Machine Problem 1 Report

Brian Yan

Code Architecture:

The code architecture is very simple. It features a class named Review, a class named Category, the main class NaiveBayesClassifier and the training class NaiveBayesTrainer. The Review class stores each review in a String array named words and has a Category variable which states what category the review is. The category class is a basic enum that states whether it is positive or negative. The main class Naïve Bayes Classifier runs the program and calls functions to train and classify the results. The NaiveBayesTrainer contains the bulk of the code. First, it parses the File given into Reviews. Then, It trains the NaiveBayesClassifer. Finally, when it comes time to classify the testing set, it calculates the probabilities of its most likely category.

Preprocessing:

I represented the product review document as the words that were in the review. I split the Review into a String array where the first index was its Category and the rest were the words that the Review was made up of. In addition, I made sure not to include punctuation and stop words as it improved the running time a very small percentage of the accuracy.

Model Building:

I trained the Naïve Bayes Classifier by counting the frequency of the words that showed up both in a positive and negative views. Then I calculated the probability of a word given a certain category using Bayes Theorem. Everything is stored in a HashMap as the word is the key and the value was an ArrayList where frequency and probabilities were stored in its indexes. The Probability of a word given a Positive or Negative Category are in the ArrayList. This was the basis of training the Naïve Bayes Classifier.

Results:

The accuracy I have obtained on the provided data sets are the following. On the
Training Data Set : 88.8% accuracy
Testing Data Set: 78.2% accuracy
The run time for Training is O(D*Ld + |C| |V|) where D represents # of reviews and Ld represents average length of those reviews, C is the category, and V is the total vocabulary
Running time for Testing Time is O(|C| * Ld) where Ld represents average length of Reviews
Top 10 words for Positive Category
Christmas, Exceptional, Wonderful, Traditional, favourite, superb, vulernable, engagement, favorites, burns, rocks
Top 10 words for Negative Category
Gore ludicrous lacks Boredom awful
Ridiculous fails Horrib gruesome lame bland

Challenges:
The most difficult challenge I faced was starting the project. I wasn't exactly sure how I wanted to organize the structure of the project nor how to calculate the probabilities of each word. But after reading the lecture slides and a few more documents online, I got a better idea of how to store certain features and calculate the probability. Also, I found it difficult to implement a weighting feature to the naive Bayes Classifier since there are some words that should be considered more heavily than others.

Weaknesses:
I think the biggest weakness I have is that my Naïve Bayes Classifier assumes that every word is independent from each other word in the review. I think that words/sentences are really dependent on their position in the review. For example, the last sentence in the review should be weighted heavily since it often sums up what the writer is stating. Also, one can say, this is not good. But my NaiveBayesClassifer has no way of figuring that the review is negative since it assumes that each word is independent from each other in terms of position. One solution that I would like to see implemented is