Topic: Binary classification of sentence sentiment

Name: Jie Liu   SUNYID: 001311503

Part 1. Data Collection
1) I download data form UCI Machine Learning Repository.
Name: Sentiment Labelled Sentences Data Set --- amazon_cells_labelled.txt
Link : https://archive.ics.uci.edu/ml/machine-learning-databases/00331/
My data file name is "amazon_cells_labelled.txt". It contains 500 positive comments (labeled as 1) and 500 negative comments (labeled as 0) from amazon reviews.
2) I plan to do binary classification on sentence sentiment and this data file contains sentiment sentences from amazon reviews, which totally matches my topic.
3) I think this model will generate an accuracy of around 0.8 on training data, and accuracy of around 0.6 on testing data.

Part 2. Exploratory Data Analysis
I put all the labeled sentiments sentences to word cloud and got figure named 'wordCloud'. In word cloud language icon, I choose show word count and then I got the frequency of each word, which is saved in "Frequent_words.csv". Figure 2 is 'boxplot' plotted by matplotlib.

1) Form this figure we can see, the most frequent words are phone, quality, great, product, good, headset, sound, recommend, battery and so on.
Form figure 2 we can see, more than 90% words' frequency are below 75, while 2 words' frequency around 300, 2 words around 200, and others below 150.
2) Ten most frequent tokens:
Phone, quality, great, product, good, headset, sound, recommend, battery and use.
3) Feature set:
Words with frequency greater than 8 (except common stop words): phone, quality, great, product, good, headset, one, sound, recommend, battery, but, use, like, recommend, work, case, price, no, up, service, money, ear, time, new and so on.

Part 3: Preprocessing
Preprocess method: 1) change all words to lower character
2) Eliminate common stop words. Such as "a", "about", "above", "above", "across", "after", "afterwards", "again", "against" nearly 300 common stop words in my code.
3) Selected 39 most frequency words (frequency greater than 8). I use a 1×39 vector to represent each sentiment sentence. Check each sentiment sentence' words to see if it has these frequency words. If it has, then it is a 1 in the corresponding position in the vector. Using same way to all the sentences, we change our input data into 600×39 matrix.

Why do I do this? First "Great" and "great " are same words. They are expressing the same meaning; we don't need to count it twice. Second, there are many common stop words like "the", "a". They appear in many sentence, no matter the sentence sentiment is positive or negative. Thus, we need to keep them as features. Third, I plan to use SVM. So I need to change the format of input data from text to numeric form.

Part 4: Classification
I use SVM and ten folder cross validation on train sentences and compute their accuracy on train process and test process respectively. The python code is "sentiment_classification.py"(using part of code from data mining class which is provided by Professor Feng Chen). Training Data file is "amazon_labelled.txt", which contains 600 labeled sentences. Test data file is "amazon_test.txt", which contains 400 labeled sentences.

Report question:
1) F1 score: 0.598930481283
    Precision score: 0.684749849307
    Recall score: 0.623042954637
F1 score works as a weighted average of the precision and recall. Precision score is the ratio of the number of true positive and detected true positive. The recall is the ratio of number of true positive and all real positive. I need these score to know how man true positive can be correctly labeled as positive, same as negative.
2) Precision score: 0.684749849307
It is a little lower than my expectation about SVM model accuracy. I choose the words that frequency is greater or equal to 8. But for some test sentences, they don't have these words. SVM model cannot detect these sentences as positive or negative.
3) I put all the positive and negative sentiment sentences together to find the most frequent words. But in some condition one word may equally exist in positive and negative sentiment sentences. In this condition, this feature is useless to distinguish positive and negative sentiment sentences.
4) Decrease the required frequency and put more words in feature set.
We could use linguistic inquiry and word count to subcategories the words of positive and negative sentiment sentences into different word subcategories.

Additional part:
1)SVM ten folder cross validation.
2) F1 score: 0.598930481283
    Precision score: 0.684749849307
    Recall score: 0.623042954637
3) Train: ten foler cross validation Model accuracy = 0.736666666667
    Precision score: 0.684749849307
 The accuracy decreases from train process to test process.
4) I think the reason that there is accuracy difference between train process and test process is that our feature set are the words that frequently exist in training data file, while these words may not frequently exist in test data file. Therefore, the accuracy will decrease.