John Geddes

CS 460G

Homework 5 Report

April 9, 2018

**Implementation Decisions**

I decided to use Python 2.7 in this project. There was no particular reason other than it was already installed on my computer.

I decided to use dictionaries to store the word counts for each word because that was the easiest built-in way to do so. When I imported training and testing data for the unigram tokens, I found a way to remove all punctuation from the data sets. I noticed that all apostrophes and contractions were split with a space between the word they should be on. I also noticed that the less-than symbol began all end-of-line markers. With this knowledge, I only had to look at the first character in any token and check if it was one of the following: a period, a comma, a question mark, an exclamation point, an apostrophe, or a less-than symbol. Removing any token with those at the beginning left me with only actual words.

In the bigram tokening, I also used a dictionary to store the word counts. This time, as I looped through a sentence, I stored one word at a time into a buffer and combined all consecutive word pairs. These combined pairs were of the form *word1 + “ “ + word2*, and this was used as the keys to the bigram dictionary. So, every two words were combined with a space in the middle.

When calculating the log probabilities of a word occurring in a given class, I decided to make the pseudo-count variable equal to 0.0001 because that seemed to be the best out of a few decimals I tested with varying numbers of zeroes. I imported the “math” library to be able to use a log function. I had to check that a word in the test set existed in the training set dictionary because the dictionary would throw an error that the key did not exist. When this happened, I removed the call to the dictionary and just included the pseudo-count in the numerator.

**Testing Results**

Unigram data accuracy:

Hamlet accuracy: 0.5

Juliet accuracy: 0.494949494949

Macbeth accuracy: 0.356643356643

Romeo accuracy: 0.376923076923

Total accuracy: 0.432128982129

Bigram data accuracy:

Hamlet accuracy: 0.523489932886

Juliet accuracy: 0.444444444444

Macbeth accuracy: 0.335664335664

Romeo accuracy: 0.323076923077

Total accuracy: 0.406668909018

On average, the unigram testing was more accurate than the bigram testing. I think this is the case because the frequencies for every word pair in the training set was not very high; they never went above 26. Compared to the unigram tokening, the frequencies reached as high as 376, but that was the word “the,” which is already very common. Therefore, if the test sets were similar, it was very unlikely that word pairs would be seen again during testing as they occurred in training. So, it was hard to predict that pair of words when they are so uncommon.

I also think that the reason that each accuracy is between 30% and 50% is because all of the characters were written in such a similar style that a lot of sentence structures are exactly the same between characters. But the accuracy of this predictor is greater than 25%, which means that it is more accurate than randomly guessing.