CSCI 5832: Homework 3
Text Classification

- By Payoj Jain

Overview

The aim of the assignment is to perform sentiment analysis by implementing text classification system. The model has been trained using collection of positive and negative hotel reviews. Each review has some *unique id.*

Below is the list of all the assumptions that have been taken into account while developing the model:

- New words may or may not occur in the test set.
- If a new word occurs in the test set, the word is simply ignored for computing its likelihood.
- Words appearing in the train and test set are in English Language and thus, no text encoding and decoding is done in the code.
- Tokenization is done by splitting words using white space. I haven't used stemming methods for tokenization as stemming won't improve the accuracy by much. Also, I have omitted punctuations for the same reason.

- To build a valid data set, I have randomly picked 10% of the positive reviews and 10% of the negative reviews from the provided training data set. Tested my model on this valid data set. All the accuracies achieved are based on this valid data set.

Baseline

Implemented standard naïve bayes using unigram language model for both positive and negative classes and labeling the class with higher probability. I have calculated log likelihood for tokens (words) to avoid problem of underflow. Also, implemented Laplace smoothing (add-1 smoothing) to give some probabilities to words which occur in a particular class but didn't occur in the other in the training data. With baseline, I am securing an accuracy between **85-93%**.

Improvement

After implementing naïve bayes, in the hope to get improved performance I tried a modified version of naïve bayes which is called binary multinomial naïve bayes or binary NB. This method uses whether a particular term is present in the document (document frequency) or not instead of the frequency of that term in that document (term frequency). For sentiment analysis (classification) document frequency seems to be more important than term frequency. And as anticipated, implementing this methodology improved model's performance **by almost 1%**.

What more could be done

Handling negation and negative connotation words may help in improving the performance. For eg. sentence like *"I didn't like this movie"* seems to have a negative connotation. The word *"didn't"* negates the sentiment of the word *"like".* Such sentences could be handled by adding prefix NOT_ with all words coming after *"not"* till next punctuation occurs. Thus, these words will act opposite to what they originally act like towards the sentiment of a sentence.

Using **sentiment lexicons** might help as training data size might not be enough to accurately predict sentiments of test set.