William Chen

Professor Sable

ECE-467-1 Natural Language Processing

17 March 2020

Project #1 Writeup

For this assignment, I implemented a Naive Bayes classifier written in Python, and the only library that it needs to have installed is NLTK. For me, I had to download a fairly sizable NLTK before I could use the library, but it should already be installed on your system.

The system tokenizes training and test files by using the functions that are available in the NLTK library. I did some processing and cleaning up on the document both before and after tokenizing it, and generally saw small improvements to the accuracy from the methods I used. First, I used regex to remove numbers out of the documents, since numbers tend to be rather specific to their documents and don’t have any meaning for classification purposes. I then tokenize the entire document with the standard NLTK tokenizer, and then set all of the characters in each token to lowercase to remove case sensitivity. All the punctuation was removed and stop words were removed, both of which resulted in small gains in accuracy around 1% or so. I also implemented the ability to either use the Porter stemmer or a lemmatizer, but it seemed that the Porter stemmer performed slightly better. I had originally had an additional library expand contractions, but I noticed that it made no difference in performance, likely since the stop list would remove words like “not” that contractions would commonly expand to.

I didn’t use any weighting scheme for tokens, but if I were to I would probably assign a weight based on parts of speech. Certain parts of speech like verbs and prepositions would probably get a higher weight especially in corpus 2 where location is especially pertinent, while parts like pronouns would be assigned a lower weight because I would imagine names of people or locations tend to be rather unique to their specific documents.

The method of smoothing that was used in my model is Laplace smoothing, which the textbook notes is the simplest but still viable enough to be commonly used for Naive Bayes text classification. The value I started with was 1, but I decided to experiment with a few different numbers and found that the value that worked best for corpus 1 was 0.1. I repeated the process of tuning for the other two corpuses, and found that 0.1 was the best compromise all around.

Since corpus 2 and 3 didn’t have a test set provided, I randomly picked 25% of the data each time the program was run instead of splitting it permanently before in order to avoid overfitting for a specific tuning set. I would run a few tests for each corpus for each tuning to get a general idea of what the average accuracy was, and make tweaks to the Laplace value try to improve on that.