**CS529 Introduction to Machine Learning**

**(Submitted by):**

**Nalluri, Varun Srinivas Chakravarthy (101671248)**

**Vattikonda, Gayathri Lakshmi Chowdary (101674556)**

# Project Report
# On
# Naïve Bayes Document Classifier

## 1. Introduction

*Scope:* This report documents the implementation details of Naïve Bayes document classifier on the given 20 newsgroups dataset provides results (accuracies) and gives our explanations for the questions 1 through 7.

*Organization of the Report:*

Section 1: *Introduction* – It defines the scope and organization of the report.

Section 2: *Explanation on code/working model* – Naïve Bayes and Formulae (statistics) used and high level explanation of how the code works.

Section 3: *Accuracies obtained* – Results obtained.

Section 4: *Questions 1 through 7* – Answers/Explanations for questions 1 through 7(Results, accuracy, which option works well and why?).

Section 5: *Conclusion* – What we want to say out loud.

## 2. Explanation on code/working model

### Naïve Bayes and Formulae (statistics):

Consider document D has d $\{X_1,\ldots\ldots\ldots,X_d\}$words, the value of the random variable $X_i$ is the word found in position i in the document.

To predict the label Y of the document (one of m categories):

$$P(Y|X_1 \ldots X_d) \propto P(X_1 \ldots X_d|Y)\, P(Y) = P(Y) \prod P(X_i|Y)$$

*Assumptions:*

i) Each $X_i$ is sampled from some distribution that depends on its position $X_i$ and the document category Y.

ii) With discrete data, assume that $P(X_i|Y)$ is a multinomial distribution over some vocabulary V => each $X_i$ can take one of |V| possible values corresponding to the words in the vocabulary.

iii) Derives to an assumption that for any pair of document positions i and j, $P(X_i|Y)$ may be completely different from $P(X_j|Y)$.

*Additional assumption:*

iv) $\forall i, j\ P(X_i|Y) = p(X_j|Y)$
In addition to estimating P (Y), also estimate the parameters for the single distribution P (X|Y), which is equal to be P ($X_i$|Y) for all Xi, each word in a document is assumed to be an iid draw from this distribution.

*Dataset:* Dataset consists of six files vocabulary.txt, newsgrouplabels.txt, train.label, test.label, train.data and test.data.

i) vocabulary.txt is a list of the words that may appear in documents. The line number is word's id in other files.
ii) newsgrouplabels.txt is a list of newsgroups from which a document may have come.
iii) train.label in this each line corresponds to the label for one document from the training set again, the document's id (docId) is the line number.
iv) test.label is same as train.label, except that the labels are for the test documents.
v) train.data specifies the counts for each of the words used in each of the documents, each line is of the form "docId wordId count", where count specifies the number of times the word with id wordId in the training document with id docId. All word/document pairs that do not appear in the file have count 0.
vi) test.data same as train.data, except that it specifies counts for test documents.

We choose *Java* programming language to implement Naïve Bayes document classifier.

1. Class Train - This class implements all the methods required to load training data and creates training modal
    a. train) - This is the parent method for training. This method calls all the necessary methods required to load and create model in the order. Returns trained model.
    b. Calc_mle(data instance) – This method calculates MLE and stores all the data in Train data model.
    c. Load Cat Data(data instance)- This method reads category information, category of each document from input files and stores in a map for future use.
    d. Load Doc Data(data instance) – This method reads train data from input files and process the train data to get word counts for each category.
2. Class Test - This class implements all the methods required to load test data and to calculate accuracy
    a. test(data instance) – This is the parent method for testing. This method calls all the necessary methods required to load test data and predict the value for each test record and compute accuracy
    b. predict(test record) - This method predicts the outcome of given test values based on the passed training data. Thread pooling is used in-order to improve the performance.
    c. Calc Accuracy – This method calculates accuracy by comparing predicted values against actual values.
3. Class LoadTestCategory - This class implements methods required to load the test document actual categories. This will be done on independent thread.

    a. run(path) – When thread is created and started this will load the the test document labels.
4. Class Classifier - This class extends thread and contains method required for prediction.
    a. run(test rec) – When a new task prediction task to assigned to thread pool, one of the thread in the pool will be used to predict the outcome of the given test data. If all threads are occupied, tasks will be added to queue and will be processed in order as they were added.
5. Class Train Data - This class has all the variables required to store the training model and all methods required to access the data stored in the variables.
6. Class Utils - This class contains all the methods which are commonly used through out the program.
    a. Entropy(pos, neg) – This method calculated the entropy for given pos and neg values.
    b. info_gain(attribute values, total size) – This method calculates the information gain using entropy.
    c. Top_words (word counts) – This method assign ranks to vocabulary and return top 100 words list based on rank
    d. Sort Map By Value(HashMap) - This method sorts the map by value.
    e. Draw_graph(values) - This method plots the graph for beta vs accuracy using JFreeChart
    f. TokenizeStopStemming(String) – This method will remove stop words and perform stemming on the passed string.
7. Class constants – This is an interface which contains all the constants.
8. Class ValueComparator – This class contains functionality required to sort map based on value.
9. NaiveBayes – This is the entry point of the program and this class contains main method. This will accept the arguments and call the functions accordingly.

**3. Accuracies obtained**

The implemented algorithm has given 78.52099% accuracy

Obtained by -

- ⇨ *Accuracy = Number of correctly classified documents in the test set/Total number of test documents*
- ⇨ *Number of correctly classified documents in the test set = 5893*
- ⇨ *Total number of test documents = 7505*
- ⇨ 5893/7505 = 0.7852099
- ⇨ 78.52099 %

**4. Questions 1 through 7**

***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).

***Explanation:*** In this model we need to find the probability of a word in $i^{th}$ position in a document, As we are aware that a position of a word is not so important as we can reframe the sentence with the same essence but changing in the position of words. Position of the word need not provide the context of the word unlike sentiment analysis where context of word is retained, So there is no use of giving importance to position of word. . Also, computing this will be time and space consuming. This makes us strongly believe that it's difficult to accurately estimate the parameters of this model on a reasonable set of documents.

***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).

***Explanation:*** *Overall testing accuracy* – 78.52099 %

Obtained by -

- ⇨ *Accuracy = Number of correctly classified documents in the test set/Total number of test documents*
- ⇨ *Number of correctly classified documents in the test set = 5893*
- ⇨ *Total number of test documents = 7505*
- ⇨ 5893/7505 = 0.7852099
- ⇨ 78.52099 %

*Confusion matrix:*

```
gayathri@brown:~/HW02$ java -jar NaiveBayes.jar
Accuracy obtained by:
Number of correctly classified documents in the test set: 5893
Total Number of test documents: 7505
Accuracy : 78.52099%.

************Confusion Matrix************
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      264     9       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
```

Figure 1: Results obtained (Confusion Matrix and Accuracy obtained)

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

***Explanation:*** Yes, there are newsgroups that the algorithm confuses often when compared to others. We have observed that this is with respect to similar/related news groups.
Examples: {3,4,5,6} Newsgroups: 3-comp.os.ms-windows.misc, 4 - comp.sys.ibm.pc.hardware, 5- comp.sys.mac.hardware, 6-comp.windows.x

***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.

***Explanation:*** After re-training Naive Bayes classifier for values of β between .00001 and 1 the accuracy over the test set for each value of β ranged from 78% to 81%.

| Beta β | Accuracy % |
|---|---|
| 0.000005 | 78.33444 |
| 0.00001 | 78.38774 |
| 0.00005 | 78.70753 |
| 0.0001 | 78.89407 |
| 0.0005 | 79.54697 |
| 0.001 | 79.70686 |
| 0.005 | 80.0533 |
| 0.01 | 80.373085 |
| 0.05 | 80.75949 |
| 0.1 | 80.466354 |
| 0.5 | 79.86675 |
| 1 | 78.107925 |

Table 1: Beta β values and corresponding Accuracies obtained respectively

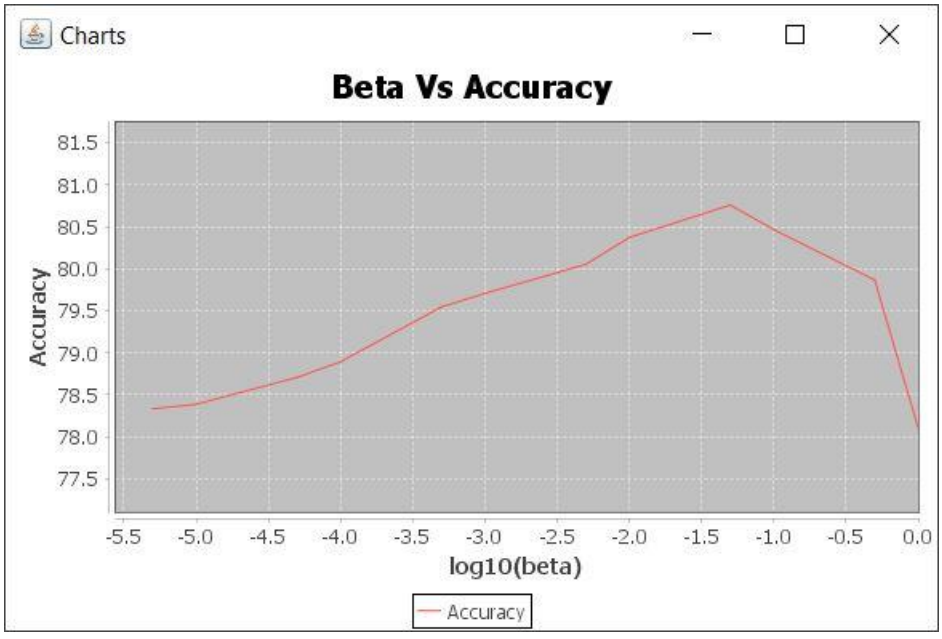


Figure 2: Beta and Accuracy plot (with values of β on the x-axis and accuracy on the y-axis)

Which option works well and why?

The accuracy is higher when compared to the initial accuracy we have obtained and dropped. It's because for lower beta values model over fits the data which leads to drop in accuracy, when beta tends to 1 model under fits the data as beta=1/|V| when Beta =1 => V=1, cannot calculate accuracy with one word, so the accuracy drops.

***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.

***Explanation:***

Proposed method: We have removed the stop words and used Entropy and information gain of each word to assign rank. The Entropy of a category given a document with a single word will be low if a word appears most of the times in a single class, when we draw a distribution of the word among all the categories the word will have a peak distribution in that particular category where it is seen most of the times. So the word with the higher Information gain will most likely points to fewer categories. And to find there measures we propose MAP Maximum a Posteriori and MLE Maximum Likelihood Estimation probabilities (P (Y) and P (X|Y)) to find the ranks.

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

***Explanation:*** When β is set back to 1/|V|, the 100 words with the highest measure are as follows:

[tgk, quelled, quintessential, philanthropy, supernaturalists, shortsightedness, innuendo, heaping, tinnsg, yohan, causation, loook, gullibility, manipulators, axiomatic, antirational, formalise, umbrellas, cartons, bonton, cheddar, sociopaths, gobbledygook, zakaria, gibreel, storyteller, postman, yaar, butbutbut, uninteresting, recanting, baal, givin, kempf, graven, holies, hiatus, tepper, bobbys, vegetarianism, schaertel, seduce, kurtz, contradition, obfuscationist, religionist, obfuscationism, subjectivism, evaluative, irrepective, noterrorism, terrorisim, indentified, bakes, tugs, gad, vindication, innknk, delhi, eov, zealotry, persue, poohed, lavey, sunscreen, towerl, veggie, hotdogs, elekta, creepeth, dgsid, assension, luk, crucifiction, bowed, salome, seeth, marys, opon, beheld, steadfastly, falicy, rebutal, qkovl, helpfully, mander, iroquois, classless, noncoercive, fairman, cfairman, aroun, puritanical, moroni, reunited, qugin, qkogg, curving, eaplain, staceys]

***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? Explain.

***Explanation:*** Yes, we have observed the biasedness of the dataset. We say his because some of the words. As 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. The data set and the classification that we obtained may not be the correct at some point in future and contradict our results as the data set could be time dependent, event dependent and many more like this.

## 5. Conclusion

The results obtained from the implementation of the Naïve Bayes document classifier is to predict a document belongs to which newsgroup. But with only 78.52099% accuracy. Naïve Bayes however is asked to implement for this project we have obtained this accuracy. Also, there are other classifiers that can give us better results.

## References


1. http://www.cs.waikato.ac.nz/~eibe/pubs/FrankAndBouckaertPKDD06new.pdf
2. http://www.jmlr.org/papers/volume3/forman03a/forman03a_full.pdf
3. https://piazza.com/class/ijekobyz30xzd?cid=86
4. http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf
5. https://lucene.apache.org/core/
6. JFree chart to plot Beta and Accuracy