# Spam Detector Project Report

Ganapathy Raman Madanagopal

October 12, 2014

## 1. Summary

The project is about developing and implementing an algorithm which can classify the mails as a spam or a ham. Given a set of mails which contains both ham mails and spam mails, when this set of mails are passed as the input to the detector program the detector program classifies the each mail either as a spam or ham based on the algorithm (set of conditions).

In phase 1, a rule based spam detector was developed, which classifies the mail either as spam or non-spam based on certain rules. A well-know 250 blacklisted words which frequently occur in spam mails are collected and stored in a bag-of-words. The detector program check if the email contains any of these black-listed word in it. If the mail has black-listed word(s), then the mail is marked as spam mail, else the mail is marked as ham mail.

In phase 2, a spam detector program was developed which classifies a mail as a spam or a ham not just based on the occurrence of the black-listed words. Because the black-listed words can also occur in ham mails. To overcome this drawback, a new algorithm which follows Naïve Bayes classification method [1] was implemented. Initially the detector is trained with a sample set of mails which contains both spam and ham mails. Also each time when the word repeats the number of occurrence of that word is incremented by count one. Finally, two list (one for spam words and other for ham words) along with occurrence count of the words are stored in two hash list (dictionary) [2]. Then probability of a two classes are calculated as follows

$P_r(y/x) = P_r\left(x_1, x_2, \ldots, x_p \middle| y\right) * P_r(y)$

$P_r(y/x) = \prod_{i=p} P_r(x_i|y) * P_r(y) \rightarrow Naïve\ Bayes\ assumption$

Then the maximum likelihood class can be determined as

$\hat{\mathrm{Y}}\mathrm{arg} = \mathrm{max}_{\mathrm{y} \in \{\text{'spam'}, \text{'non-spam'}\}} \mathrm{Pr(y|x)}$

And calculation are made in log range. And from the probabilities obtained if the threshold is greater than 0.7 then the mail is marked as spam, else ham. Finally the total number of spam and ham mails are displayed. Also for the detector was tested against various test sets. TPR and FPR are calculated and ROC Graph [3] was plotted.

In the spam detector logic was implemented in Python language [4] [5] and tested against mails in (.txt) format.

## 2. Classifier Code

Algorithm 1: Simple Spam Detector

```
Global Declarations:
1: var:  bagOfWords:List<String> {};  → given a list containing 400 black-listed words
                                         that frequently occur in spam mails
2:       spamCount: int;              → contains total spam mails count
3:       hamCount: int;               → contains total ham mails count
4:       directoryPath: String;       → given a path to the directory which
                                         contains all the mails
Procedure initiate()
/* Initialize global variables */
5: spamCount  := 0;
6: hamCount   := 0;

Procedure main()
/* This the function which is invoked on the beginning of the program */
7: readAllFiles(directoryPath);
8: printSummary();

Procedure readAllFiles(directoryPath)
/* This is function reads all files in the given directory and call spam detector function*/
9:   var: fo: FileObject;                     → file object for file related operations
10:       mail: String;                       → string to store the contents in a mail
11: for all the files in the directoryPath do → reads all the files in the directory
12:       fo := open(file,'readMode');        → open the file for reading
13:       mail := fo.read();                  → read the entire contents in the mail
14:       isSpamOrNot(mail);                  → call the function isSpamOrNot to detect.

Procedure isSpamOrNot(mail)
/* Function is to detect whether the give file is a spam or not */
15: var: spamFlag: Boolean;                   → flag variable to denote
                                                the mail is spam or not
16: spamFlag := 0;
17: for word in mail do                       → for all words in the mail content
18:       for blackWord in bagOfWords do      → for all words in the bag of words
19:             if word == blackWords then    →if word is in black-listed word then
20:                   spamFlag = spamFlag + 1;
```

```
21: if spamFlag != 0 then
22:     spamCount := spamCount + 1;          → increase the spam count
23: else
24:     hamCount := hamCount + 1;            → increase the ham count
```

**Procedure printSummary()**

```
/* The function is to print the summary */
25: print 'Number of mails classified as Spam:', spamCount
26: print 'Number of mails classified as Ham:', hamCount
```

Algorithm 2: Spam Detector using Naïve Bayes Detector

**Global Declarations:**

```
1: var:  spamList:HashTable<String,int> {}; → List containing spam words and a count for
                                               the corresponding words, which denotes
                                               the number times the particular word has
                                               occurred in spam mails
2:       hamList:HashTable<String,int> {}; → List containing ham words and a count for
                                               the corresponding words, which denotes
                                               the number times the particular word has
                                               occurred in ham mails
3:       spamTrainingFilesCount: int;   → contains total spam mails count in trainer
4:       hamTrainingFilesCount: int;    → contains total ham mails count in trainer
5:       spamWordsCount: int;           → contains the total words in spamList
6:       hamWordsCount: int;            → contains the total words in hamsList
7:       spamSum: int;                  → total sum of the (hash keys) words occurrences
8:       hamSum: int;                   → total sum of the (hash keys) words occurrences
9:       directoryPath: String;         → given a path to the directory which
                                           contains all the mails
10:      spamCount: int;                → number of mails detected as spam in tester
11:      hamCount: int;                 → number of mails detected as ham in tester
12:      threshold: float;              → threshold value for classifying spam or not
```

**Procedure main()**

```
  /* This the function which is invoked on the beginning of the program */
13: initiate();
14: loadFromBagOfWords();
15: readAllFiles(directoryPath);
16: printSummary();
```

```
Procedure initiate()
17: spamCount := 0;              → Initializing number of spam file count during testing to 0
18: hamCount := 0;               → Initializing number of ham file count during testing to 0
19: threshold := log(0.7/0.3)    → 0.7 for spam 0.3 for ham. Converted to log domain

Procedure loadFromBagOfWords()
/* Function loads the below variables from the file 'bagOfWords.txt'*/
20. open('bagOfWords.txt','readMode');  → read from bag of words
21: from bagOfWords do          → load all the variables from bag of words
22:     load(spamList, hamList, spamTrainingFilesCount, hamTrainingFilesCount,
             spamWordsCount, hamWordsCount, spamSum, hamSum);

Procedure readAllFiles(directoryPath)
/* This is function reads all files in the given directory and call spam detector function*/
23:   var: fo: FileObject;                    → file object for file related operations
24:     mail: String;                         → string to store the contents in a mail
25: for all the files in the directoryPath do  → reads all the files in the directory
26:     fo := open(file,'readMode');          → open the file for reading
27:     mail := fo.read();                    → read the entire contents in the mail
28:     fo.close();                           → close the file object
29:     isSpamOrNot(mail);                    → call the function isSpamOrNot to detect.

Procedure isSpamOrNot(mail)
/* Function is to detect whether the give file is a spam or not.
   It used Naïve Bayes classifier method for classifying a mail as spam or as ham
   To avoid underflow, all probability calculations are done in Log domain */
30: var: logSpamProb: float;   → probability of each word being spam
31:      logHamProb: float;    → probability of each word being ham
32:      spamGivenFile: float; → probability that given file is spam
33:      hamGivenFile:float;   → probability that given file is ham
34:      trainerTotalMails: int → total number of mails in the training set
35:      logSpamProb := 0;     → initializing a probability of a mail being spam to zero
36:      logHamProb := 0;      → initializing a probability of a mail being ham to zero
37:      totalTrainingMails = spamTrainingFilesCount + hamTrainingFilesCount;
38: for word in mail do                       → for all words in the mail content
39:     if word in spamList then
40:             logSpamProb := logSpamProb + log(spamList[word] + 1) \
                                - log(spamSum + spamWordsCount)
41:     else
42:             logSpamProb := logSpamProb - log(spamSum + spamWordsCount)
```

```
        /* Check the word is in ham list and calculate the corresponding probability */
43:     if word in hamList then
44:             logHamProb := logHamProb + log(hamList[word] + 1) \
                                - log(hamSum + hamWordsCount)
45:     else
                logHamProb := logHamProb - log(hamSum + hamWordsCount);

46:     spamGivenFile := spamProbs + log(spamFilesCount / totalTrainingMails);
47:     hamGivenFile := hamProbs + log(hamFilesCount / totalTrainingMails);

48:     if ((spamGivenFile - hamGivenFile) > threshold) then
49:             spamCount = spamCount + 1;
50:     else
51:             hamCount = hamCount + 1;
```

**Procedure printSummary()**

```
/* The function is to print the summary */
52: print 'Number of mails classified as Spam:', spamCount
53: print 'Number of mails classified as Ham:', hamCount
```

## 3. Evaluation

The below table shows the confusion matrix for a rule-based detector. The performance of the detector is tested against a mail set containing 10,000 Spam mails and 10,000 Non-Spam mails. Below table show the results obtained during the evaluation.

*Table 1: Confusion Matrix for Rule-Based detector*

| | **Actual Spam** | **Actual Non-spam** |
|---|---|---|
| **Classified as Spam** | 7228 | 2772 |
| **Classified as Non-spam** | 2513 | 7487 |

The below table shows the confusion matrix for a Naïve Bayes based spam detector. The program is trained over a set consisting of 2000 spam mails and 2000 non-spam mails. The performance of the detector is tested against a test mail set containing 2000 Spam mails and 2000 Non-Spam mails. Below (three) tables show the results obtained during the evaluation for the threshold 0.7, 0.8 and 0.9 respectively.

*Table 2: Confusion Matrix for Naïve Bayes Detector -Threshold 0.7*

| | **Actual Spam** | **Actual Non-spam** |
|---|---|---|
| **Classified as Spam** | 1852 | 139 |
| **Classified as Non-spam** | 148 | 1861 |

*Table 3: Confusion Matrix for Naïve Bayes Detector -Threshold 0.8*

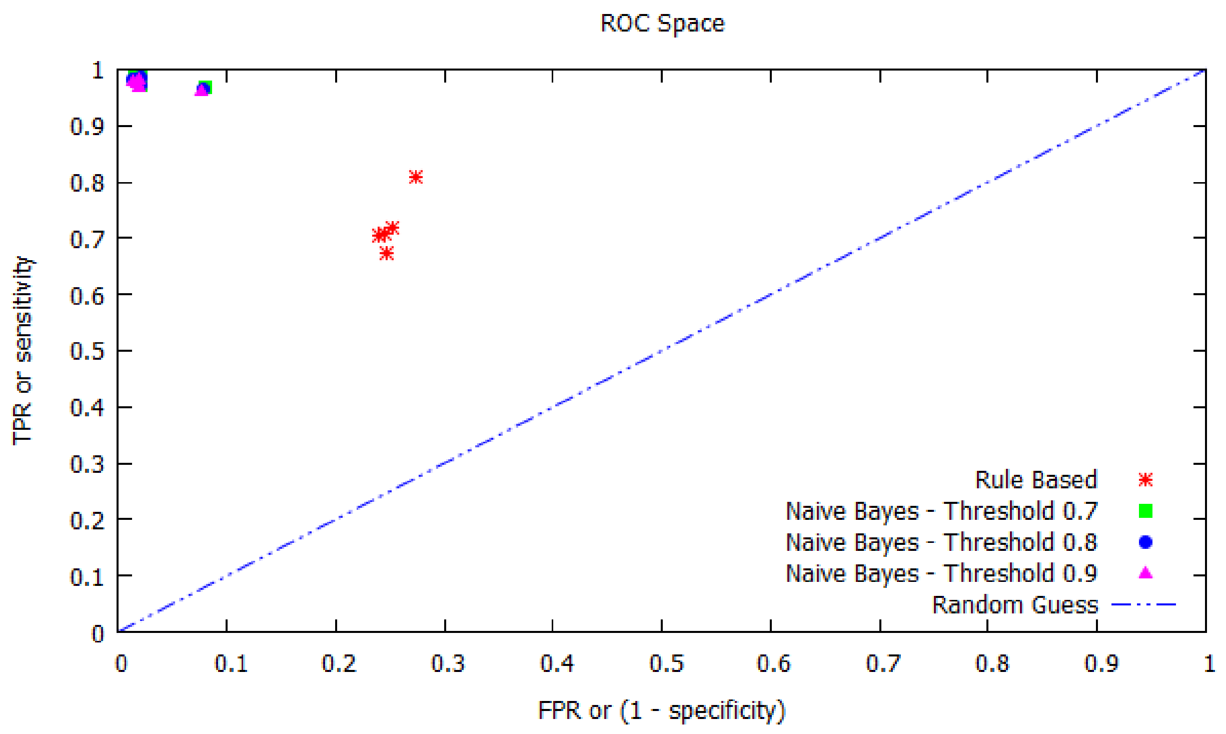
| | **Actual Spam** | **Actual Non-spam** |
|---|---|---|
| **Classified as Spam** | 1848 | 137 |
| **Classified as Non-spam** | 152 | 1863 |

*Table 4: Confusion Matrix for Naïve Bayes Detector -Threshold 0.9*

| | **Actual Spam** | **Actual Non-spam** |
|---|---|---|
| **Classified as Spam** | 1839 | 136 |
| **Classified as Non-spam** | 161 | 1864 |

From the obtained values, True Positive Ratio (TPR) and False Positive Ratio (FPR) are derived and the corresponding values are plotted on ROC space. The below graph shows the values of TPR and FPR obtained from Rule-Based detector and Naïve Bayes detector.

*Figure 1: ROC Curve*



From the graph it is evident that the TPR for rule-based detector lies around 0.7, whereas the TPR obtained from Naïve Bayes based detector is around 0.95. Similarly, the FPR for rule-based is around 0.25, whereas for the Naïve Bayes based detector the value of FPR is around 0.0.7.

From the results obtained it can be concluded that Naïve Bayes spam detector perform better in detecting spam mails and the accuracy of the Naïve Bayes spam detector is also high [6].

## References

[1] D. a. C.Manning, "Text Classification (Week 3), in Stanford University Coursera Course Natural Language Processing," [Online]. Available: https://www.coursera.org/course/nlp.

[2] V. Lavrenko, "Naive Bayes: Spam Detection - Simple Example," [Online]. Available: https://www.youtube.com/watch?v=8aZNAmWKGfs.

[3] T. Fawcett, "ROC Graphs: Notes and Practical Considerations for Researchers, Technical Report," HP Labs, 2003. [Online]. Available: http://home.comcast.net/~tom.fawcett/public_html/papers/ROC101.pdf.

[4] "LearnPython.org Interactive Python Tutorial," [Online]. Available: http://www.learnpython.org/.

[5] "Python Tutorial - Tutorials Point," [Online]. Available: http://www.tutorialspoint.com/python/.

[6] "Comparision of various spam detection techniques," [Online]. Available: http://www.paulgraham.com/stopspam.html.