Sonia Nigam

**Problems 1 and 2**

Implemented in “spamfilter.py”. This can be run with the following command:

python spamfilter.py spam/ easy\_ham/ test

**Problem 3**

**Problem 4**

**A)** I used 500 random emails from the downloaded directory *20030228\_spam.tar.bz2* and 500 random ham files from the downloaded directory *20021010\_easy\_ham.tar.bz2*. I chose data from these two files because they were the recommended directories, and in turn, I hoped that they would best emulate real information that the system is intended to be built for.

I split the data into 50 folds, such that the training set had 49/50th of the spam data and 49/50th of the ham data, while the testing set had 1/50th of the spam data and 1/50th of the ham data. In turn, the system was not trained with the same data it was tested with. I thought it was important to truly test how well my system can be trained, and in turn it was necessary to separate the training and testing data. If there was overlap, my system we would be inherently biased towards the testing data, which would provide skewed results.

I tested the system with 50% spam files and 50% ham files. Looking at the spam documents, I would say that the spam files are relatively representative of real-world spam. The content of these files focus on a given scam, disguised as charity or a non-intrusive task. Moreover, they often request personal info, money, or link directly to spam sites. These attributes are all very realistic of real-world spam. One key difference, however, is that oftentimes spam mail comes from addresses disguised as one in your contact book. While the actual email address may be slightly different than the real email address, it leverages a close friend or family member’s name to come off as safe mail. Because this system does not emulate the idea of stored contacts, it cannot fully account for this common spam tactic.

**B)** In order to test my spam filter against prior probability, I used cross validation. I chose to do 50-fold cross validation with the dataset described above, 1000 total files where 50% are spam and 50% are ham. With this structure, I split the spam section and ham section into 50ths. For each iteration, I combined 1/50th of the spam files and 1/50th of the ham files to create the testing set. The other 49/50ths of both file sets were combined to create the training set. I did this 50 times, such that each file was in the testing set exactly one time. I did this to get a better idea of the effectiveness of my system, as I had 50 samples’ data points to compare. If I only tested my system once, it would be hard to prove that the returned error measures are representative of the system rather than just the given data. Meaning, I wanted to ensure that my experiment was run on several combinations of data samples, such that the results were evidently not biased towards the training and testing data set choices.

I chose to use f-measure to assess my error and general accuracy of the method. Meaning, a higher f-measure represents a higher combination of precision and recall. I thought it was important to weigh false positives, false negatives, true negatives, and true positives accordingly. Instead of just noting misclassification, this process accounts for what type of misclassification (false positive, false negative, etc.). Moreover, by averaging the accuracy of spam classification (f-measure) and ham classification (f-measure), this holistic sample measure accounts for the two-part system. Meaning, if one system classifies spam very well, but ham very poorly, it will be captured within this measure system, and one part cannot skew the overarching results.

The following formulas are what I used.

*Spam Precision = True Spam / (True Spam + False Spam)*

*Spam Recall = True Spam / (True Spam + False Ham)*

*Spam F-Measure = 2\* ((Spam Precision \* Spam Recall) / (Spam Precision + Spam Recall))*

*Ham Precision = True Ham / (True Ham + False Ham)*

*Ham Recall = True Ham / (True Ham + False Spam)*

*Ham F-Measure = 2\* ((Ham Precision \* Ham Recall) / (Ham Precision + Ham Recall))*

**Sample F-Measure = (Spam F-Measure + Ham F-Measure) /2**

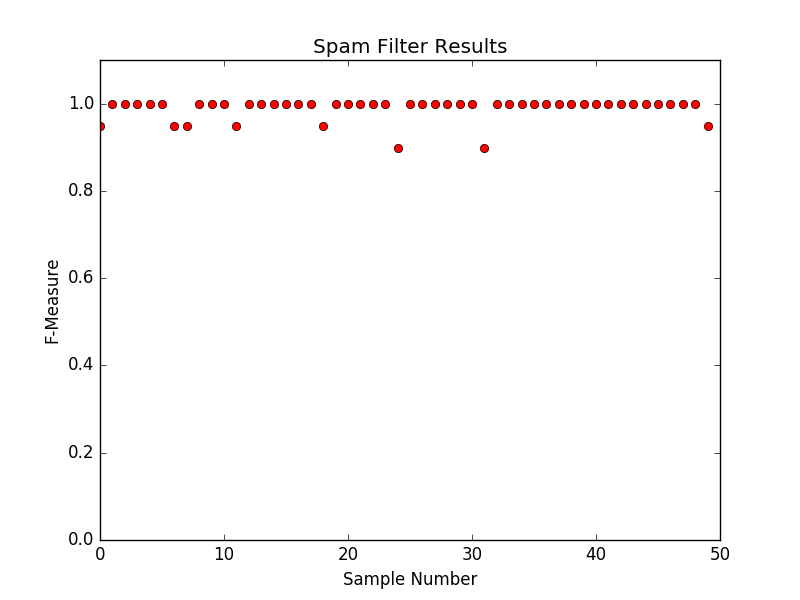
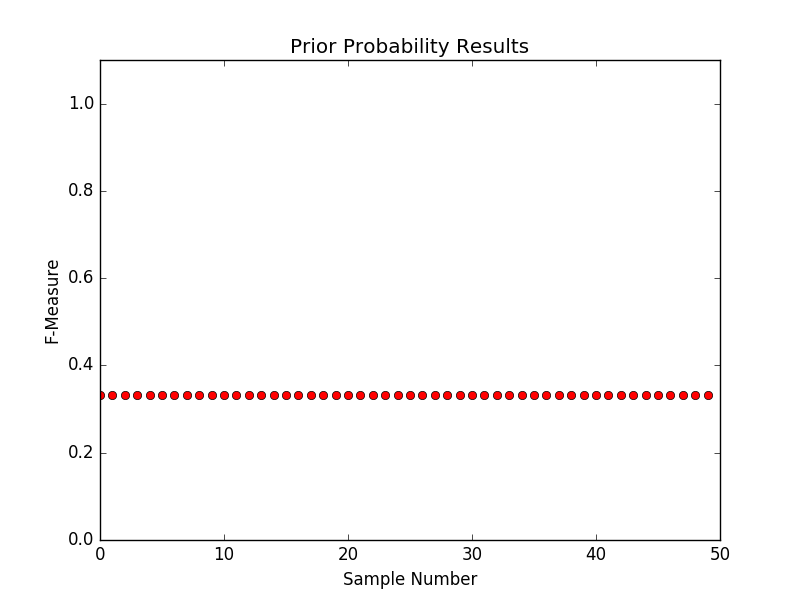
**C)**

Based on my experiment, the spam filter does a better classification job than prior probability.

The mean f-measure for spam filter is: 0.989944558366

The mean f-measure for prior probability is: 0.333333333333

Below is a depiction of the f-measures across all 50 samples.



Below are the exact data distributions: each f-measure corresponds to the sample index.

Spam Filter:

[0.949874686716792, 1.0, 1.0, 1.0, 1.0, 1.0, 0.949874686716792, 0.949874686716792, 1.0, 1.0, 1.0, 0.949874686716792, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.949874686716792, 1.0, 1.0, 1.0, 1.0, 1.0, 0.898989898989899, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.898989898989899, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.949874686716792]

Prior Probability:

[0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333, 0.3333333333333333]