INSTRUCTIONS

1.1 Run Naïve Bayes on movie review data

Input Files:

- Rotten Tomatoes review containing positive review sentences.
- Rotten Tomatoes review containing negative review sentences.

Aim:

To apply Naïve Bayes rules on the Rotten Tomatoes review data.

Methodology:

Divide the Rotten Tomatoes reviews into 10% testing set and 90% training set.

1. Build the positive words dictionary and negative words dictionary out of the training set.
2. Calculate important probability values, $p(word|negative)$ and $p(word)$

$$p(word|positive) = \frac{N_{word_{pos}}}{N_{all_{pos}}}$$

$$p(word|negative) = \frac{N_{word_{neg}}}{N_{all_{neg}}}$$

$$p(word) = \frac{N_{word}}{N_{all_{word}}}$$

where

$\boldsymbol{N_{word_{pos}}}$	equals the number of times a word appears in the positive dictionary.
$\boldsymbol{N_{word}}$	equals 1, when there is no showing up of the word in the dictionary, which is a smoothing method.
$\boldsymbol{N_{all_{pos}}}$	equals the number of all the positive sentiment words from the training set (words are counted if it appears repetitively.)
$\boldsymbol{N_{word}}$	equals the number of times a word appears in both dictionary
$\boldsymbol{N_{all_{word}}}$	equals the number of all words.

3. Apply the Naïve Bayes Rules on the test set data, where the following two probabilities is assumed to be known:

   $p(positive) = 0.5$

   $p(negative) = 0.5$

   which means that the fraction of either positive or negative reviews in the dataset is 0.5. The core algorithm can be illustrated as following: Sentence S_i(i = 1,2,3 ...) from the testing set contains $\boldsymbol{K_i}$ words, among them there are k words satisfy the condition of $p(word) > 0.00000001$ ($k \in \boldsymbol{K_i}$)

$$p(predicate_{positive}) = p(positive)\prod_k p_k(word|positive)$$

$$p(predicate_{negative}) = p(negative)\prod_k p_k(word|negative)$$

if $p(predicate_{positive}) > p(predicate_{negative})$, then it is predicted that the testing sentence had positive sentiment, the vice versa.

1.2 Most useful words

Aim:

Find the most useful words (or bigrams) for deciding sentiment.

Methodology:

Function $mostUseful()$ is used to print the most useful words (including bigrams) are used for deciding sentiment by looking at the proportion of

$$Power(word) = \frac{p(word|positive)}{p(word|positive) + p(word|negative)}$$

Where the bigger $Power(word)$ is, the more useful a word is.

A function $matchDict()$ is developed to calculate how many words labeled as “most useful” are from the sentiment dictionary.

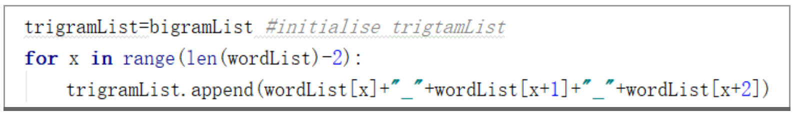
1.3 Include trigrams

Aim:

To include trigrams in the algorithm

Methodology:

The following algorithm is developed to include trigrams. It’s worth noticing that trigramList consists of unigrams, bigrams and trigrams.



```
trigramList=bigramList #initialise trigtamList
for x in range(len(wordList)-2):
    trigramList.append(wordList[x]+"_"+wordList[x+1]+"_"+wordList[x+2])
```

1.4 Rule based system

Input:

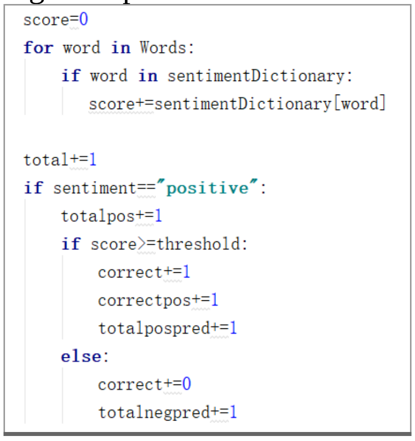
- A positive sentiment dictionary containing 2006 positive-sentiment words.
- A negative sentiment dictionary containing 4783 negative-sentiment words
- Rotten Tomatoes Training Data and All Nokia product review data.

Aim:

Use Rule based approach to do the sentiment classification.

Methodology:

1. First process is to assign values to the words in the sentiment dictionary. In the dictionary, if the word is positive sentiment, it will be assigned a value of “1”, and if it is negative, it will be assigned a value of “-1”.
2. For the sentences in testing set, only words that match the sentiment dictionary will be counted. For the matched words in the testing set, to sum up each the word’s value (refereeing to the sentiment dictionary) and compare to the threshold, if it is bigger than the threshold, then the word will be classified as positive. The vice versa.
3. In the core algorithm, when the number of positive and negative words are equal (score=0), the sentence is classified as positive (due to the operator >=). So, the calculation has more weight for positive classification.



```
score=0
for word in Words:
    if word in sentimentDictionary:
       score+=sentimentDictionary[word]

total+=1
if sentiment=="positive":
    totalpos+=1
    if score>=threshold:
        correct+=1
        correctpos+=1
        totalpospred+=1
    else:
        correct+=0
        totalnegpred+=1
```

Theoretically, it is more reasonable to set the threshold to be smaller than 0 for movie comments. Because people are likely to discuss the plot of a movie in the comments, and when the targeted movie is a tragedy movie, people might make comments like “This is a sad story” or “It describes a frustrating story.” Where sad and frustrating can be classified as negative sentiment.

4. Multiple trials have shown what most accuracy is generated when threshold of either Movie comments or Nokia comments are set as “0”.