Name: Leesha Maliakal

1. **Design choices:**
    a. Describe how you decided to build your dictionary. How did you decide to deal with non-alphanumeric characters? How about plurals? What other issues did you build your system to handle? Is every word from the training corpus in your dictionary or did you remove some words or limit the dictionary size? Why?
        i. I built my dictionary with the following steps:
            1. For each document in Spam and Ham
                a. Convert document contents to a string
                b. Used BeautifulSoup to remove HTML and retrieve text from the emails
                c. Tokenized the BeautifulSoup text using NLTK
                d. removed duplicates from the token representation of the doc
            2. Added the tokenized document to dictionary using this algorithm:
            function name - `addToDictionary(tokens, dictionary, isSpam, totalSpamCount, totalHamCount)`

```
#if spam document
      #for each token
            #if in dictionary
                  #increment spam frequency of token
                  #update token's p(spam)
            #else
                  #add token to dictionary (append [token, (spam freq/spam count), (ham freq/ham count)] )
#if ham document
      #for each token
            #if in dictionary
                  #increment ham frequency of token
                  #update token's p(ham)
            #else
                  #add token to dictionary (append [token, (spam freq/spam count), (ham freq/ham count)] )
```

        ii. I used the features available in BeautifulSoup and NLTK to extract “words” from the email and toss out any characters related to HTML. I also encoded in unicode to handle non-alphanumeric characters like Chinese characters that are often found in spam. It cannot differentiate from plurals. Every word from the training corpus was used, and I used the entirety of the two ham/spam sample sets that were suggested in the HW document. I decided to use all training data because a larger training set will lead to a “smarter” classifier, because it has trained on more examples.

b. Describe how you decided to build your spam sorter. How did you calculate the probabilities? Did you end up using equation (6) or (7)? If you used (7), give a proof it would return same results as (6), if underflow were not an issue.
    i. I did not implement the spam sorter, but here is how I designed it:
        1. Algorithm:
            a. for each document in ham and spam
                i. and for each token in that document
                    1. pSpam *= P(token|spam)
                    2. pHam *= P(token|ham)
                ii. for each token not in the document but in the dictionary
                    1. pSpam *= [1-P(token|spam)]
                    2. pham *= [1-P(token|ham)]
                iii. spamScore = ceiling(log(pSpam))
                iv. hamScore = ceiling(log(pHam))
                v. if spamScore >= hamScore
                    1. move document to spam directory
                vi. if hamScore >= spamScore
                    1. move document to ham directory
        2. I should also be careful about the zero probability issue:
        A drawback of relative frequency estimates—equation (8) for the multinomial model—is that zero counts result in estimates of zero probability. This is a bad thing because the Naive Bayes equation for the likelihood (7) involves taking a product of probabilities: if any one of the terms of the product is zero, then the whole product is zero. This means that the probability of the document belonging to the class in question is zero—which means it is impossible. Just because a word does not occur in a document class in the training data does not mean that it cannot occur in any document of that class. I circumvented this problem by using a pseudo count when initializing the spam counter and the ham counter. I set them to 0.1 so as not to dramatically affect the actual probability ratios.

2. **Datasets:** In your experiments, what data set did you use to build your dictionary? What data set did you use to test the resulting spam filter? Were they the same? Why did you choose the data sets you chose? How many files are in your data? What is the percentage of spam in the data? Is your data representative of the kinds of spam you might see in the real world? Why or why not?
    a. I used the data sets that were suggested by the homework: *20030228_spam.tar.bz2, 20021010_easy_ham.tar.bz2*. For the spam filter, I would have used a different set of data that was not already used in training, like 20021010_spam.tar.bz2 and 20030228_easy_ham_2.tar.bz2. I chose these sets because they had plenty of examples, and were suggested by the homework document. There are 2,551 ham documents and 502 spam documents. The data is 19.68% spam. The data was

selected from the spam corpus supplied by SpamAssassin, which is a reputable program for finding spam <http://spamassassin.apache.org/publiccorpus/>. It uses real world spam/ham examples, making it a highly reputable spam corpus.

3. **Your Testing Methodology:** Explain how you tested the spam filter to see if it worked better than just using the prior probability of spam learned from your training set. Did you do cross validation? If so, how many folds? If not, why not? What is your error measure?
    a.
4. **Experimental Results:** Does your spam filter do better than just using the prior probability of spam in your data set? Give evidence to back up your assertion. This may include well-labeled graphs, tables and statistical tests. The idea is to convince a reader that you really have established that your system works (or doesn't).
    a.
5. **Future Work:** How might you try to redesign the system to improve performance in the future? Would you take into account higher-level features (e.g. sentence structure, part-of-speech tagging) when categorizing as spam? Would you exclude portions of the mail as meaningless noise? Why? Be specific. Saying only "I will use part-of-speech information" is not a meaningful response. I want a description of what improvements you would make, why you think they would help and how you would implement that.
    a. For future work, I would go beyond the analysis of word to word and think more about the actual grammar, sentence structure, context, etc. It would also be smart to go through the header information in the future and search for key components that are usually associated with spam, like unusual IP addresses, domains, and email addresses. I could also attempt to safely examine any links in the documents and evaluate whether they are safe or not by scraping the website contents. For much of this analysis, I would be able to use existing tools for natural language processing, like NLTK, and web scraping tools like Kimono. For the exisiting word to word approach, it would be wise to eliminate words from my dictionary that only appear once in either spam or ham, where pSpam or pHam = 0. Some examples include

        ```
        Claims 0.00407996736026 0
        Clarity 0.00225682690138 0
        Clark 0 0.0235026062711
        ```

        This would significantly shrink our dictionary size and it would be much easier and faster to use for the queries in our classifier, thus speeding up the classification process. It would also be smart to not differentiate between works with different capitalizations, like Claim vs claim. That way, I am accurately assessing the frequency of the word, and not the frequencies of different variations of the word. I could do this by applying a function that strips words of their capitalization and sets all characters to lowercase, for example.