Naïve Bayes Classifier

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set.

I have used Java SE 1.7 to develop the application.

High-level description of the code:

1) The application is run with the following command in Command Prompt/Terminal
   java -jar NaiveBayes.jar [beta_value]
   beta_value - value varies from 0.00001 to 1
   if no "beta_value" is given, 1/|V| is considered.
   Note: Conventions followed in the application:
   a. Xi - WordId
   b. Yk - Category/Class
2) Get the train.label contents in the matrix [DocId, Yk]
   - Store this in a HashMap (trainLabelsMap)
3) Get the test.label contents in the matrix [DocId, Yk]
   - Store this in a HashMap (testLabelsMap)
4) Calculate Yk counts into the matrix [Yk, Count] using trainsLabels.
   - Store this in a HashMap (trainLabelsCount)
5) Calculate Priors using P(Yk) = (# of docs labeled Yk)/(total # of docs) using trainLabelsCount.
   - Store this in a HashMap (pYk_trainLabelsMle)
6) Load test.data file into the application in the form of matrix (docId, wordId, count)
   - Store this in a HashMap (testDataMap)
7) Count of Xi in Yk - Calculate and load Count of all the WordIds in different Categories using trainLabelsMap, trainDataMap.
   - Store this in a HashMap (countOfXiInYk).
8) Calculate MAP for P(X|Y) and load these values into the HashMap (pXiYk_trainMap) using countOfXiInYk, trainLabelsCount, beta_value, vocabularyCount.
   - P(Xi|Yk)=(count of Xi in Yk)+(beta)(total words inYk)+((beta)*(length of vocab list)))
   - Store this in a HashMap (pXiYk_trainMap)
9) Calculate the classifications of all the documents using pXiYk_trainMap, testDataMap, pYk_trainLabelsMle.
   - Use Ynew=argmax[log2(P(Yk))+(sum over i)(# of Xnewi)log2(P(Xi|Yk))] to classify a document.
   - Store this in a HashMap (pYkDocId_testMap)
10) Declare Confusion Matrix and initialize all the values with zero.
11) Calculate correctly classified labels and populate Confusion Matrix by comparing the values in testLabelsMap and pYkDocId_testMap.
12) Calculate accuracy using (Correctly Classified Labels/Total Labels)*100.
13) Print the top 100 words with highest measure.

Accuracies obtained under various conditions:

Beta value	Accuracy
0.000001	70.60626249
1.63E-05	69.51365756
0.00001	69.74017322
0.00005	68.80746169
0.0001	68.34110593
0.0005	67.18187875
0.001	66.46235843
0.005	64.62358428
0.01	63.71752165
0.05	59.53364424
0.1	57.01532312
0.15	55.09660227
0.2	53.85742838
0.25	52.93804131
0.3	51.88540973
0.35	51.005996
0.4	50.19320453
0.45	49.64690207
0.5	48.95403065
0.55	48.32778148
0.6	47.75483011
0.65	47.18187875
0.7	46.79546969
0.75	46.46235843
0.8	45.99600266
0.85	45.5562958
0.9	45.20986009
0.95	44.71685543

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).

- X is a sequence of words in a document
- X (and hence P(X|Y)) is huge!
    - For Eg: A document with at least 1000 words, $X = \{X_1, X_2, \ldots\ldots, X_{1000}\}$

- X_i represents i^{th} word in document i.e., the domain of X_i is entire vocabulary which may contain 10000 words or even more.
- Therefore number of calculations is $10000^{1000} = 10^{4000}$
- This is why it is difficult to accurately estimate the parameters of this model.

- In Naive Bayes, we make the additional assumption –
  - Position in document doesn't matter:
    - $P(X_i = x_i \mid Y=y) = P(X_k = x_i \mid Y=y)$
      This means that all positions have the same distribution
  - Ignore the order of words

Question 2: In your answer sheet, report your overall testing accuracy (Nubmer 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).

- Beta value : 1.6343073805321303E-5 (1.0/vocabularyCount)
- Accuracy of classification of test labels: 69.51365756162559%

	alt.atheism	comp.graphics	comp.os.ms-windows.misc	comp.sys.ibm.pc.hardware	comp.sys.mac.hardware	comp.windows.x	misc.forsale	rec.autos	rec.motorcycles	rec.sport.baseball	rec.sport.hockey	sci.crypt	sci.electronics	sci.med	sci.space	soc.religion.christian	talk.politics.guns	talk.politics.mideast	talk.politics.misc	talk.religion.misc
alt.atheism	226	0	1	0	2	0	0	0	0	0	0	0	4	5	0	5	0	2	2	32
comp.graphics	0	215	22	11	19	19	18	1	1	1	0	2	8	4	5	1	0	0	0	0
comp.os.ms-windows.misc	0	4	123	19	2	0	4	1	0	0	0	0	0	0	0	0	0	0	0	0
comp.sys.ibm.pc.hardware	0	7	41	246	24	3	47	0	1	1	0	1	8	1	1	0	0	0	0	0
comp.sys.mac.hardware	0	2	6	11	188	0	14	0	0	0	0	0	1	0	0	0	0	0	0	0
comp.windows.x	1	62	82	9	15	319	7	0	0	1	0	1	5	0	4	0	1	0	1	0
misc.forsale	0	0	0	3	2	0	136	2	0	0	0	0	2	0	0	0	1	0	0	0
rec.autos	0	0	1	0	2	0	14	301	22	0	0	0	5	0	0	0	0	0	0	0
rec.motorcycles	0	0	0	1	1	0	4	6	309	1	0	0	3	0	0	0	0	0	0	0
rec.sport.baseball	0	0	0	0	0	0	2	0	0	282	1	0	0	0	0	0	0	1	0	0
rec.sport.hockey	0	2	0	1	0	0	3	0	0	23	373	0	0	0	1	0	1	0	0	0
sci.crypt	2	45	55	34	49	28	18	6	5	4	1	364	85	4	2	0	7	1	6	3
sci.electronics	0	3	2	28	8	0	15	3	1	0	0	0	202	3	3	0	0	0	0	0
sci.med	2	11	14	3	13	5	13	3	2	1	0	1	22	288	4	2	3	0	3	1
sci.space	3	10	8	7	6	3	16	5	1	3	0	2	17	5	319	0	1	0	4	5
soc.religion.christian	31	3	3	3	1	0	4	3	0	6	0	0	3	14	1	354	1	2	2	33
talk.politics.guns	0	1	5	0	7	1	9	9	7	5	3	7	7	6	4	0	258	1	37	9
talk.politics.mideast	18	7	6	6	13	6	23	13	19	13	10	5	13	24	15	12	22	354	29	11
talk.politics.misc	11	17	21	9	31	6	33	42	28	56	11	12	8	38	33	14	58	15	221	18
talk.religion.misc	24	0	1	1	0	0	2	0	1	0	0	0	0	1	0	10	11	0	5	139

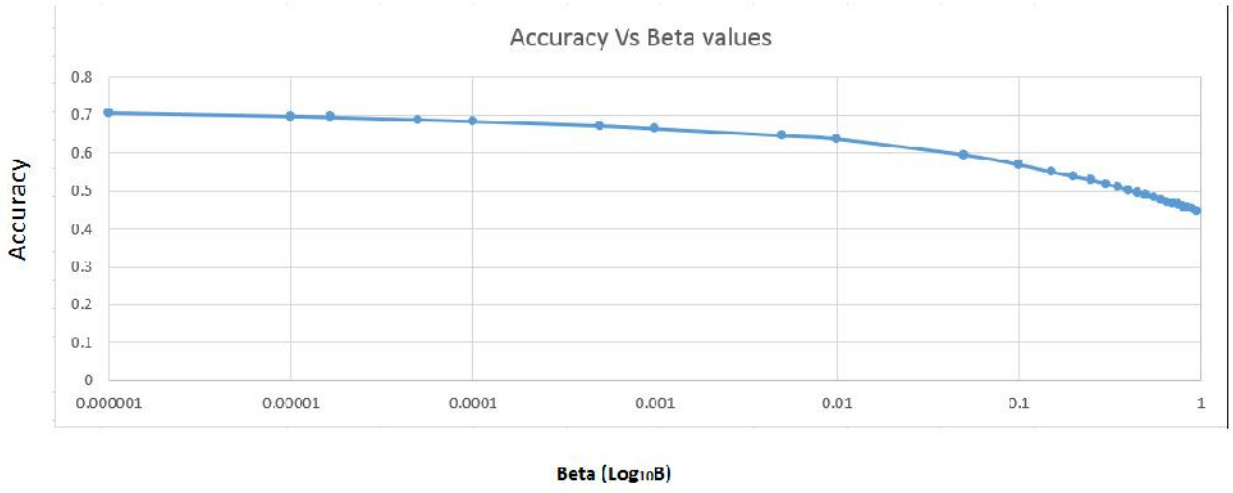
Question 3: Are there any newsgroups that the algorithm confuses more often than others? Why do you think this is?

comp.windows.x comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware sci.crypt
talk.politics.guns talk.politics.mideast talk.politics.misc
soc.religion.christian talk.religion.misc alt.atheism

From the confusion matrix, I have found that the above news groups which have similar topics have confused notably.

This may be because of the documents having similar kind of words.

Question 4: Re-train your Naive Bayes classifier for values of α 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



In my implementation, for the beta values from 0.000001 to 1, the accuracy tends to decrease gradually and reaches a particular point in between 40-50%. For large values of β, the likelihood estimates are almost equal to prior. The rare words in different categories are almost negligible which is why the accuracies drop when β tends to 1.

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 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.

I have used priors P(Y) and P(X|Y) for all the words over 20 different categories. For each word I have calculated the measures using the likelihood estimates for all the words irrespective of the categories. I stored the results in a HashMap in the application. The HashMap consists of the data in the form of [WordId, Measure]. I sorted the HashMap based on measure which gives the ranking of the measures of different words.

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

I have considered the top 100 words with $P(X_i|Y_k)$ from the matrix [Wordid, Measure] which have the highest measure. The following are the words which I got from the application:

'the', 'of', 'to', 'and', 'in', 'that', 'is', 'it', 'you', 'they', 'was', 'for', 'not', 'be', 'are', 'on', 'this', 'have', 'we', 'as', 'were', 'by', 'god', 'from', 'with', 'but', 'he', 'there', 'space', 'or', 'key', 'their', 'if', 'can', 'will', 'all', 'people', 'who', 'at', 'what', 'windows', 'an', 'no', 'israel', 'gun', 'would', 'one', 'about', 'edu', 'had', 'do', 'my', 'car', 'them', 'has', 'writes', 'window', 'so', 'me', 'your', 'his', 'encryption', 'turkish', 'com', 'article', 'armenian', 'when', 'more', 'scsi', 'drive', 'team', 'out', 'don', 'jews', 'israeli', 'said', 'think', 'armenians', 'file', 'jesus', 'president', 'game', 'nasa', 'mr', 'hockey', 'chip', 'been', 'just', 'image', 'some', 'government', 'db', 'which', 'any', 'use', 'graphics', 'clipper', 'bike', 'up', 'only'

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?

Looking at the accuracies I got in the application, with high β values i.e., low vocabulary count - have the less accuracies of the classification. This is because, as the vocabulary count is low, the training dataset may not be sampled independently from the same distribution of data we plan to classify in the future. This training set may be biased. But for low β values i.e., high vocabulary count, the situation is vice versa. The document classification may also depend on the words present in that document. The words may be present at that point of time when the document is generated, but the same words may not repeat in the future. So, our data set should be consistent over the time which produces higher accuracy results.