Birla Institute of Technology & Science , Pilani Campus

MACHINE LEARNING ASSIGNMENT

SPAM FILTER

GROUP MEMBERS:

1. Priyank Gupta 2013A7PS060P
2. Saurabh Anand 2013A7PS139P
3. Shivank Garg 2013A7PS133P

PROBLEM STATEMENT

Spam mail filter --Our motive is to filter out mails which are spam, based on the prior readings of training set and thus develop a algorithm with maximum accuracy of judging the mail whether it is ham or spam. The accuracy should depend on the error rate and false positives with both of them being as less as possible.

ASSUMPTIONS

- The emails are given to the program in .txt format.
- The probability of a mail being ham or spam is equal.
- Rare word count is 5 and rare word weight is 3.
- We can successfully ignore words that have almost equal probability of being ham or spam.(ie, spamicity between 0.2 and 0.8).

DATA AND SOURCE

The dataset that we have used to train and test our filter is the famous Enron dataset. This is available at :-

https://www.cs.cmu.edu/~./**enron**/

We used Enron Corpus 1 for training and Enron Corpus 2 and Enron Corpus 3 for testing.

Training Dataset-Enron Corpus 1
Ham

- Owner: farmer-d
- Total number: 3672 emails

- Date of first email: 1999-12-10
- Date of last email: 2002-01-11
- Similars deletion: No
- Encoding: No

Spam

- Owner: GP
- Total number: 1500 emails
- Date of first email: 2003-12-18
- Date of last email: 2005-09-06
- Similars deletion: No
- Encoding: No

Spam : Ham rate = 1:3
Total number of emails (Ham + spam): 5172

Testing dataset-
#Enron corpus 2
Ham

- Owner: beck
- Total number: 1500 emails
- Date of first email: 1998-10-05
- Date of last email: 2002-12-18
- Similars deletion: No
- Encoding: No

Spam

- Owner: SA_AND_HP
- Total number: 3675 emails
- Date of first email: 2000-12-18
- Date of last email: 2002-02-18

- Similars deletion: No
- Encoding: No

Spam : Ham rate = 3 : 1
Total number of emails (Ham + spam): 5175

#Enron corpus 3
Ham

- Owner:lokay-m
- Total number: 1500 emails
- Date of first email: 1999-06-06
- Date of last email: 2002-03-11
- Similars deletion: No
- Encoding: No

Spam

- Owner: BG
- Total number: 4500 emails
- Date of first email: 2004-12-18
- Date of last email: 2005-09-06
- Similars deletion: No
- Encoding: No

Spam : Ham rate = 1:3
Total number of emails (Ham + spam): 6000

DETAILS OF PRE-PROCESSING DONE

The Enron dataset that we used already had pre-processed emails in the form of text files. So there was no requirement of pre-processing in our case. But during training we ignored the mails that could not be decoded into UTF-8 format.

COMMON TERMS USED

Ham – Email which is genuine.
Spam – Email which is spam.
Epsilon – Small number used to remove cases with 0 or 1 spamicity in calculations.
Rare word count – Minimum number of occurences for a word to be classified as rare.
Rare word weight – Weight assigned to the occurrence of rare word.
Spamicity – The probability that a given word occurs in a spam email.
Stop-words – List of common words used in the English language. We have ignored these words in our calculations.

FEATURES USED

We have used the following features :-

- Words (tokens which are words and not in stop-words. Eg:- food ,sport ,court)
- Numbers (tokens which are numbers. Eg:- 1234,9774321456)
- Punctuations (tokens composed only of punctuations. Eg:- //,.* , /\\//)
- Links (tokens which are links. Eg:- https://google.com/)
- AllCaps (tokens composed of capital words. Eg:- SLEEP,WORK)

Other features which could be incorporated in the future :-

- Text size
- Attachments used
- Pictures
- Text colour

TRAINING APPROACH USED

First, we extracted all the emails. Then we created two dictionaries spam_dict and ham_dict which contained all the features occurring in spam and ham emails along with their frequencies respectively.
Then we assigned each feature with a spamicity value and stored it another dictionary 'spamicity'.

CLASSIFICATION AND ALGORITHM USED

We have used Naive Bayes Classifier to test the spam mail.
Computing the probability that a message containing a given feature is spam:

The formula used by our program to determine that is derived from Bayes' theorem

$$\Pr(S|W) = \frac{\Pr(W|S) \cdot \Pr(S)}{\Pr(W|S) \cdot \Pr(S) + \Pr(W|H) \cdot \Pr(H)}$$

where:

- Pr(S|W) is the probability that a message is a spam, knowing that the given word is in it;
- Pr(S) is the overall probability that any given message is spam;
- Pr(W|S) is the probability that the given feature appears in spam messages;

- Pr(H) is the overall probability that any given message is not spam (is "ham");
- Pr(W|H) is the probability that the given feature appears in ham messages.

Most bayesian spam detection software makes the assumption that there is no a priori reason for any incoming message to be spam rather than ham, and considers both cases to have equal probabilities of 50%: Pr(S)=0.5;Pr(H)=0.5
The filters that use this hypothesis are said to be "not biased", meaning that they have no prejudice regarding the incoming email. This assumption permits simplifying the general formula to:

$$\Pr(S|W) = \frac{\Pr(W|S)}{\Pr(W|S) + \Pr(W|H)}$$

Combining individual probabilities

Most bayesian spam filtering algorithms are based on formulas that are strictly valid (from a probabilistic standpoint) only if the words present in the message are independent events. This condition is not generally satisfied (for example, in natural languages like English the probability of finding an adjective is affected by the probability of having a noun), but it is a useful idealization, especially since the statistical correlations between individual words are usually not known. On this basis, one can derive the following formula from Bayes' theorem:

$$p = \frac{p_1 p_2 \cdots p_N}{p_1 p_2 \cdots p_N + (1 - p_1)(1 - p_2) \cdots (1 - p_N)}$$

where:

- p is the probability that the suspect message is spam;
- p_1 is the probability $p(S|W_1)$ that it is a spam knowing it contains a first word (for example "replica");
- p_2 is the probability $p(S|W_2)$ that it is a spam knowing it contains a second word (for example "watches");

- etc...
- p_N is the probability $p(S|W_N)$ that it is a spam knowing it contains an Nth word (for example "home").

Dealing with rare features

In the case a feature has never been met during the learning phase, both the numerator and the denominator are equal to zero, both in the general formula and in the spamicity formula. Our program discards such features for which there is no information available.

More generally, the features that were encountered only a few times during the learning phase cause a problem, because it would be an error to trust blindly the information they provide. A simple solution is to simply avoid taking such unreliable features into account as well.

Applying again Bayes' theorem, and assuming the classification between spam and ham of the emails containing a given feature is a random variable with beta distribution, we decided to use a corrected probability when its frequency of occurrence <= Rare word count:

$$\Pr'(S|W) = \frac{s \cdot \Pr(S) + n \cdot \Pr(S|W)}{s + n}$$

where:

- $\Pr'(S|W)$ is the corrected probability for the message to be spam, knowing that it contains a given feature;
- s is the strength we give to background information about incoming spam ;
- Pr(s)is the probability of any incoming message to be spam ;

- n is the number of occurrences of this feature during the learning phase ;
- Pr(S|W) is the spamicity of this feature.

Calculation Assumptions

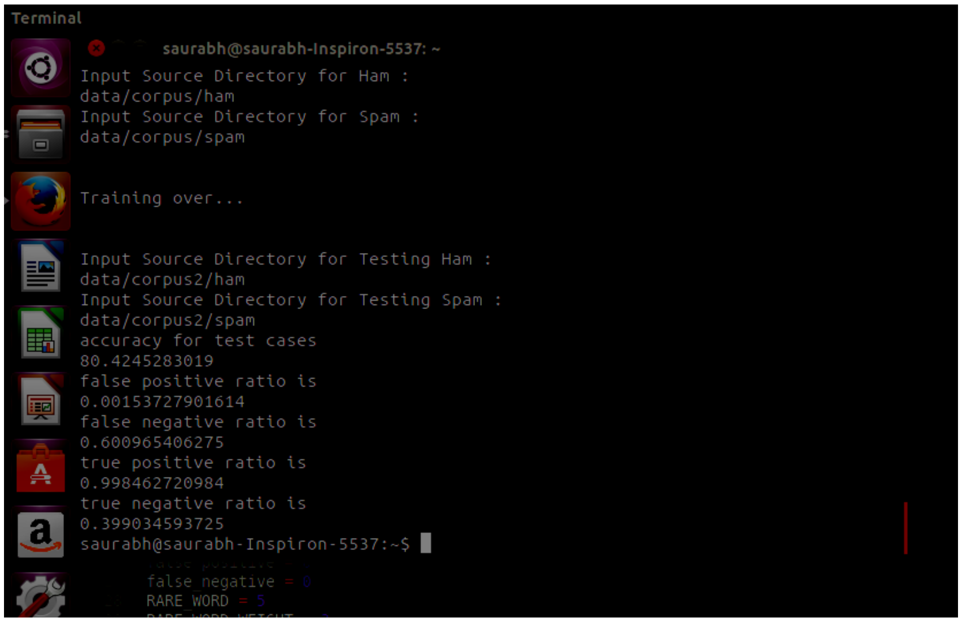
S(Rare word weight)=3 whenever n is less than equal to 5 (Rare word count).

Now when the calculated probability using the formula of combining individual probability is greater than or equal to 0.8, algorithm considers it to be spam.

RESULT

We tested our algorithm with the Corpus2 and Corpus3 datasets that we had set aside for testing. We used different values for Epsilon and spam threshold. The results are as follows :-

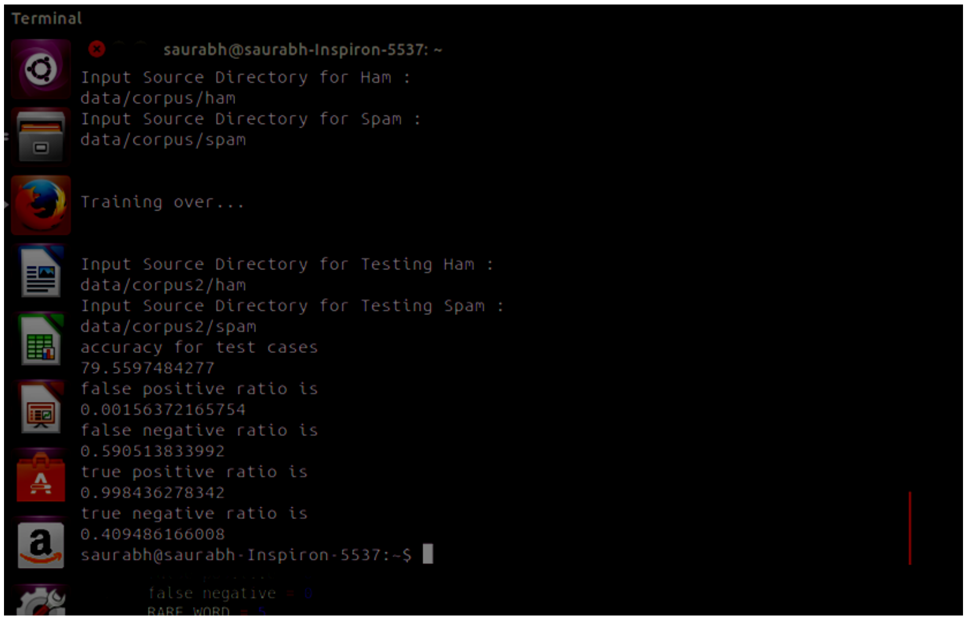
	Epsilon	Spam Threshold	Accuracy	False positive ratio
Corpus2	0.0001	80	79.559	0.00156
	0.00001	80	80.4245	0.001537
	0.0001	85	79.088	0.00118
Corpus3	0.0001	80	88.686	0.00274
	0.00001	80	88.929	0.003278
	0.0001	85	88.2189	0.00221



```
saurabh@saurabh-Inspiron-5537: ~
Input Source Directory for Ham :
data/corpus/ham
Input Source Directory for Spam :
data/corpus/spam


Training over...


Input Source Directory for Testing Ham :
data/corpus2/ham
Input Source Directory for Testing Spam :
data/corpus2/spam
accuracy for test cases
80.4245283019
false positive ratio is
0.00153727901614
false negative ratio is
0.600965406275
true positive ratio is
0.998462720984
true negative ratio is
0.399034593725
saurabh@saurabh-Inspiron-5537:~$ 
```



```
saurabh@saurabh-Inspiron-5537: ~
Input Source Directory for Ham :
data/corpus/ham
Input Source Directory for Spam :
data/corpus/spam


Training over...


Input Source Directory for Testing Ham :
data/corpus2/ham
Input Source Directory for Testing Spam :
data/corpus2/spam
accuracy for test cases
79.5597484277
false positive ratio is
0.00156372165754
false negative ratio is
0.590513833992
true positive ratio is
0.998436278342
true negative ratio is
0.409486166008
saurabh@saurabh-Inspiron-5537:~$ 
```

Analysis

The result shows that the Naïve Bayesian algorithm is a very good method for designing spam filters. We were able to get a high accuracy (80% - 90 %)and a quite low amount of false positives (0.001-0.003). Also to reduce the number of false positives, we tried experimenting with the threshold value and epsilon value, ie. increasing it which led to an increase in the error rate as well.

All in all, Naïve Bayesian proved to be most efficient at spam threshold value= 80 and Epsilon value 0.00001.