Philip Blumin
NLP Spring 2021
Text Categorization Project
Carl Sable

The code for this project was written in python. Both the training and testing takes place in one script. As the script runs it automatically downloads nltk as I use this library for tokenizing. Upon starting up the program the user is prompted to enter a file that contains the labeled training documents, a file that contains unlabeled test documents, and an output file. If an invalid file is entered the program will end and a python error message will be returned.

The machine learning method I used for my program was Naïve Bayes. I followed the method and pseudo code outlined in the textbook step-by-step, with a minor change when calculating the log likelihood. Aside from the basic training and testing function I have a function that fills out input dictionaries with counts just like the tables in the textbook. In this function I add the smoothing method described in the textbook, where everywhere that there is a count of 0 is changed to 1. When calculating for probabilities I followed the textbooks **equation 4.14** and pseudo code in **figure 4.2.** When using the exact equation in the textbook, the results I got at first were not too good, between 60% and 80% for the 3 provided corpuses. The first thing I decided to experiment with was to change the alpha value (the +1 in equation 4.14).

I got pretty good improvements when I changed the alpha value to 0.1. I got results between 70% and 85% for the 3 corpuses. I still wanted to see if I could improve my results some more so I experimented with a few new parameters and features. After a suggestion from a peer, I found that multiplying the log by a weight made a huge difference. The weight that I multiplied it by was the count of the number of words in testdoc (a dictionary of words and their respective count in the test corpus). This ended up improving my results significantly as I got 90-92% for corpuses 1 and 3 and 81-83% for corpus 2. I was fairly happy with these results.

I evaluated my system's performance for the second and third data sets by creating a short python script that split the data using sklearn's train_test_split function. I chose a 80-20 split both corpuses.