Jessica Brennan

Machine Learning

Homework 4 – Naïve Bayes

I started off in main by loading each text file into a numpy array, splitting by each new line (‘\n’). I then created a corpus for each file and iterated through each row of the respective numpy array to append to these corpuses. Afterwards, I combined the positive and negative corpuses while adding class labels to each review (pos for positive reviews and neg for negative reviews). I shuffled this array so that when I split the data up into train, development, and test sets, I would have both positive and negative reviews included. I printed the number of reviews to double check I got all 10,662 of them.

In order to use the words in the reviews as features in the classifier, I preprocessed and normalized the data in the preprocess function using NLTK’s tokenizers and lemmatizers. The tokenizer splits the text by white spaces and the lemmatizers liken different forms of the same word. The text was already completely lowercase; otherwise I would have added this in as well.

My get\_features function extracts the features that identify positive versus negative reviews. I created a file using the stop words you recommended and added in some punctuation. If a particular word in the review were in that list, it would be excluded. For words not caught by this stop word file, the function would calculate how frequently it occurred in the text using the counter method. This allows the classifier to notice that a particular word may appear in both the positive and negative reviews, but perhaps with different frequencies. Back in the main method, I extracted these features from each review while retaining the class label.

I used my train method to first split up the data into training, developing, and testing sets. First, I calculated the training set size by multiplying the number of total features by a passed percentage (e.g. 0.7 for 70%). I then calculated the development size by multiplying the number of total features by half of what was left of the percentage and added the value I received for the training set size. Using the training set size and development set size, I split up my data. I set train to features[:train\_size], dev to features[train\_size:dev\_size], and test to features[dev\_size:]. I printed the number of reviews per set to double check they were split correctly. After splitting these up, I trained the classifier on the test set. The classifier tries to choose the most probable class (positive or negative) based on what it learned about the features (the frequencies of words in each type of review).

Next, I evaluated my development set on the training set. I printed the accuracy for both training and development sets here. I ended up with 95.1% accuracy on the training set (not surprising) and 77.5% on the development set. I also printed the most informative features, as I was curious. The top negative words were flat, pretentious, and generic. The top positive words were engrossing, wonderful, and refreshing. Finally, I tested the classifier on my test set and received 78.25% accuracy. This time, the top negative words were generic, mediocre, and stupid and the top positive words were engrossing, warm, and provides.