Naive Bayesian Spam Filtering

Team members:
Sai Santhosh Vaidyam Anandan (SBU ID: 109819365)
Sowmiya Narayan Srinath (SBU ID: 109747156)

Assumptions:
The test and training datasets (files named test and train) are present in the same directory as the code.

Logic and Execution:
We have two spam filtering (spam_filter.py and detect.py) programs to demonstrate how choosing of features affect efficiency.

1. Occurrence of word as feature:
Here the occurrence of each word is considered a feature and probabilities of spam and ham given a word is calculated based on whether the word occurs in train data set under spam or ham categories.
(i.e) $P(yes).P(word1|yes).P(word2|yes)$ ...

For the given test data set this gave 91% accuracy but for some broader inputs this will perform relatively poorly. But in this method we are not considering the number of times a word occurs. Some common words may occur more times in spam and not in ham.
So we choose number of occurrences as the better feature for more accurate detection.

2. Number of occurrences as the feature
Initially we pick words occurring more than a certain threshold (*min_word_count*) as features.
For each data in the test set, we pick all words which are chosen as features. Then we check the probability of the word appearing more than the input in spam and ham.
(i.e) $P(yes).P(word1>val1|yes).P(word2?val2|yes)$...
As min_word_count is reduced, it means more words are picked as features and thus gives a far better estimate. Even with min_count as high as 50 , we saw 90% accuracy with given data set. As it reduces, it is clear that this model performs much better against the previous one. The downside is the increase in runtime from the order of few seconds to more than a minute now.

Output Trace:

1. Naive filter
```
$ python spam_filter.py
Training data processed, total records is  9000
Please check results.txt file for inferred results
Predicted  91.3 % of results correctly.
Runtime  0.99050617218 s.

$ more results.txt
```

ham
ham
spam
spam
spam
spam
spam
spam
spam
spam
spam
spam
spam
spam
spam
spam
.
.
.

2. More accurate filter:

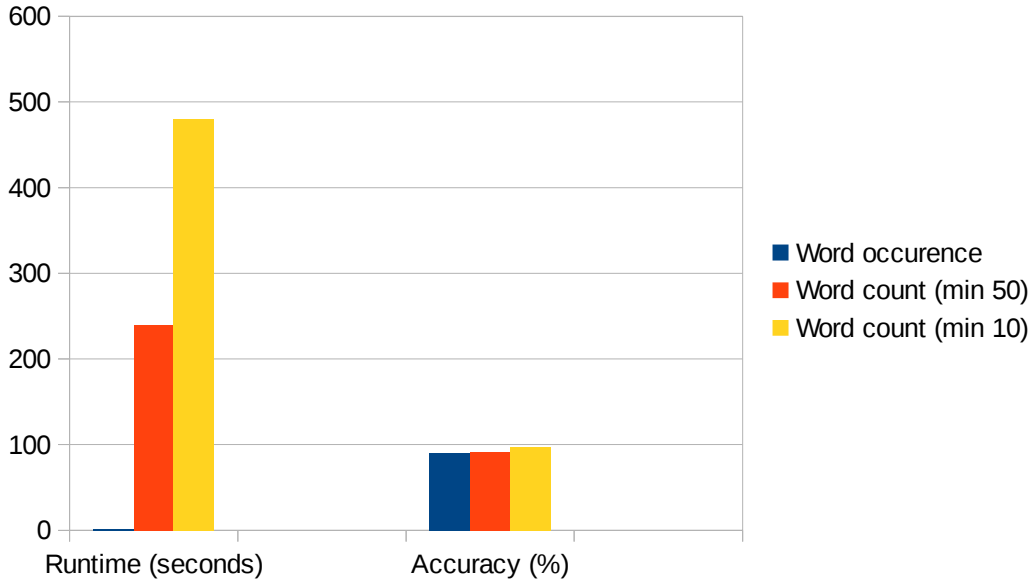
$ python learning.py
Number of features: 227
Building feature table...

$ python detect.py
Ham !
Ham !
Spam !
Spam !
Spam !
.
.
.

Results Analysis:

Method	Word occurrence as feature	Word count as feature (min = 50)	Word count as feature (min= 25)	Word count as feature (min=10)
Runtime	1s	240s	280s	480s
Accuracy	**91.3%**	**90%**	**92.3%**	**97%**

The key parameters in comparing the two filtering mechanisms are the **runtime** and **accuracy of detection**. Below graph compares the filtering strategies on these parameters for the given test data.



Files Submitted:

spam_filter.py – Naive bayes spam detector with word occurrence as features
learning.py and detect.py – Two phases for the spam detector based on word count as the features
train – learning data set
test – Input dataset to check accuracy of the code
results.txt – generated output file by spam_filter
Report.pdf – this file.