Himanshu Gupta CS-510 HW2

{'negative': 26235, 'neutral': 34451, 'positive': 66514}

Answer for Reflection Question:

As we have trained our algorithm from the set of movie and product reviews, it is very likely that our algorithm would not accurately predict words with foreign names. I have tested my algorithm for my name: Himanshu Gupta which is very unlikely to show up in any of the reviews. In these types of cases, our result would only be dependent on the fraction of the respective positive or negative class. Whatever fraction would be greater, the result would be of that document class. In the case of my name, as from the above dictionary, the positive fraction would be greater than the negative fraction, the result for my name generated as the positive class.

Similarly, our training set unlikely to contain words not used in reviews. One more example is “I am currently pursuing computer science”. The result generated is negative. In this case, some part of the sentence has the greater conditional probability that there counter conditional probabilities in positive. This tilted the scale in favor of the negative class.

Similarly, I have tested out for a sentence containing the numbers: "60 degrees > 80 degrees". The result generated is positive as neither the positive nor the negative dictionary supposed to contain these types of numbers. Thus, our result would only be dependent on the fraction of the respective positive or negative class. As from the above dictionary, the positive fraction would be greater than the negative fraction, the result is positive.

We can not tabulate the results from the above examples to be accurate.

Bayes_template.py vs Bayes_template_best.py:

Bayes_template implements the Bayes algorithm. It took care of the underflow problem by adding the logarithms, also one smoothing has been added to take care of all the scenarios. For the words which don’t suppose to be in the training set, the result would be more accurate than the bayes_template_best. This is because of unigram modeling. The bigram would generate the conditional probability lesser than that generated by unigram program bayes_template.

Bayes_template_best implements a bigram feature on top of Bayes_template. This ensures more accurate results including better grammar check, better natural language sentences, even better accuracy for highly unlikeable words (not supposed to be in the training set). Through using bigrams, we are training the algorithm to learn meaningful elements. For example, the P(want to | ….) should be higher than P (to want |) because of grammatical rules.

For the same set of tests, The compilation time for bayes_template_best is greater than the bayes_template because of the added bigrams.

There are some factors that would affect our results like the number of positive/negative filenames. If the positive files are greater than the negative filenames, the results would be more positive as the number of negative files would be less. This is because of fraction in probability formula would contribute to the result more of the negative class.

The future improvements in the algorithm would be as following:

1) Data set could be better chosen which has more variety of words so that results could be more accurate
2) More level of smoothing could be implemented to train our algorithm to recognize more sensible sentences.
3) Accuracy scores could be checked using libraries like Scikit-learn to improve our accuracy.
4) For increasing the performance of our algorithm, we could set up a database for faster calculations.
5) Use of different threads, taking advantage of parallel programming would significantly increase the performance of our algorithm.