Sentiment Analysis using Naïve Bayes Algorithm.

**Overview:**

I have implemented the Naive Bayes algorithm with maximum likelihood and MAP solutions and evaluated it using cross validation on the task of sentiment analysis (as in identifying positive/negative product reviews).

**Text Data for sentiment analysis**

Each dataset is given in a single file, where each example is in one line of that file. Each such example is given as a list of space separated words, followed by a tab character (\t), followed by the label, and then by a newline (\n). Here is an example from the yelp dataset:

Crust is not good. 0

**Implementation:**

First split the dataset into separate train and test sets.

Given a training set for Naive Bayes we will parse each example and record the counts for class and for word given class for all the necessary combinations. These counts constitute the learning process since they determine the prediction of Naive Bayes (for both maximum likelihood and MAP solutions).

Now, given the test set, we parse each example, calculate the scores for each class and test the prediction.

**NOTE for prediction:**

If a word in a test example did not appear in the training set at all (i.e. in any of the classes), then we simply skip that word when calculating the score for this example. However, if the word did appear with some class but not the other then we use the counts we have (zero for one class but non zero for the other).

**Maximum Likelihood and MAP Solutions**

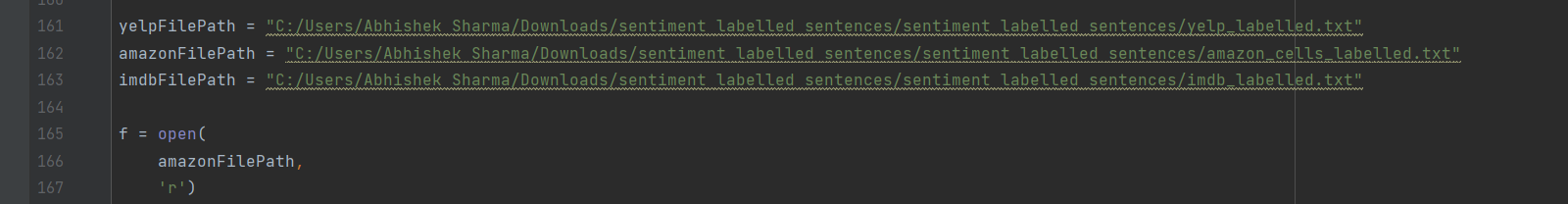
The maximum likelihood (and MAP) estimates of parameters are given by the solution for a Discrete distribution (with a Dirichlet prior) for its parameters.

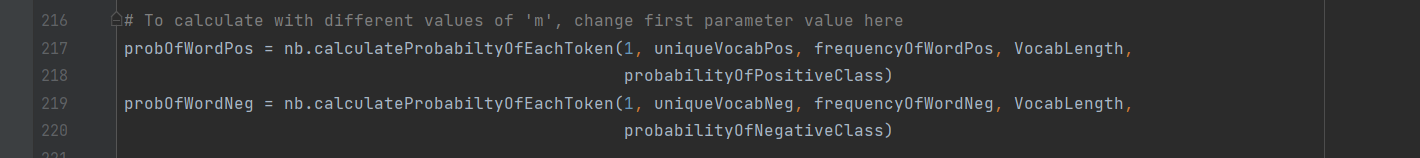
The maximum likelihood estimate of p(w|c) for word w and class c is #(w∧c) / #(c) where #(w ∧ c) is the number of word tokens in examples of class c that are the word w and #(c) is the number of word tokens in examples of class c. If we use a prior with parameter vector where all entries are equal to m + 1, that is, (m + 1)1, the effect is that of adding a pseudo count of m to all entries. In this case, the MAP estimate of p(w|c) is (#(w∧c)+m) / (#(c)+mV) where V is the vocabulary size and other parameters are as above. For example, if m = 1 and V = 1000, and out of 10000 word locations in examples of class c the word w appeared 100 times, the probability is estimated to be

(100+1) / (10000+1000) . This estimate is often referred to as “smoothing” in the literature because it smoothes out the maximum likelihood values and avoids 0/1 extreme solutions. Note that the maximum likelihood solution is simply the special case of smoothing with m=0.

**Steps to run the code:**

1. You will see three variables namely: yelpFilePath, amazonFilePath and imdbFilePath all containing the paths to their respective files. REPLACE THE PATH WITH YOUR LOCAL PATH.

Also change the path variable in f.open() depending on the dataset you wish to run the code.  
  


1. In the part where we calculate the probabilities of each word, replace the value of ‘m’ according to the value you want to test on and run the code.  
     
   
2. The program gives in the output : value of ‘m’ used, list of Predicted Labels, list of Actual Labels and the Accuracy.  
     
   