Name: Magesh Rajasekaran

UNM ID: 101676478

Mail Id: mrajasekaran@unm.edu

Project: Naïve Bayes Classifier

High Level Code Description:

For NB_Project2.py:

1) Imported all the necessary libraries for using scipy, numpy python.
2) Opted Numpy textread to read all the training and testing datasets inclusive of vocabulary.txt
3) Created a counter to find the number of occurrences of doc ids in the train label. And printed out the counter values, hardcoded it to create a numpy array of MLE P(Y) (Maximum Likelihood)
4) Created MAP matrix from the training datasets which will tell me the frequency of words for the doc ids. (Matrix of 20*61188)
5) Created a sparse matrix to find the $P(Y_{new})$ which we can calculate from argmax $[\log_2(P(Y_k))+\sum X_i*\log_2 P(X_i|Y_k)]$ (Matrix of 20*7505)
6) Found the maximum of each column and this will return the doc ids of each column.
7) Create the confusion matrix with the testlabel and the maximum_value from the above step.
8) This will give you the accuracy of the classification.

For NB_Project2_Method.py:

1) This python follows the same as the one above.
2) This is used to classify the highly ranked words in the document.
3) With the MAP found, I am taking the value of maximum number of times a word is repeated, which will give me an array of 61188 values.
4) I am finding the mean of the above created array, then by setting the mean as the threshold value, I am taking the words which are repeated more than 10 times in any number of documents.
5) To get the item frequency, I am calling a method called itemfreq() which will give me the number of times the word has been repeated in the array.
6) To ignore the same words repeated, I am implementing a method which will check whether the same word id is being repeated in one or more doc ids, if that's repeated in more than two doc ids then am removing the word id from the array
7) Thus, with the final word ids, I am calling vocabulary.txt to print the words.

Question 1:

As given the samples are less that is 1000 documents which has only 1000 words in it, the possibilities of placing the 1000 words in a document is easy when compared to placing at the least of 50, 000 words in a document. So, by having small sample set, there can be many possibility that $P(X_i|Y)$ is equal to the $P(X_j|Y)$ (most likely to be equal). Hence it will be hard to estimate the parameters of this model given 1000 words in a document.

Question 2:

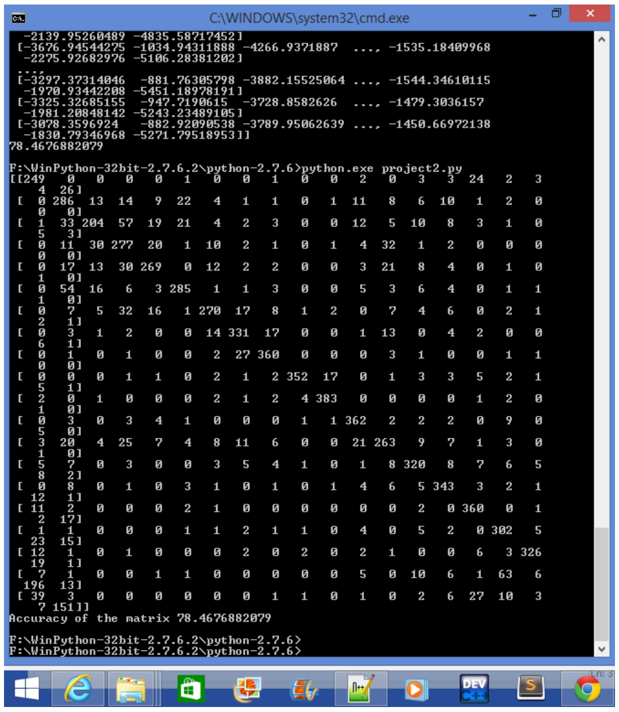
Total Accuracy = ((Number of correctly classified documents)/ (Number of test documents))*100

Number of correctly classified documents: 5889

Number of test documents: 7505

Total Accuracy: 78.46 %

Confusion Matrix:



```
  -2139.95260489 -4835.58717452]
 [-3676.94544275 -1034.94311888 -4266.9371887  ..., -1535.18409968
  -2275.92682976 -5106.28381202]
 ...,
 [-3297.37314046  -881.76305798 -3882.15525064 ..., -1544.34610115
  -1970.93442208 -5451.18978191]
 [-3325.32685155  -947.7190615  -3728.8582626  ..., -1479.3036157
  -1981.20848142 -5243.23489105]
 [-3078.3596924   -882.92090538 -3789.95062639 ..., -1450.66972138
  -1830.79346968 -5271.79518953]]
78.4676882079

F:\WinPython-32bit-2.7.6.2\python-2.7.6>python.exe project2.py
[[249   0   0   0   0   1   0   0   1   0   0   2   0   3   3  24   2   3
    4  26]
 [  0 286  13  14   9  22   4   1   1   0   1  11   8   6  10   1   2   0
    0   0]
 [  1  33 204  57  19  21   4   2   3   0   0  12   5  10   8   3   1   0
    5   3]
 [  0  11  30 277  20   1  10   2   1   0   1   4  32   1   2   0   0   0
    0   0]
 [  0  17  13  30 269   0  12   2   2   0   0   3  21   8   4   0   1   0
    1   0]
 [  0  54  16   6   3 285   1   1   3   0   0   5   3   6   4   0   1   1
    1   0]
 [  0   7   5  32  16   1 270  17   8   1   2   0   7   4   6   0   2   1
    2   1]
 [  0   3   1   2   0   0  14 331  17   0   0   1  13   0   4   2   0   0
    6   1]
 [  0   1   0   1   0   0   2  27 360   0   0   0   3   1   0   0   1   1
    0   0]
 [  0   0   0   1   1   0   2   1   2 352  17   0   1   3   3   5   2   1
    5   1]
 [  2   0   1   0   0   0   2   1   2   4 383   0   0   0   0   1   2   0
    1   0]
 [  0   3   0   3   4   1   0   0   0   1   1 362   2   2   2   0   9   0
    5   0]
 [  3  20   4  25   7   4   8  11   6   0   0  21 263   9   7   1   3   0
    1   0]
 [  5   7   0   3   0   0   3   5   4   1   0   1   8 320   8   7   6   5
    8   2]
 [  0   8   0   1   0   3   1   0   1   0   1   4   6   5 343   3   2   1
   12   1]
 [ 11   2   0   0   0   2   1   0   0   0   0   0   0   2   0 360   0   1
    2  17]
 [  1   1   0   0   0   1   1   2   1   1   0   4   0   5   2   0 302   5
   23  15]
 [ 12   1   0   1   0   0   0   2   0   2   0   2   1   0   0   6   3 326
   19   1]
 [  7   1   0   0   1   1   0   0   0   0   0   5   0  10   6   1  63   6
  196  13]
 [ 39   3   0   0   0   0   0   0   1   1   0   1   0   2   6  27  10   3
    7 151]]
Accuracy of the matrix 78.4676882079

F:\WinPython-32bit-2.7.6.2\python-2.7.6>
F:\WinPython-32bit-2.7.6.2\python-2.7.6>
```

confusion_matrix.txt

Question 3:

As you can see the confusion matrix which will tell you where it classified wrongly. The doc id 19(row) to the doc id 17(row) got 63 documents and also the doc id 6(row) to the doc id 2(column) got 54 documents classified wrongly.

Doc id 19: talk.politics.misc

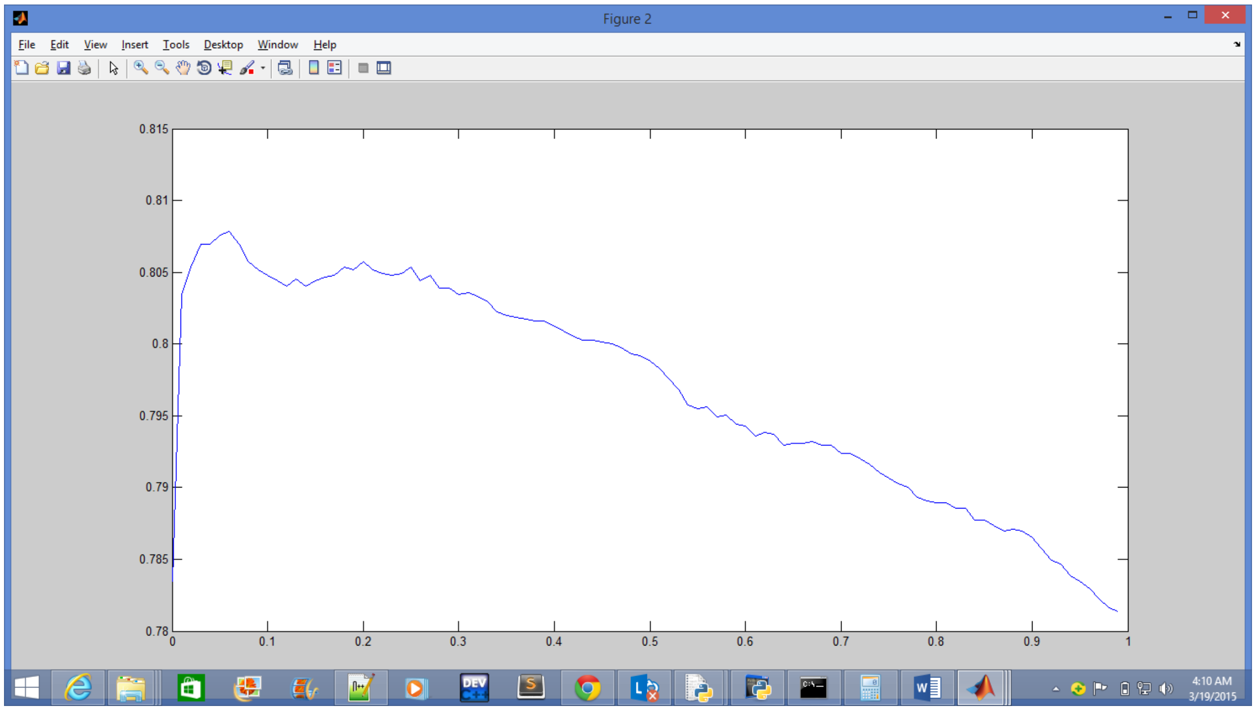
Doc id 17: talk.politics.guns

Doc id 6: comp.windows.x

Doc id 2: comp.graphics

Question 4:

As the beta value goes down, i.e. decreases, we get the lower values for MAP (P (X|Y)). Proabability value of many training documents will tend to zero and when the classification is applied on testing documents, the system tends to overfitting, by learning the noise of the training datasets and so it classifies the testing documents wrongly. When the beta value goes up, the testing data doesn't fit well on the training data and there will be high chance of getting Underfitting problem.



Accuracy value	**Beta value**
78.3477681545636	0.00001
78.8674217188541	0.000100000000000000
79.6802131912059	0.00100000000000000
80.3464357095270	0.0100000000000000
80.4796802131912	0.100000000000000
78.1079280479680	1

<u>Question 5:</u>

Method to find the highest rank of the words which are classified. [Used Maximum Likelihood]

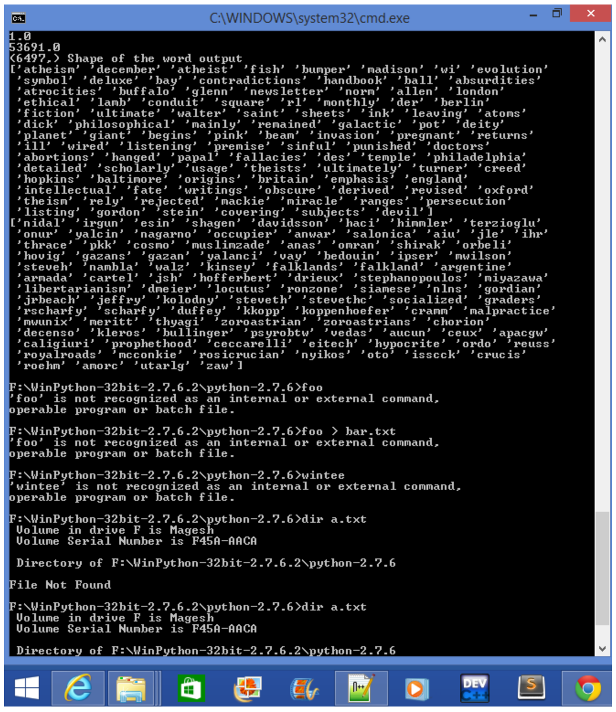
At first, with the matrix P (X|Y) which has the frequency of the words repeated for each doc id and the vocabulary, I have found the maximum Probability of the each words repeated in all the documents and then maximum times of the words repeated in each documents and stored in array named maximum_value_threshold. Then, I found the mean of the array maximum_value_threshold to set the threshold for the words which are repeated in each documents. The mean after rounded off, I got 10. I created a matrix, rank_matrix, which takes the value of all the words (can also have repeated word ids because one word may repeat in other documents also) above the threshold 10. With this matrix, I found the item frequency of it, which will give me the exact count of the each word ids repeated in it. This mean that it tells me how many documents have the same word. I have implemented a method to remove the word ids if it has been repeated in more than two doc ids (which means it is repeated in more than two news groups). With this, I got an array new_matrix which has the word ids (which are not repeated more than two documents).   And by using the new_matrix which has the word ids of the vocabulary, I retrieved the words and printed the top 100 and bottom 100 of it.

STEPS: (Used Maximum Likelihood $P(Y_k)$)

1) Find the Maximum Probability of $P(X_i|Y)$ and choose the "i", find the maximum number of i repeated in the Y_k , and store it in mean_array, where i = words' id and k = document' id
2) Find the mean of the mean_array, and have that as the threshold, where mean is a number of average time every word is repeated[X_{mean} value]
3) Find the word id's (i) which are repeated more than the mean number of times in every documents. [$X_i > X_{mean}$] (word id can be repeated since one word may occur in multiple documents)
4) Choose word id's which are not repeated more than two documents by using the itemfreq() method and store it in new_matrix.
5) With the word id's we have in the new_matrix, we can get the words from the vocabulary.txt

Question 6:

Please find the python program NB_Project2_Method.py for the implementation of this method, I have clearly mentioned how to execute this in readMe.txt and python program is also commented thoroughly.



```
1.0
53691.0
(6497,) Shape of the word output
['atheism' 'december' 'atheist' 'fish' 'bumper' 'madison' 'wi' 'evolution'
 'symbol' 'deluxe' 'bay' 'contradictions' 'handbook' 'ball' 'absurdities'
 'atrocities' 'buffalo' 'glenn' 'newsletter' 'norm' 'allen' 'london'
 'ethical' 'lamb' 'conduit' 'square' 'rl' 'monthly' 'der' 'berlin'
 'fiction' 'ultimate' 'walter' 'saint' 'sheets' 'ink' 'leaving' 'atoms'
 'dick' 'philosophical' 'mainly' 'remained' 'galactic' 'pot' 'deity'
 'planet' 'giant' 'begins' 'pink' 'beam' 'invasion' 'pregnant' 'returns'
 'ill' 'wired' 'listening' 'premise' 'sinful' 'punished' 'doctors'
 'abortions' 'hanged' 'papal' 'fallacies' 'des' 'temple' 'philadelphia'
 'detailed' 'scholarly' 'usage' 'theists' 'ultimately' 'turner' 'creed'
 'hopkins' 'baltimore' 'origins' 'britain' 'emphasis' 'england'
 'intellectual' 'fate' 'writings' 'obscure' 'derived' 'revised' 'oxford'
 'theism' 'rely' 'rejected' 'mackie' 'miracle' 'ranges' 'persecution'
 'listing' 'gordon' 'stein' 'covering' 'subjects' 'devil']
['nidal' 'irgun' 'esin' 'shagen' 'davidsson' 'haci' 'himmler' 'terzioglu'
 'onur' 'yalcin' 'nagarno' 'occupier' 'anwar' 'salonica' 'aiu' 'jle' 'ihr'
 'thrace' 'pkk' 'cosmo' 'muslimzade' 'anas' 'omran' 'shirak' 'orbeli'
 'hovig' 'gazans' 'gazan' 'yalanci' 'vay' 'bedouin' 'ipser' 'mwilson'
 'steveh' 'nambla' 'walz' 'kinsey' 'falklands' 'falkland' 'argentine'
 'armada' 'cartel' 'jsh' 'hofferbert' 'drieux' 'stephanopoulos' 'miyazawa'
 'libertarianism' 'dmeier' 'locutus' 'ronzone' 'siamese' 'nlns' 'gordian'
 'jrbeach' 'jeffry' 'kolodny' 'steveth' 'stevethc' 'socialized' 'graders'
 'rscharfy' 'scharfy' 'duffey' 'kkopp' 'koppenhoefer' 'cramm' 'malpractice'
 'mwunix' 'meritt' 'thyagi' 'zoroastrian' 'zoroastrians' 'chorion'
 'decenso' 'kleros' 'bullinger' 'psyrobtw' 'vedas' 'aucun' 'ceux' 'apacgw'
 'caligiuri' 'prophethood' 'ceccarelli' 'eitech' 'hypocrite' 'ordo' 'reuss'
 'royalroads' 'mcconkie' 'rosicrucian' 'nyikos' 'oto' 'isscck' 'crucis'
 'roehm' 'amorc' 'utarlg' 'zaw']

F:\WinPython-32bit-2.7.6.2\python-2.7.6>foo
'foo' is not recognized as an internal or external command,
operable program or batch file.

F:\WinPython-32bit-2.7.6.2\python-2.7.6>foo > bar.txt
'foo' is not recognized as an internal or external command,
operable program or batch file.

F:\WinPython-32bit-2.7.6.2\python-2.7.6>wintee
'wintee' is not recognized as an internal or external command,
operable program or batch file.

F:\WinPython-32bit-2.7.6.2\python-2.7.6>dir a.txt
 Volume in drive F is Magesh
 Volume Serial Number is F45A-AACA

 Directory of F:\WinPython-32bit-2.7.6.2\python-2.7.6

File Not Found

F:\WinPython-32bit-2.7.6.2\python-2.7.6>dir a.txt
 Volume in drive F is Magesh
 Volume Serial Number is F45A-AACA

 Directory of F:\WinPython-32bit-2.7.6.2\python-2.7.6
```

words_classified.txt

Question 7:

It’s very common that, for the future performance in the real world with the same training dataset, it is not advisable to use the same training datasets. Because day by day usage of words are getting changed a lot. It’s proven from a study that people have started using the bible words which they have never used before, so when usage of words change, using the same training datasets will give you real problem. The words like “Scharfy, Nidal” these words were used in the past. Nidal means “Fight” in Arabic language called Urdu, and Scharfy is a german surname. I don’t really think people will use the same set of words in the future, so it’s not always advisable to use the same training datasets to predict the future.

References & Citations:

[1] www.stackoverflow.com, tag# numpy, scipy, python, etc.

[2] www.docs.spicy.org, for referencing the python syntaxes

[3] http://pydoc.net/Python/scikit-learn/0.15.1/sklearn.cluster.tests.test_hierarchical/, warning catches in the sklearn import for confusion matrix

[4] Referenced Piazza codes, discussions and formula for implementing MAP and Classify

[5] Stanford University Video which helps understand the logic behind MAP