Name: Sen Lin Perm:6959399

CS165A Machine Problem 1 Report

Architecture:

In my .cpp file, the main class is majorly separated into three different part. with a lot of global variables including timers, counters for total commons, positive commons, negative commons, number of words in positive, negative and total, positive and negative rates. And most important, 2 maps of bags of words. One of them contains positive words from training set, another one contains negative words.

1. The programe read the first file which is the training set of commons. Then stored all the words in that training file into map(bag of words) for positive words and negative words according to the last char of each line of the file. If the character is “0” store words from that line to map of negative words, vise versa. From that, the program also count how many words in total and how many words count as positive and negative. At the same time, a start timer was set at the beginning of this part and end timer at the end. Count and record the running time of this training time.
2. The program read the first file again to guess the common is positive or negative according to the words given by training set using naive bayes classifier with smoothing value. It determine the percentage of a line of words to be positive or negative, then compare with the real answer and finally get the total percentage of correctness and print it out.
3. The program read the second file which is a test file to guess the common is positive or negative according to the words given by testing set using naive bayes classifier with smoothing value. It determine the percentage of a line of words to be positive or negative and then print 1 for positive and 0 for negative. It determine the percentage of a line of words to be positive or negative, then compare with the real answer and

finally get the total percentage of correctness and print it out. At the same time, a start timer was set at the beginning of this part and end timer at the end. Count and record the running time of this testing time.

Preprocessing:

I used bag of words model for this naive classifier. Just like introduced in architecture, this program takes in training sets from file and process it line by line to gather words into the bag of words(Hash Map) separated by positive and negative input. Then calculate the probability of a sentence of words make that sentence a positive review or negative and compare with the real answer to get the rate of correctness.

Model Building:

I used the formular $c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$ given by lecture slides. The P(x_i|c_j) is to divide the time of existence of a word in a map by the total words in positive map. The final calculation formular is:

(trainpositive[tempstr[j]]+smoothing)/(numtrainpositive+smoothing*trainpositive.size()).

And the P(c_j) can be calculated by the counter of number of positive common divide by the number of total commons.(But it seems lower the accuracy, so I removed this part from the actual code) And according to my test, the best performance of correctness of testing data is when smoothing number is 1.1.

Result:

In the given local dataset, my program spent 22.194 seconds to store the training set and 15.285 seconds to run the testing set. The correctness of testing training set is 68.95%, and testing the testing set is 84.89%.

10 most important features for bag of positive words (words with highest probability to be found in positive review): the, and, a, to, i, you, of, it, is, game .
10 most important features for bag of negative words (words with highest probability to be found in negative review): the, to, and, i, a, it, of, you, game, is .
Most of these words are set to be ignored in testing function since most of them are stop words and has no attitude of positive or negative.

Challenges:

I was confused by the formula about weather I should put possibility of positive words over all words or just positive words, or even the number of positive common divided by number of all commons. I also tried many different number for smoothing until it finally makes me satisfy. And I also didn't know how to write a timer clock to count the time of program running, so I searched it up and learnt from it.

Weakness:

Overall, the persentage of correctness in the outcome of the program isn't hight while running in local. I got only 85.00% of correctness rate after running training and testing locally. To increase the correct rate, we can set the program to ignore more stop words like "I" "a" "an". Because the exist rate of these words is more depend on how training set tilted towards (More negative common in training set would cause the higher rate of this words to be considered as negative).