Steps for running the program:

------------------------------------------------------------------------------------------------------------------------

1. Install Python if not already installed from https://www.python.org/downloads/ version 2.7.10

2. Unzip the file "Akshaya\_Khare\_Mohit\_Gupta\_CS6200\_HW6.zip" to get the contents

3. The file 'nbtrain.py' is the file which contains the source code to prepare the model-test file from the given training

directory.

nbtrain <training-directory> <model-file>

The file is called as ->

./nbtrain.py <training-directory> <model-file>

WHERE: <training-directory> is the path of training-directory

<model-file> is name of the model-file

4. The "model-file.txt" will contain the json dump of two dictionaries containing negative and positive probabilities of all the words except for the words which have frequency less than 5 over the whole collection.

5. The file 'nbtest.py' is the file which contains the source code to classify the given list of documents into either a

negative or positive class, using the 'model-file' which was created in 'nbtrain.py'.

The file is called as ->

./nbtest.py <model-file> <input-directory> <predictions-file>

WHERE: <model-file>

<input-directory> (which contains the folder containing files which need to be classified)

<predictions-file>(which contains the file-name and scores model assigned to the possibilities of positive and negative review)

------------------------------------------------------------------------------------------------------------------------

Results:

1. Naive Bayes classifier using unigram + add-1 (Laplace) smoothing with terms fewer than 5 times ignored

75 % of positive reviews in the development data were correctly classified

80 % of negative reviews in the development data were correctly classified

90 % of reviews in the test data were classified as positive

110 % of reviews in the test data were classified as negative

2. The below requested data is in "A list of the 20 terms with the highest (log).txt"

A list of the 20 terms with the highest (log) ratio of positive to negative weight.

A list of the 20 terms with the highest (log) ratio of negative to positive weight.

3. The Prediction file for test and dev have been created separately as mentioned below:

“Prediction-file-test.txt” contains the predictions for test files

“Prediction-file-dev.txt” contains the predictions for dev files

NOTE: All the “.txt” files should be opened in Notepad++ or the equivalent editor in Linux machine

In the smoothing folder, the two predictions file have been added a suffix “D” and “JM” to distinguish each smoothing.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Extra Credit: 20 points

In the assignment above, we specified which features to use and how to smooth the probability distributions. For extra credit, try other features besides unigrams that occur five times or more, other smoothing methods and parameters, and so on. The basic naive Bayes training and testing procedure should remain the same, of course. You can use the development data to track your progress. Describe the new features you tried and their accuracy on the development data in your README. In addition to the results from the assignment above, turn in the modified code and its predictions on the test data.

NOTE: The smoothing techniques are commented out in our code, but can be easily identified and be used, in the folder \Extra-credit\Smoothings.

The following features + smoothing methods were experimented:

1. unigram + Removal of special characters (all non-alphabetic characters) + Laplace smoothing

2. Bigram + Laplace smoothing

3. unigram + Jelinek-Mercer (JM) smoothing

4. unigram + Dirichlet (Dir) smoothing

Accuracy on development data:

1. unigram + Removal of special characters (all non-alphabetic characters) + Laplace smoothing

79 % of positive reviews in the development data were correctly classified

77 % of negative reviews in the development data were correctly classified

94 % of reviews in the test data were classified as positive

106 % of reviews in the test data were classified as negative

2. Bigram + Laplace smoothing

80 % of positive reviews in the development data were correctly classified

84 % of negative reviews in the development data were correctly classified

89 % of reviews in the test data were classified as positive

111 % of reviews in the test data were classified as negative

3. unigram + Jelinek-Mercer (JM) smoothing

75 % of positive reviews in the development data were correctly classified

80 % of negative reviews in the development data were correctly classified

90 % of reviews in the test data were classified as positive

110 % of reviews in the test data were classified as negative

4. unigram + Dirichlet (Dir) smoothing

69 % of positive reviews in the development data were correctly classified

80 % of negative reviews in the development data were correctly classified

93 % of reviews in the test data were classified as positive

107 % of reviews in the test data were classified as negative

Conclusion:

It is observed that Bigram + Laplace smoothing classifies the development data close to the actual classification of development data compared to other features + smoothing.

The excel file “Smoothing\_Graphs.xls” contains the graph which compares the different accuracies for smoothing techniques and other features.

References:

----------------------------------------------------------------------------------------------------------------------------

http://stackoverflow.com/-> Helping out for various logic and syntax issues in python

https://www.python.org/doc/

https://web.stanford.edu/class/cs124/lec/naivebayes.pdf -> For implementing normal smoothing and classifier

http://www.ntu.edu.sg/home/gaocong/papers/wpp095-yuan.pdf -> Used to implement Dirichlet and Jelinek-Mercer smoothing

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*