in this case the same as (3) to classify the email vector.
   - Once a decision is made on all email vectors in the training corpus, check the accuracy of all the class predictions. Based on this accuracy, adjust the threshold accordingly.
   - Retain the “best so far” value of the threshold, depending on the maximum classification accuracy yielded.
   - When training is complete, the classifier now has a threshold value that it expects will yield the highest accuracy.

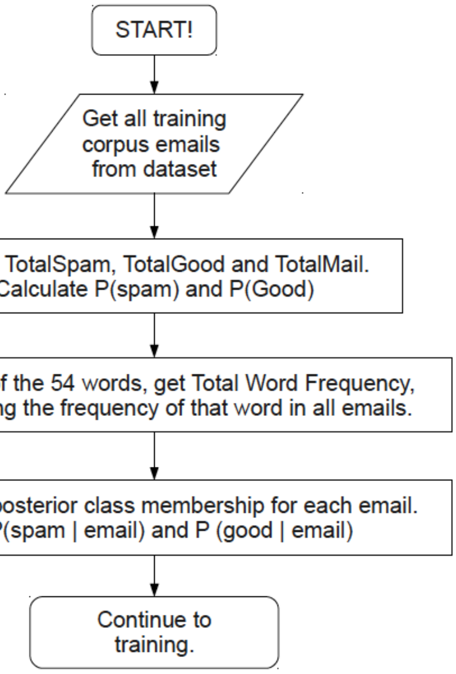
START!
Get all training corpus emails from dataset
Count TotalSpam, TotalGood and TotalMail. Calculate P(spam) and P(Good)
For each of the 54 words, get Total Word Frequency, by summing the frequency of that word in all emails.
Calculate posterior class membership for each email. Ie: P(spam | email) and P (good | email)
Continue to training.

Fig. 1. Preprocessing for training only

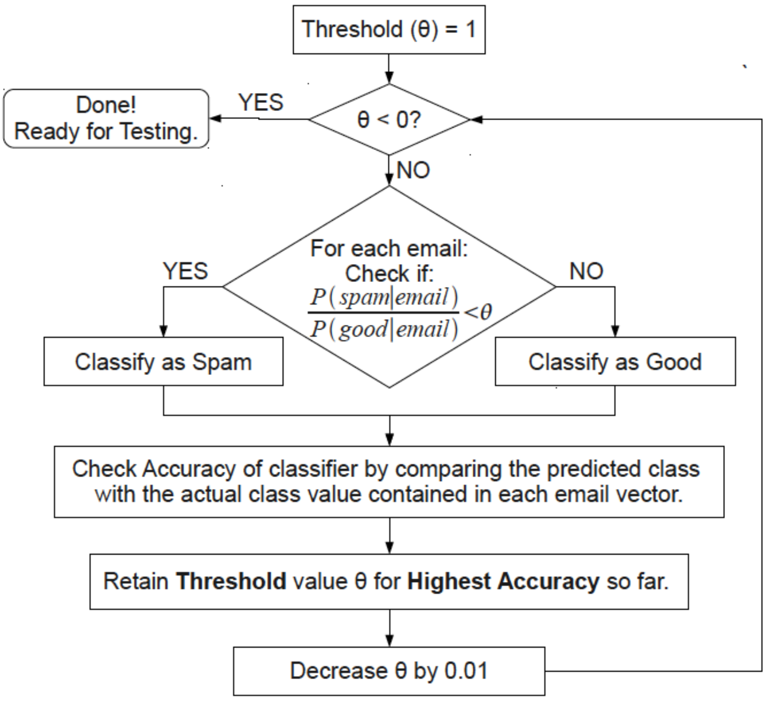
Threshold (θ) = 1
θ < 0?
YES
Done! Ready for Testing.
NO
For each email: Check if: P(spam|email) / P(good|email) < θ
YES
NO
Classify as Spam
Classify as Good
Check Accuracy of classifier by comparing the predicted class with the actual class value contained in each email vector.
Retain Threshold value θ for Highest Accuracy so far.
Decrease θ by 0.01

Fig. 2. Overview of the training process

2. Testing:
   - The threshold is set as the "best so far" value that was derived during training. This is no longer altered.
   - For the email vector to be tested, the posterior class membership is calculated.
   - The decision rule of (3) is invoked to allow for a classification to be made.
   - Results are reported for all email vectors that were tested.

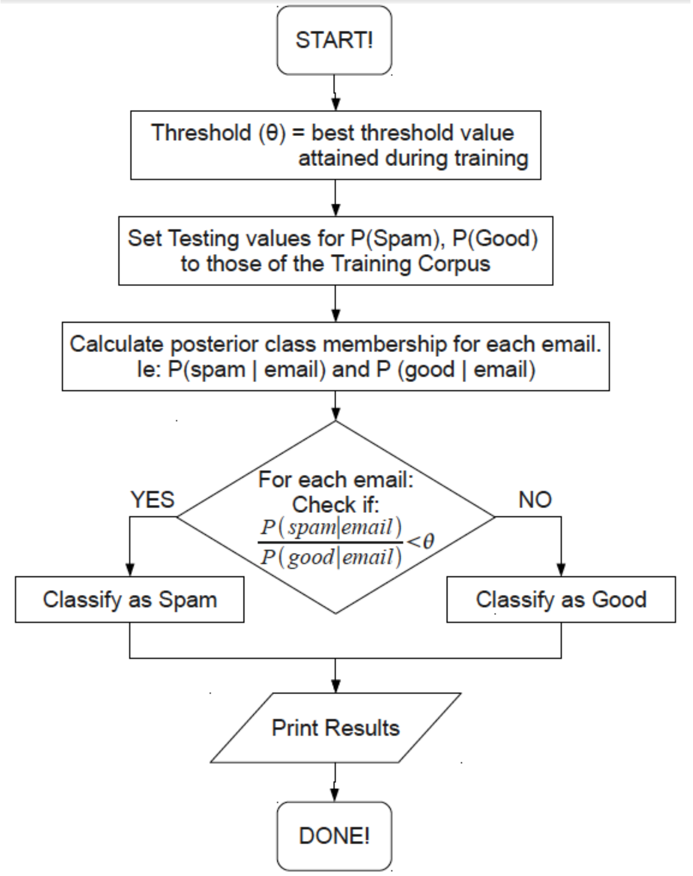
START!
Threshold (θ) = best threshold value attained during training
Set Testing values for P(Spam), P(Good) to those of the Training Corpus
Calculate posterior class membership for each email. Ie: P(spam | email) and P (good | email)
For each email: Check if: $\frac{P(spam|email)}{P(good|email)} < \theta$
YES
NO
Classify as Spam
Classify as Good
Print Results
DONE!

Fig. 3. Testing process

6.2 NB Classifier with Cascaded W-RBF filter (NBC + FP Filter)

This cascaded classifier follows the same general flow as the simple NB classifier. The difference lies in the decision rule during *testing* time only. Here, if the decision rule of the NB classifier suspects the given email as spam, then W-RBF filter is invoked.

In the work of Sharma and Paliwal [8], they utilize the W-RBF method in conjunction with an NB classifier where this method is in itself a weight upon the NB posterior class membership probabilities. However, we implement it as a discrete feature of the classifier that is cascaded like what is proposed by Giles et. al [7] and discussed in [12]. This W-RBF method can therefore be said to be used as a False Positive (FP) Filter (Fig. 4).

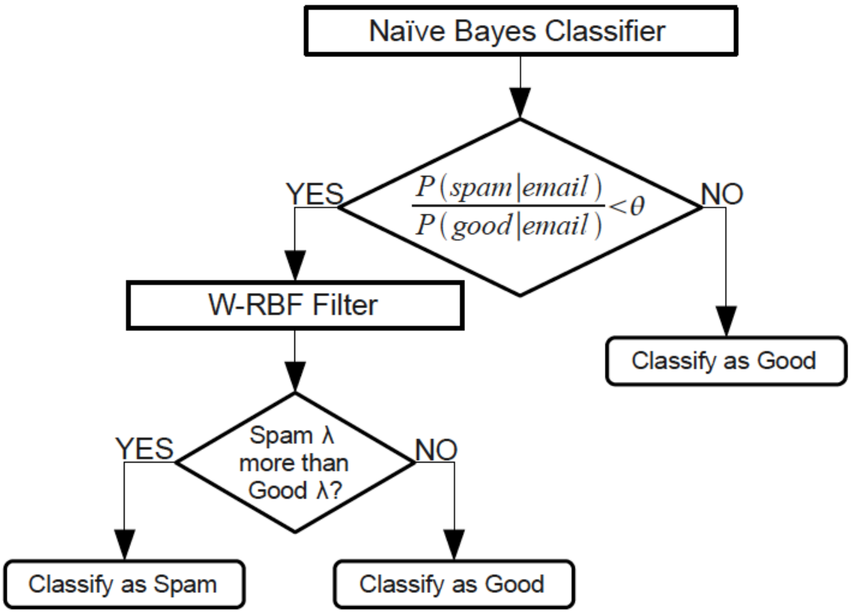

Fig. 4. Cascaded NB classifier with W-RBF filter

The filter works as follows:

- W-RBF filter is invoked whenever NB Classifier classifies a particular email vector as spam.
- The email vector is compared with every single training corpus email vector (1 × 3220 comparisons) via invocation of the Similarity function.
- This results in a total of 3220 λ values for each test email vector calculated using (4), (5) and (6). The filter then picks the best (maximum) λ values of all the spam emails and another group of best (maximum) λ for all the good emails the test was compared with. These are the two neighbourhood sets to be considered. Experimentation was performed to decide upon the optimum number of neighbours to consider. The final λ values for each set of neighbours (Good and Spam) were calculated by finding the average λ value (Sum of values / Total number of values). These were designated as the finalized Max Good λ and Max Spam λ values.

- The Similarity function results, Max Good λ and Max Spam λ, are then compared for a final classification to be made.

7 Experimentation Results using UCI Dataset

The dataset was run through both classifiers which will henceforth be referred to as the NBC and NBC + FP filter. The training phase was the same for both classifiers as the first step of the NBC + FP filter was also to perform basic Naïve Bayes classification with the threshold obtained.

7.1 NBC

Figure 2 captures the training process for tuning the NB classifier by varying the threshold θ within the range $[0, 1]$.

The optimum threshold found was $\theta = 0.12$ for the Naïve Bayes portion of the classifiers. During the training phase, this threshold classified 2962 out of 3220 training emails, with an accuracy of 91.99%. This threshold achieved the results in Table 1 for the NBC tests which translates to 1269 out of 1381 test emails (not seen by the classifier before) being correctly classified. This is supported by the Receiver Operating Characteristic (ROC) curve (Fig. 5).

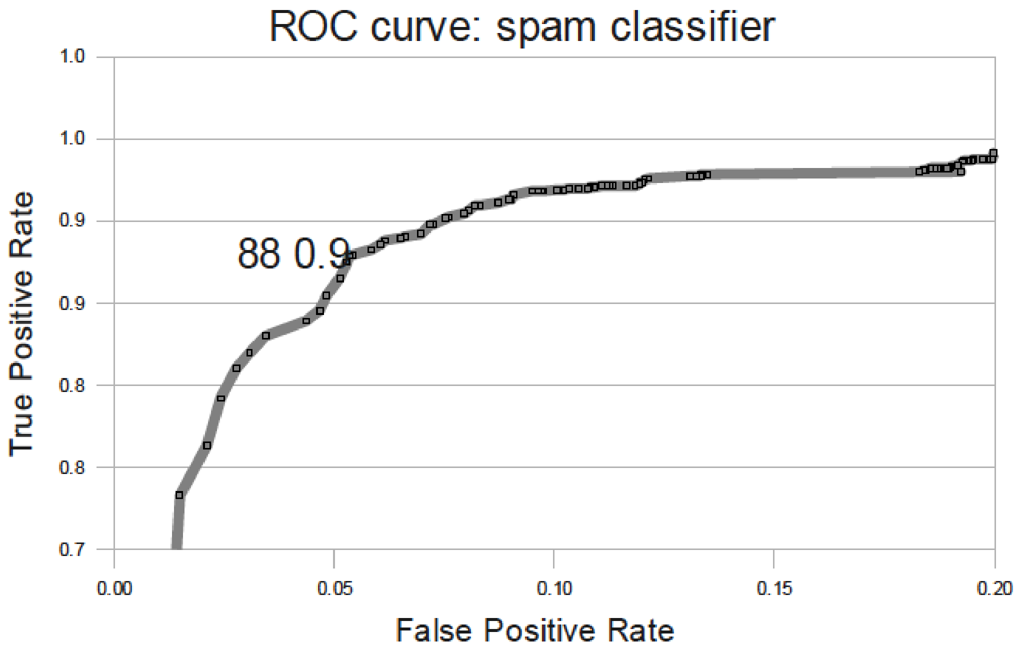

Fig. 5. ROC curve for the simple Naïve Bayes spam classifier. The optimum threshold ($\theta = 0.12$) was obtained at the point marked Run 88 with a $TPR \approx 0.9$

7.2 NBC + FP Filter

The aim of the FP filter is to minimize the false positive rate. This would occur after majority of the spam has already been weeded out by the first stage Naïve Bayes classifier.

In order to decide upon the optimum number of neighbours to be utilized by the W-RBF based FP filter, the relationship between the false positive rate and the number of neighbours being considered was investigated (Fig. 6). The testing data was run through the filter 15 separate times, with value of neighbours to be considered being incremented gradually from 1 to 15 neighbours.

It was found, that using one neighbour was optimal. That is, only the maximum λ value in the array of 3220 λ values was to be utilized.

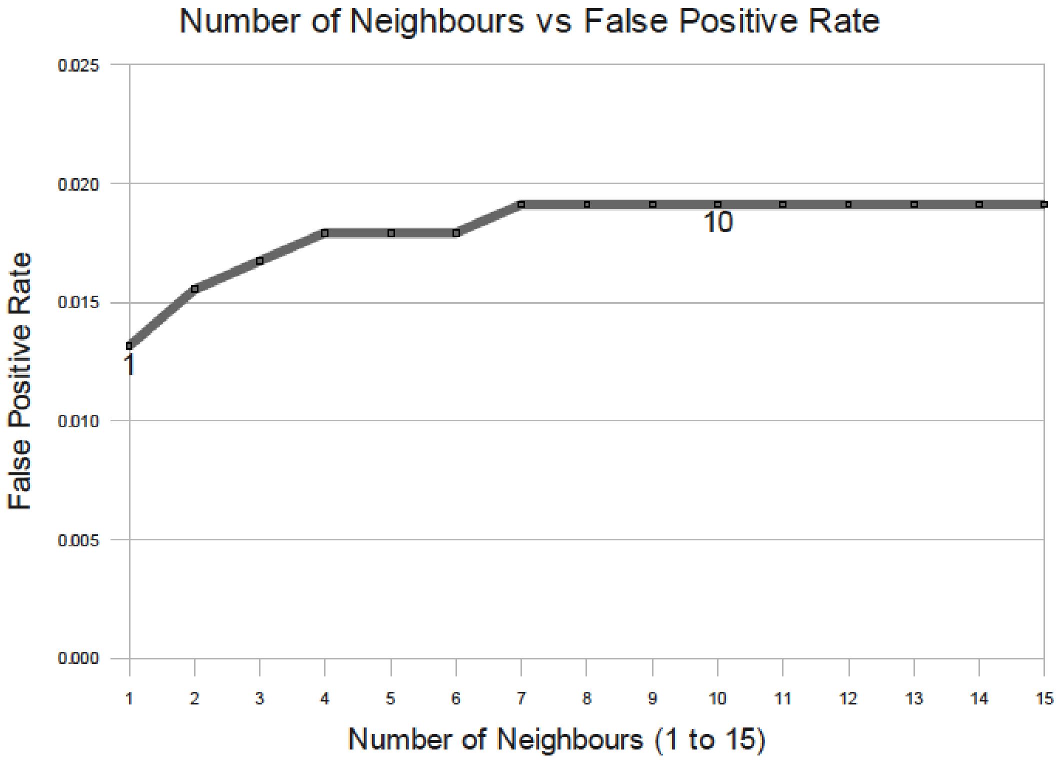

Fig. 6. Investigation of the effects of the number of neighbours being considered on the false positive rate

7.3 NBC vs NBC + FP Filter

Table 1 is a summary of the accuracy, false positive rate (FPR) and true positive rate (TPR) obtained during the Testing of both classifiers. Here, the FPR is calculated via (9) and the Accuracy via (10). The TPR is a measure of the ratio of Spam accurately classified [13], which is described in (11).

$$FPR(\%) = \frac{FalsePositives}{TotalGood} \tag{9}$$

$$Accuracy(\%) = \frac{TruePositives + TrueNegatives}{TotalMail} \tag{10}$$

$$TPR(\%) = \frac{TruePositives}{TotalSpam} \tag{11}$$

For comparison purposes, the NBC and NBC + FP filter were run using the same training and testing data. The training portion of both classifiers were identical. The NBC + FP filter was tested using 1 neighbour. The results in Table 1 are after the classifiers were trained and then subjected to testing data it had not previously encountered.

Table 1. Comparison of the two classifiers during testing using $\theta = 0.12$.

Classifier	Accuracy (%)	FPR (%)	TPR (%)
NB classifier	91.89	2.4	95.38
NBC + FP filter (1 neighbour)	91.02	1.3	97.51

8 Conclusion

The aim of this work was to successfully implement a Naïve Bayes Classifier and the improve upon the performance of this classifier. There are several aspects to classification that were tweaked. After a implementing a basic NB classifier that worked on the initial spam filtering, a cascaded False Positive filter was incorporated into the design to ensure that good emails were not lost due to misclassification. It is often stated in the field of spam filtering that it is better to get 100 spam than to lose 1 good email [1, 2].

Thus, while the NBC+FP filter had a minimal decrease (- 0.87%) in accuracy as compared to the plain NB Classifier, the false positive rate was cut in almost half, from 2.4% in the plain NBC to 1.3% in the NBC using the FP filter. In addition, the true positive rate of the NBC+FP filter, that is, its ability to catch spam, increased by 2.27%.

It would be interesting to investigate different types of cascaded classifiers using the concept of multi-stage filter, with the aim of improving both the accuracy and reducing the FPR. Further work could also include the investigation of the effects of different forms of term weighting, similarity measures and formal neighbourhood algorithms on the performance of the Naïve Bayes Classifiers.

References

1. P. Graham. ”A Plan for Spam”. Internet: `http://www.paulgraham.com/spam.html`, Aug 2002 [Apr, 2011].
2. J.A. Zdziarski. Ending spam : Bayesian content filtering and the art of statistical language classification. (1st ed). San Fransisco, CA: No Starch Press (2005)
3. W. M. Camhinas, T.S. Guzella. ”A review of machine learning approaches to Spam filtering”. Expert Systems with Applications. vol 36 (7), pp. 10206-10222 (2009)
4. D.D. Lewis. ”Naïve Bayes Text Classification for Spam Filtering” in ASA Chicago Chapter Spring Conference, Loyola Univ. (2007)
5. R. Rizvi. ”Spam Filter ; Using Data Mining techniques to counter Spam”. Internet: `http://raza-rizvi.blogspot.com/2010/03/creating-spam-filter-usingnaive-bayes.html`, Mar 2010, [Apr 2011]
6. V. Metsis, I. Androutsopoulos, G. Paliouras. Spam filtering with Naïve Bayes Which Naïve Bayes? in The Third Conference on Email and Anti- Spam (2006)
7. C.L. Giles, A. Kolcz, Y. Song. ”Better Naïve Bayes classification for high-precision spam detection”. Software-Practice and Experience. vol 39(11), pp. 1003-1024 (2009)
8. A. Sharma, K. Paliwal. ”Detecting masquerades using a combination of Naïve Bayes and weighted RBF approach”. Journal in Computer Virology, vol 3(33), pp. 237-245 (2007)
9. T. Mitchell. Machine Learning. (1st ed). McGraw-Hill Science/Engineering/Math, pp. 154-184 (1991)
10. W. M. Bolstad, Introduction to Bayesian Statistics. Hoboken, NJ: John Wiley & Sons, Inc, pp. 62-108 (2004)
11. A. Asuncion, A. Frank. UCI Machine Learning Repository Internet: `http://archive.ics.uci.edu/ml` University of California, Irvine, CA [Apr 2011].
12. S. Alag. Collective Intelligence in Action. (1st ed). Greenwich, CT: Manning Publications Co., pp. 173-306 (2009)
13. T. Fawcett. ROC Graphs: Notes and Practical Considerations for Researchers. Kluwer Academic Publishers. (2004)