MP1 Report

Lan Liu 7956394
CSIL Account: liulan19920228

python version: 2.7.11
To run the code:

```
./NaiveBayesClassifier training.txt testing.txt          or
python NaiveBayesClassifier.py training.txt testing.txt
```

• Architecture

My program consists of NaiveBayesClassifier.py, stopword.txt. It contains a class: NaiveBayesText, and process following functionalities:

def negated(DataSet,stopword):
Preprocess dataset by convert word to lower case, remove stopword and nonalphanumeric, handle negation, remove label from review.

def MutualInformation(self,word):
Compute the mutual information of each word.

def Feature(self):
Rank the feature by the descending order of mutual information

def Train(self,TrainDataSet,Label_Set):
Construct hash table to store the key features and the value frequency of features. Remove the features that only occur once, select the top 2000 features to do the classification.

def SingleClassify(self,SingleReview):
Classify a single review using Naïve Bayes Classification model. Do smoothing.

def NaiveBayesClassify(self, TestDataSet):
Classify a set of reviews.

My program also consists of Stemmer.py, this is borrowed from reference 3. (As instructed, we can borrow code for stemming, while using stemming library is not allowed.)

• Preprocessing:

My program preprocesses both training data and testing data by doing the following things:

1. Read the review set by lines, and thus review set will be a list of reviews.
2. For each review in the review list, split it into a list of words. I.e., the review set will be list of list.
3. Remove the labels (0 and 1) from the review set, and generate the label set.
4. Convert all words to be lowercase.
5. Handle negation by transferring not&n't to not_ (not good becomes not_good)
6. Remove stopwords and <br/br/>
7. Remove non alphanumeric such as punctuation. Remain "?" and "!" since they might express negative feelings.
8. Do stemming (exciting, excited, excite will be the same word) –borrowed code

After these steps, each review set will become a list of list of words, label set will become a list of labels. Then we can use dictionary in python to count the frequency of words. (detailed below)

• **Model Building**:
After the previous step, we get Review set: list of list of words. Label set: list to labels 0,1. And we have two parts to build the model.
Part 1: Data structure to store the features
My model is using the class NaiveBayesText, and the information is stored in Hash table. First, merge all positive reviews with label 1 to a new positive list, and all negative reviews with label 0 to another new negative list. Second, use dictionary in python to store the frequency of words in each list. For example, for the positive list, self. MoiveReview[1] will be a hash table so that **key is the word and value is frequency of the word**. All keys in the hash table will be the features.

Part 2: Feature Selection
First, we remove the features which only appear a few times up to a certain threshold value. Second, we compute the Mutual Information of each feature, and select the top k features with maximum mutual information. The mutual information of X and Y are of form as follows:

$$MI(X,Y) = \sum_{y \in Y} \sum_{x \in X} P(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)}\right)$$

Here X means each individual feature and Y means the positive or negative class.Then we can use Naïve Bayes Model to predict the label of a movie review by only looking at the selected features.

• **Results**:
The result of the given training.txt and testing.txt are as follows:
27 seconds (training)
3 seconds (labeling)
0.926 (training)
0.852 (testing)

Top 10 Features:
Based on the Mutual Information, we select 10 features with the largest MI value.

bad
?
worst
great
awful
waste
boring
terrible
worst
excellent

We can see that "?" is the second important feature which verify our statement that "?" and "!" might convey negative feelings before.

• **Challenges**:

The challenge I faced is about how to sort the features by their importance, and then I found reference1 online that can compute mutual information, but still, even though I can sort the features by mutual information, it is hard to say how many top features I shall select to apply on the Naïve Bayes Classify process. To figure this, I start from 1000 features, and add 100 at a time, and check which value will achieve the best accuracy for doing the classification. And it turns out that 2100 features will be the best to achieve the accuracy 0.848.

• **Weaknesses**:

1. The code can work well on 2 classes classification problem, but need to modify for multi classes problem.
2. The feature selection process does not increase the accuracy a lot, from 0.820 to 0.832, there must be better ways to assign weights to different features. If I have more time to work on it, I will try to compute information gain to further select the features, and also assign more weight to the evaluation type of words.

Reference:
1. https://en.wikipedia.org/wiki/Mutual_information
2. https://arxiv.org/pdf/1610.09982.pdf
3. https://github.com/mdirolf/pyporter2