

## CS165A Machine Problem 1 Report

## Architecture:

In my .cpp file, the main class is majorly separated into three different parts. 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 the training set, another one contains negative words.

1. The program read the first file which is the training set of commons. Then store all the words in that training file into the 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 counts 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 an 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 the training set using a naive Bayes classifier with a smoothing value. It determines the percentage of a line of words be positive or negative, then compares with the real answer and finally gets the total percentage of correctness and prints 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 the testing set using a naive Bayes classifier with a smoothing value. It determines the percentage of a line of words being positive or negative and then print 1 for positive and 0 for negative. It determines the percentage of a line of words be positive or negative, then compares

with the real answer and finally gets the total percentage of correctness and prints it out. At the same time, a start timer was set at the beginning of this part and an end timer at the end. Count and record the running time of this testing time.

Preprocessing:

I used a bag of words model for this naive classifier. Just like introduced in architecture, this program takes in training sets from file and processes them 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 making that sentence a positive review or negative and compare with the real answer to get the rate of correctness.

Model Building:

I used the formula 
$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_{i \in positions} P(x_i | c_j)$$
 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 a positive map. The final calculation formula is:

$$(\text{trainpositive}[\text{tempstr}[j]] + \text{smoothing}) / (\text{numtrainpositive} + \text{smoothing} * \text{trainpositive.size}())$$

And the  $P(c_j)$  can be calculated by the counter of the number of positive common divided 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 the 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 the testing training set is 68.95%, and testing the testing set is 84.89%.

10 most important features for a bag of positive words (words with the highest probability to be found in positive review): the, and, a, to, I, you, of, it, is, game.

10 most important features for a bag of negative words (words with the 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 the testing function since most of them are stop words and have no attitude of positive or negative.

#### Challenges:

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

#### Weakness:

Overall, the percentage of correctness in the outcome of the program isn't high while running locally. I got only an 85.00% of the 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 existing rate of these words is more dependent on how training set tilted towards (More negative common in training set would cause the higher rate of these words to be considered as negative).