Project-2: Naïve Bayes Document Classifier

Jeevan Vankayala
jeevan4@unm.edu

Project Task:

In this project I implemented Naive Bayes technique to classify a document and applied it to the given classic newsgroups. Goal is to write a program that can predict which newsgroup a given document was posted to. The project has been implemented in Python using Numpy.

High-Level description of the code:

Code has been implemented in **Python 3.4.3 64** bit version using **numpy** for matrix representations and **matplotlib** lib. MLA,MAP and Classifications are calculated based on the formulae posted in Piazza.

mla_calc() :
This function calculates the likelihood probabilities for each label in the given set of documents. *label_dict* maintains the dictionary of number of documents in that particular news group with label number as key and *label_prob* holds the probability of the newsgroup over the given documents respectively.

map_calc(beta_arg) :
The function takes an argument for β value and calculates the probability of occurrence of each word for a given label. Initializes them into *map_matrix* (20 X 61188) and returns accuracy from the function *classify().* Here, the default β value used is 1/|V| and $\alpha = 1+\beta$.

classify() :
The function applies the log2 to *label_prob, map_matrix* and classifies the test data based on the argmax of the classification formula given in Piazza. The values are stored in *argmax_matrix* (7505 X 1). Then, the program compares the *argmax_matrix* with the provided test labels to obtain accuracy and confusion matrix (20 X 20).
The program calculates accuracy and confusion matrix for the default β which is 1/|V| and allows user to input a desired β value to recalculate accuracy and confusion matrix

vocab_rank():

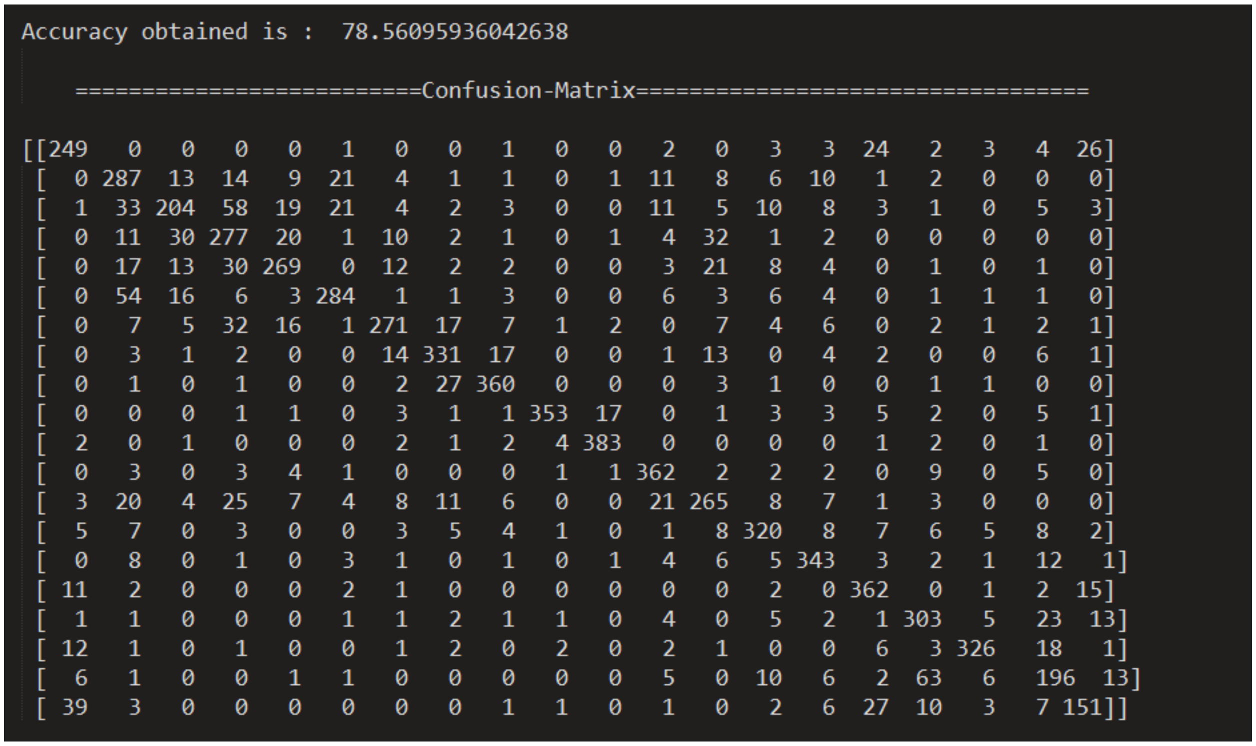
The function calculates uses the Bayes Theorem to calculate Posteriors based on Priors and Likelihood. Then, calculates the sum of probabilities for all labels given a word. By sorting these probabilities in ascending order we can give better rank to low frequency words.

Question 1: In your answer sheet, explain in a sentence or two why it would be difficult to accurately estimate the parameters of this model on a reasonable set of documents (e.g.1000 documents, each 1000 words long, where each word comes from a 50,000 word vocabulary)?

Ans: From the assumption, we know that the value of random variable Xi is the word found in position i of that document. That is the probability of finding a particular word in ith position is the key factor. We can't classify a document considering the position probability of a particular word because it is possible to paraphrase a sentence without affecting its own meaning. Therefore it will be difficult to accurately estimate the parameters for this mode. Also, the given sample of 1000 documents and 1000 words would be too small to sample a vocabulary of 50,000 words.

Question 2: In your answer sheet, report your overall testing accuracy (Number of correctly classified documents in the test set over the total number of test documents), and print out the confusion matrix (the matrix C, where cij is the number of times a document with ground truth category j was classified as category i)?

Ans: The overall testing accuracy achieved is **78.561 %.** Please find below the screen shot for confusion matrix.



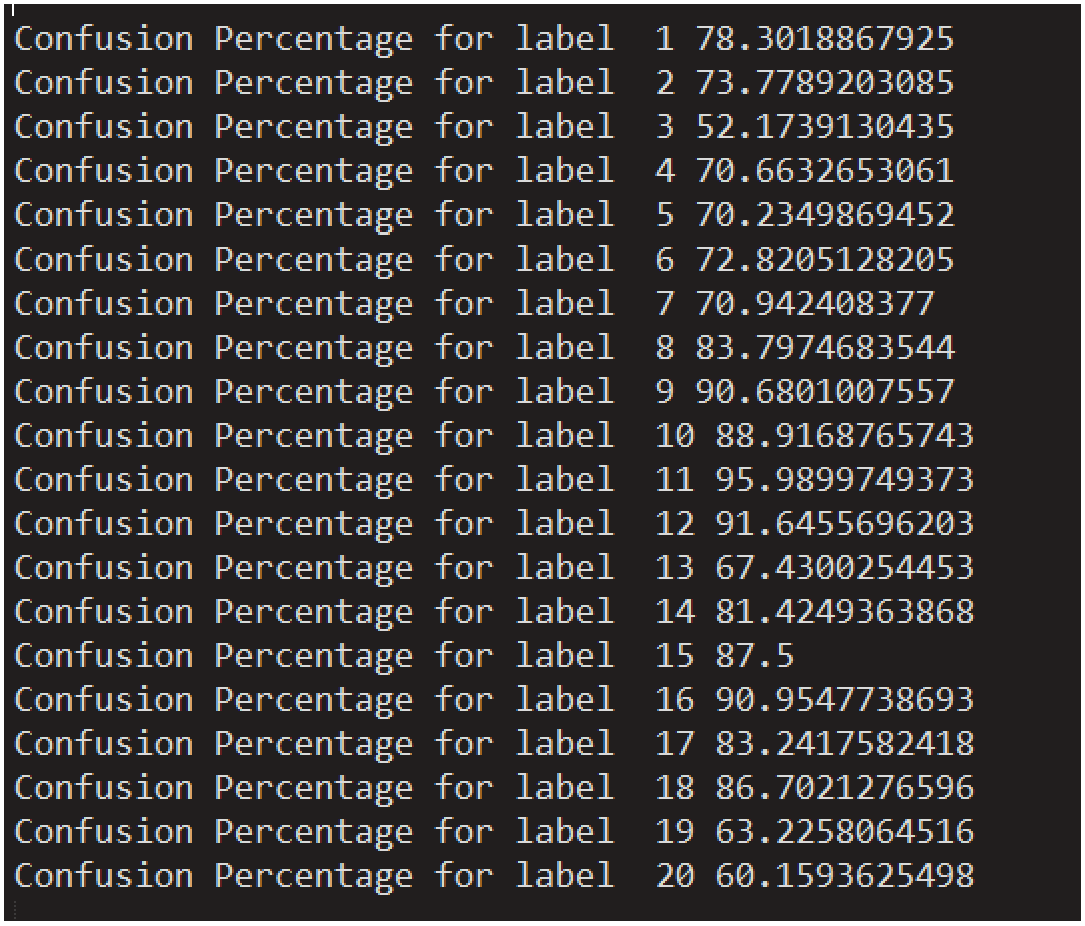
```
Accuracy obtained is :  78.56095936042638

    =========================Confusion-Matrix==================================

[[249   0   0   0   0   1   0   0   1   0   0   2   0   3   3  24   2   3   4  26]
 [  0 287  13  14   9  21   4   1   1   0   1  11   8   6  10   1   2   0   0   0]
 [  1  33 204  58  19  21   4   2   3   0   0  11   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   3 284   1   1   3   0   0   6   3   6   4   0   1   1   1   0]
 [  0   7   5  32  16   1 271  17   7   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   3   1   1 353  17   0   1   3   3   5   2   0   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  21 265   8   7   1   3   0   0   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   0 362   0   1   2  15]
 [  1   1   0   0   0   1   1   2   1   1   0   4   0   5   2   1 303   5   23  13]
 [ 12   1   0   1   0   0   1   2   0   2   0   2   1   0   0   6   3 326   18   1]
 [  6   1   0   0   1   1   0   0   0   0   0   5   0  10   6   2  63   6   196  13]
 [ 39   3   0   0   0   0   0   0   1   1   0   1   0   2   6  27  10   3   7 151]]
```

Question 3: Are there any newsgroups that the algorithm confuses more often than others? Why do you think this is?

Ans: From the confusion matrix above, we can say that the newsgroup belonging to similar categories are confused more often. Please find below the screenshot to observe confusion percentages for each newsgroup.

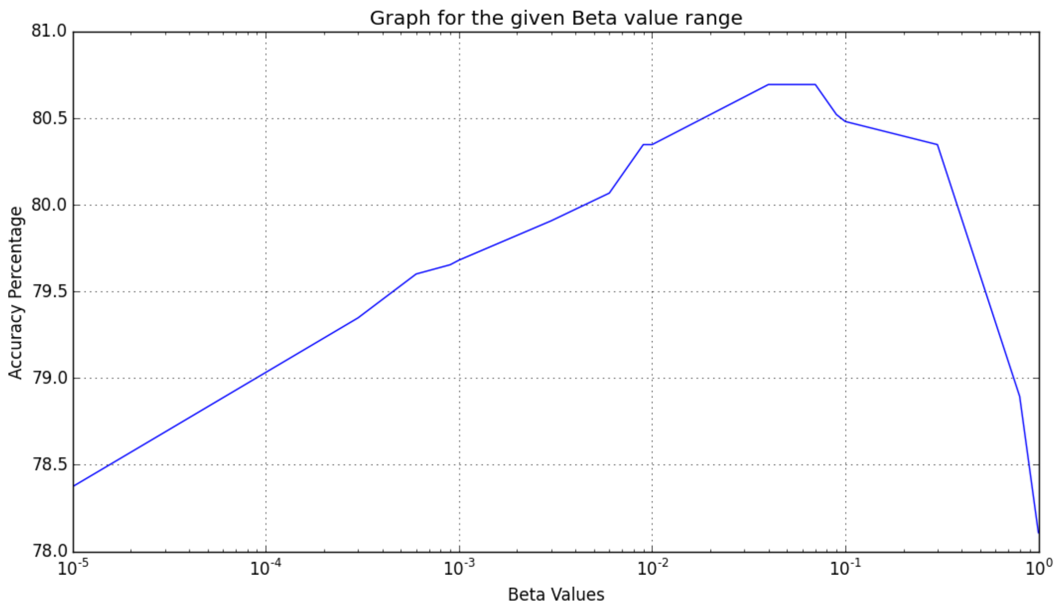


```
Confusion Percentage for label  1 78.3018867925
Confusion Percentage for label  2 73.7789203085
Confusion Percentage for label  3 52.1739130435
Confusion Percentage for label  4 70.6632653061
Confusion Percentage for label  5 70.2349869452
Confusion Percentage for label  6 72.8205128205
Confusion Percentage for label  7 70.942408377
Confusion Percentage for label  8 83.7974683544
Confusion Percentage for label  9 90.6801007557
Confusion Percentage for label  10 88.9168765743
Confusion Percentage for label  11 95.9899749373
Confusion Percentage for label  12 91.6455696203
Confusion Percentage for label  13 67.4300254453
Confusion Percentage for label  14 81.4249363868
Confusion Percentage for label  15 87.5
Confusion Percentage for label  16 90.9547738693
Confusion Percentage for label  17 83.2417582418
Confusion Percentage for label  18 86.7021276596
Confusion Percentage for label  19 63.2258064516
Confusion Percentage for label  20 60.1593625498
```

For example the algorithm confuses with 3, 4, 5 newsgroup's i.e. **comp.os.ms-windows.misc comp.sys.ibm.pc.hardware, comp.sys.mac.hardware** as they all belong to similar topic hardware. Same case with 19, 20 **i.e. talk.politics.misc, talk.religion.misc**

Question 4: Re-train your Naive Bayes classifier for values of β between .00001 and 1 and report the accuracy over the test set for each value of β. Create a plot with values of β on the x-axis and accuracy on the y-axis. Use a logarithmic scale for the x-axis (in Matlab, the semilogx command). Explain in a few sentences why accuracy drops for both small and large values of β?

Ans: Re-trained the classifier with the β values for the given range and below is the graph obtained.



Where Beta =
0.00001,0.0003,0.0006,0.0009,0.001,0.003,0.006,0.009,0.01,0.04,0.07,0.09,0.1,0.3,0.8,1

As β value tends to 0, the probabilities of rare words tends to dominate and this leads to mis-classification. If β value tends to 1, then the topic contains a mixture of most of the given words and again leads to mis-classification. Thus, β can only be estimated domain specific or based on statistical analysis. (http://stats.stackexchange.com/questions/37405/natural-interpretation-for-lda-hyperparameters)

Question 5: Propose a method for ranking the words in the dataset based on how much the classifier 'relies on' them when performing its classification (hint: information theory will help). Your metric should use only the classifier's estimates of P (Y) and P (X|Y). It should give high scores to those words that appear frequently in one or a few of the newsgroups but not in other ones. Words that are used frequently in general English ('the', 'of', etc.) should have lower scores, as well as words that only appear extremely rarely throughout the whole dataset. Finally, in your method this should be an overall ranking for the words, not a per-category ranking?

Ans: From Bayes theorem, we know that Posteriors = (Likelihoods)*(Priors)
i.e. P (Label/Word) = P (word/Label) P (Label)
We have already calculated P (word/Label) and P (Label) from MAP and MLE formula. So I would calculate P (Label/Word) which gives me a matrix of 20 X 61188. Now I would calculate the sum of probabilities for a word "w" for all the 20 labels. I will give rank for the ascending order for these sorted summations so that the high frequency words which have high probability

distribution will be getting the highest rank and the low frequency words that actually involve in judging a documents category will be getting best rank.

Question 6: Implement your method, set β back to 1/|V|, and print out the 100 words with the highest measure?
Ans:

```
['stimulte', 'lactoferrin', 'mahoney', 'cdromxa', 'rrgu', 'keytone', 'spokespersons', 'apron', 'tonja', 'spacers', 'imt', 'qiyong', 'swimsuit', 'cntf', 'zaf', 'masculinity', 'sensitizing', 'moyman', 'mfg', 'conserving', 'taint', 'frolv', 'draughn', 'xof', 'simultanously', 'circumspectly', 'aodc', 'hermetically', 'oobe', 'toddi', 'baysian', 'fugu', 'thyroidal', 'jamaica', 'pathologists', 'llpost', 'turcs', 'slabodkin', 'hnbak', 'understaffing', 'koitai', 'bueller', 'bruder', 'idh', 'narcosis', 'agens', 'strolls', 'graven', 'notkestr', 'wanderers', 'sutters', 'pateretur', 'suecide', 'jlredd', 'bensalem', 'siap', 'roskilde', 'azerineft', 'bagman', 'eitan', 'diques', 'nemonics', 'photoetching', 'sinan', 'indentured', 'unipd', 'hillig', 'seryozha', 'outsmarting', 'carboxylic', 'cdware', 'xgl', 'havetiff', 'vanishes', 'atmbree', 'advices', 'scarry', 'subpixel', 'yannis', 'indiavidual', 'wbx', 'sustenance', 'ethinic', 'knickknack', 'lasergraphics', 'chronis', 'erao', 'sib', 'jhpark', 'bruising', 'urodynamic', 'chafing', 'rewritable', 'hoogvorst', 'badger', 'chronologically', 'basra', 'intangibles', 'thereunder', 'mnv']
```

Question 7: If the points in the training dataset were not sampled independently at random from the same distribution of data we plan to classify in the future, we might call that training set biased. Dataset bias is a problem because the performance of a classifier on a biased dataset will not accurately reflect its future performance in the real world. Look again at the words your classifier is 'relying on'. Do you see any signs of dataset bias?
Ans: I can see a data set bias for some of the above words as there are many words which does not make much sense as they might belong to different language. We may not rely on these words in future as , most of the documents and articles are mainly concentrated in English.