**ABSTRACT**

Sentimental analysis is a process of determining the opinion or feeling of a piece of text.The texts from reviews or Blogs is processed to get an accurate description of how the writer feels regarding the subject. The problem in sentimental analysis is classifying the polarity of a given text at the document,sentence ,or feature/aspect level,Whether the expressed opinion in a document , a sentence or an entity feature or aspect is positive,negative, or neutral .It is one of the major tasks of NLP (Natural Language Processing). Sentiment analysis has gain much attention in recent years .we aim to tackle the problem of sentiment polarity categorization, which is one of the fundamental problems of sentiment analysis. A general process for sentiment polarity categorization is proposed with detailed process descriptions.

**Keywords**

Sentiment analysis; Sentiment polarity categorization; Natural language processing; Product reviews; DocumentTermMatrix; VectorSource; Corpus.

**1.TECHNOLOGY OPTED**

**1.1 R programming Language**

In R there is more than 6000 packages for performing different analysis operations I am using DocumentTermMatrix which describes the frequency of distinct terms that occur in a collection of documents. I am not using any kind of algorithms because I don’t have any labelled data to train the algorithms, I am using a list of positive words and negative words to classify whether a given review or comment is positive or negative. Posterior probability concept is using with the help of Naïve Bayesian algorithm.

**1.2**.**Motivation**

we generally perform analysis on labelled data and generate some classififcation rules, basing on the classififcation rules we determine the nature of feedback. Just think, what if we don’t have any labelled data, then how to perform analysis on data without any learning?

There is a way!

we can perform sentimental analysis on feedback or reviews by " sentimental analysis with bag of words" without any learning from labelled data

**2.TECHNOLOGY RELEVANCE**

R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, etc) and graphical techniques, and is highly extensible. One of R’s strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control. Most of the industries are using R to access their company insights and R is open source too, so I was opted R to do my project.

**3**. **IMPLEMENTATION**

**3.1.IMPLEMENTATION OF “BAG OF WORDS MODEL”**

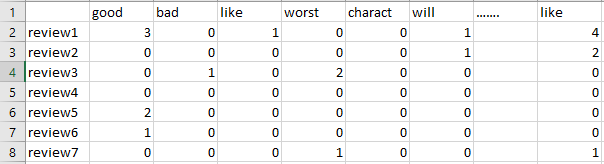
* There are two phases in this model
* **PHASE I**: understanding the basic nature of data
* **PHASE II**: counting number of positive words and negative words in the reviews consecutively.
* Displaying total number of positive, negative and neutral reviews
* among the whole data

**3.1.1 PHASE I**

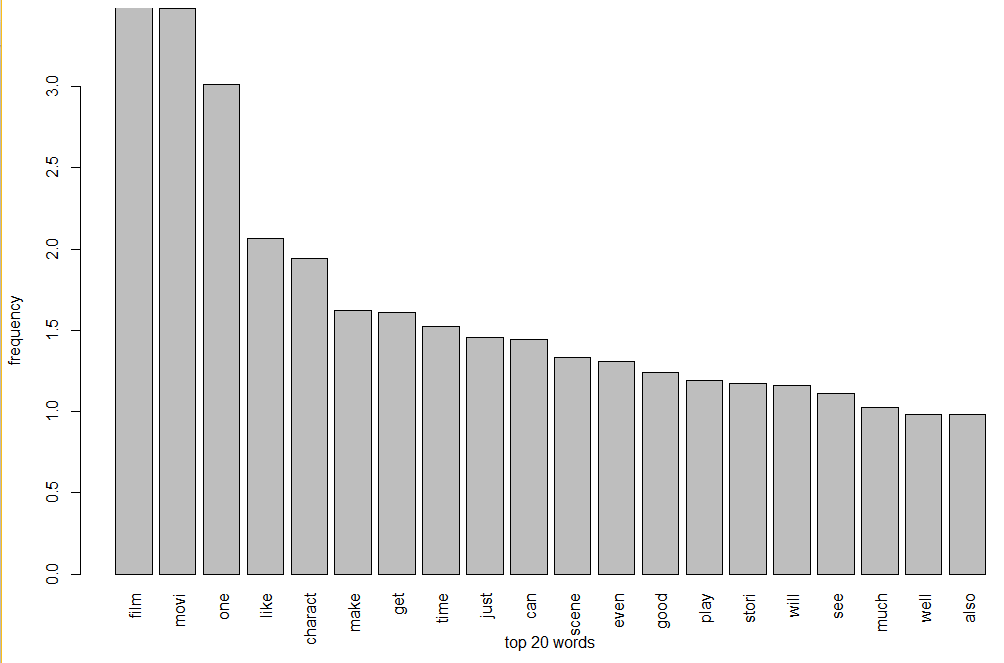
1. load the data on which you want to perform sentimental analysis
2. create a corpus of the data, which you loaded
3. Now clean the corpus i.e., punctuations, numbers, remove stopwords and the noisy data which does not help you in making decisions
4. now convert the document as a stem document
5. make a document term matrix of stemmed data
6. convert the DTM into matrix
7. sort the columns of document term matrix from highest to lowest mean order
8. display the 20 top frequency words in bagplot
9. create a word cloud of top 100 frequency words

**3.1.1 .PHASE I IMPLEMENTATION**

* Required libraries
* library(tm) # text mining
* library(wordcloud) #to genearate wordclud
* library(SnowballC) #word streaming
* library(caret) # partitioning,plots,etc.,
* setwd("C:/Users/ghari/Desktop/newattempt")
* movie<-read.csv("movie.csv",header = T,stringsAsFactors = F) # reding the file
* moviec<-Corpus(VectorSource(movie$review)) #converting it into corpus
* length(moviec)
* moviec$review[1]
* cleaning the DATA
* doc<-tm\_map(moviec,removeNumbers)
* doc<-tm\_map(doc,removeWords,stopwords("english"))
* doc<-tm\_map(doc,removePunctuation)
* doc<-tm\_map(doc,stripWhitespace)
* doc<-tm\_map(doc,stemDocument,language="english")
* moviedtm<-DocumentTermMatrix(doc)
* **## A document matrix create table with distinct words in colums**
* A document term will look like this

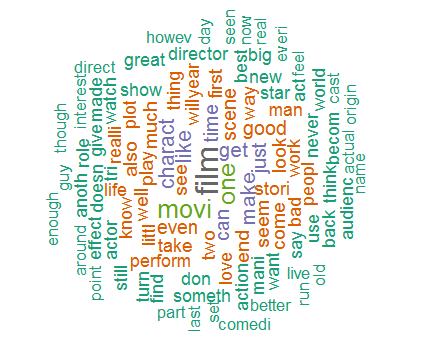


* A 2000 Review data set document term have a dimension of (2000,25000)
* The DTM describes the frequency of distinct terms that occur in a collection of documents.
* rr<-removeSparseTerms(moviedtm,0.8) # removing low frequency terms
* mean\_movie<-sort(colMeans(as.matrix(rr),na.rm = TRUE),decreasing = T)
* mean\_movie[1:20] ## top 20 frequency words
* **### barplot**
* barplot(mean\_movie[1:20],xlab="top 20 words",ylab="frequency",las=3,ylim=c(0,3))
* **## wordcloud**
* wordcloud(names(mean\_movie[1:100]), mean\_movie[1:100], min.freq = 0,scale=c(2,1) ,max.words=200, random.order=FALSE, rot.per=0.80, colors=brewer.pal(8, "Dark2"))



Figno.1.1

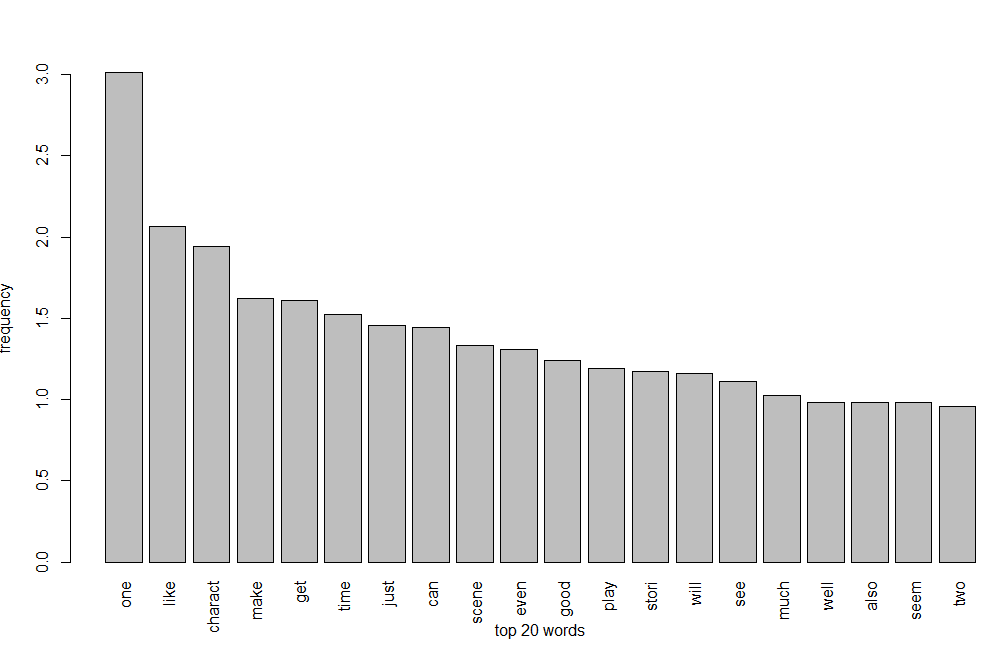
**Figno.1.1:** This figure Show a Barplot of top 20 Freqent Words



Figno.1.2

Figno.1.2: This figure Show a WordCloud of top 100 Freqent Words

* As we can see , the word cloud consists of some words which will not help us in making decision so removing that stream of unnecessary words
* unwanted<-c("film","movie")
* unwanted<-tm\_map(tempp,removeWords,unwanted)
* unwanted<-tm\_map(unwanted,stemDocument,language="english")
* ## unwanted words must remove before stemming
* moviedtm<-DocumentTermMatrix(unwanted)
* mean\_movie<-sort(colMeans(as.matrix(rr),na.rm = TRUE),decreasing = T)
* **#baarplot**
* barplot(mean\_movie[1:20],xlab="top 20 words",ylab="frequency",las=3,ylim=c(0,3))
* **## wordcloud**
* wordcloud(names(mean\_movie[1:100]), mean\_movie[1:100], min.freq = 0,scale=c(2,1) ,max.words=210, random.order=FALSE, rot.per=0.80,colors=brewer.pal(8, "Dark2"))



Figno. 1.3

**Figno 1.3**: This figure Show a Barplot of top 20 Freqent Words after removing Unwanted from Data



Figno.1.4

**Figno.1.4**: This figure Show a Wordcloud of top 100 Freqent Words after removing Unwanted from Data

**3.1.2.PHASE II**

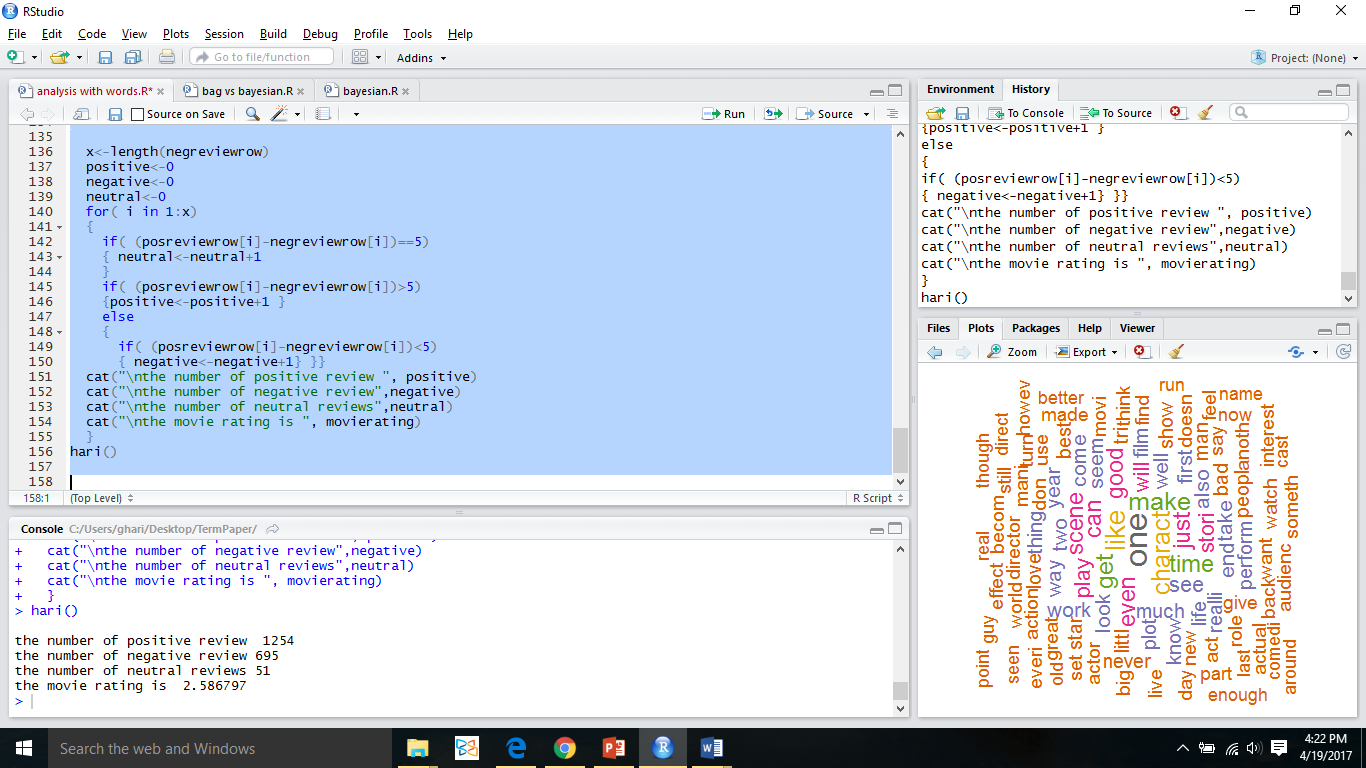
* load a list of stemmed positive words
* load a list of stemmed negative words
* load the data and clean it
* cleaning includes removing numbers, stemming, removing punctuation ,removing stop words,and the noisy data which is not helpful in making decisions
* create a positive document term matrix by passing postive word list as argument to DTM
* create a negative document term matrix by passing negative word list as argument to DTM
* convert the negative and positive DTM into matrix form
* if review[1] poswords - review[1] negwords >=1 count it as a positive review
* if review[1] poswords - review[1] negwords <=1 count it as a negative review
* if review[1] poswords - review[1] negwords ==0 count it as a NEUTRAL REVIEW
* **EXAMPLE**
* **REVIEW** :: The movie is the worst movie
* Poswords=0 negwords=1
* Poswords-negwords=-1 <0 so it is counted as a negative review

**PHASE II IMPLENATION**

* **#loading positive words**
* neg<-read.csv("neg.csv",header = F,stringsAsFactors = T)
* negc<-Corpus(VectorSource(neg))
* negc<-tm\_map(negc,stemDocument,language="english")
* negdtm<-DocumentTermMatrix(negc)
* negwords<-findFreqTerms(negdtm)
* **#loading negative words**
* pos<-read.csv("pos.csv",header = F,stringsAsFactors = T)
* posc<-Corpus(VectorSource(pos))
* posc<-tm\_map(posc,stemDocument,language="english")
* posdtm<-DocumentTermMatrix(posc)
* poswords<-findFreqTerms(posdtm)
* **## we are creating a DTM with postivie words**
* posdtm<-DocumentTermMatrix(doc, control=list(dictionary=poswords))
* posreviewrow<-rowSums(as.matrix(posdtm),na.rm = TRUE)
* **## we are creating a DTM with negative words**
* negdtm<-DocumentTermMatrix(doc, control=list(dictionary=negwords))
* negreviewrow<-rowSums(as.matrix(negdtm),na.rm = TRUE)
* **# my function**
* hari<-function()
* {
* **### movie rating**
* negrate<-colMeans( as.matrix(negdtm),na.rm = TRUE)
* d<-length(negrate)
* negt<-mean(negrate[1:1280])
* posrate<-colMeans( as.matrix(posdtm),na.rm = TRUE)
* e<-length(posrate)
* post<-mean(posrate[1:e])
* movierating<-post-negt
* x<-length(negreviewrow)
* positive<-0
* negative<-0
* neutral<-0
* for( i in 1:x)
* {
* if( (posreviewrow[i]-negreviewrow[i])==0)
* { neutral<-neutral+1
* }
* if( (posreviewrow[i]-negreviewrow[i])>0)
* {positive<-positive+1 }
* else
* {
* if( (posreviewrow[i]-negreviewrow[i])<0)
* { negative<-negative+1} }}
* cat("\nthe number of positive review ", positive)
* cat("\nthe number of negative review",negative)
* cat("\nthe number of neutral reviews",neutral)
* cat("\nthe movie rating is ", movierating)
* }
* hari()

**RESULT**

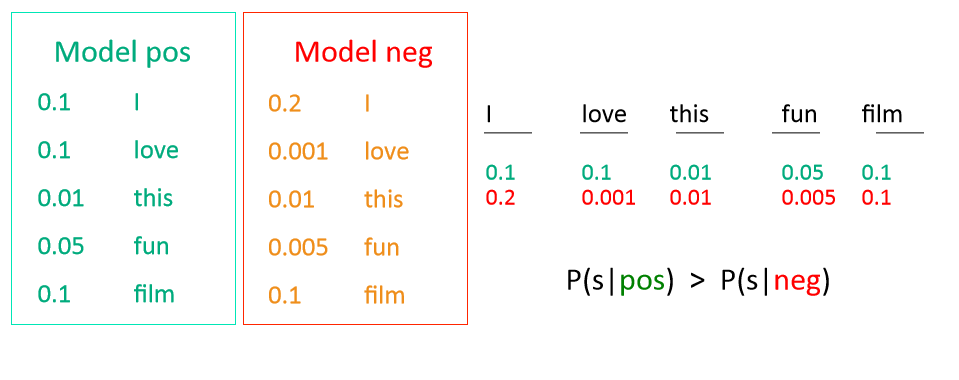
* Sentimental Analaysis with BAG of words achieved 69.5 percent of accuracy



**3.2.IMPLEMENTATION OF “NAÏVE BAYESIAN”**

**3.2.1 Naïve Bayes as a Language Model**

* For a document d and a class c
* P(c|d)= P(d|c)P(c) P(d)



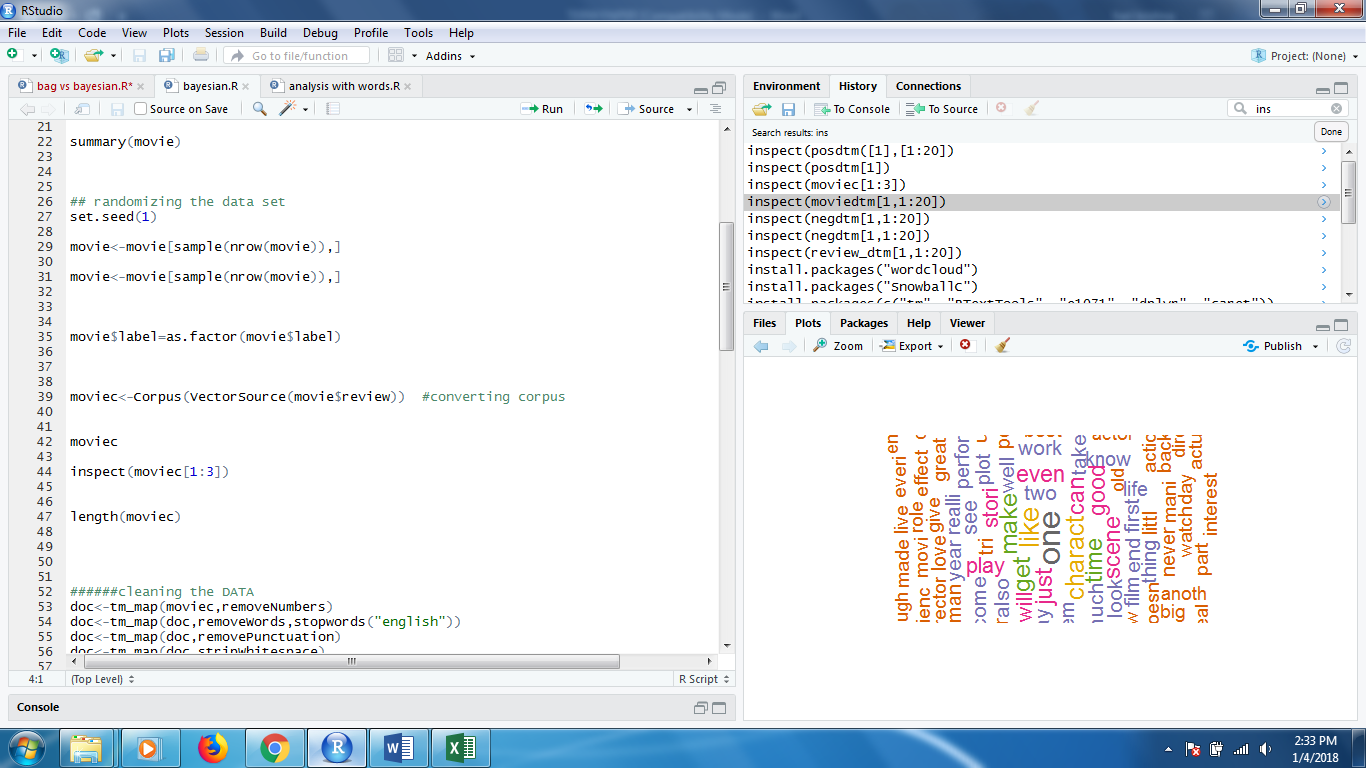
**3.2.2 IMPLEMENTATION STEPS**

1. load the data on which you want to perform sentimental analysis
2. create a corpus of the data, which you loaded
3. Now clean the corpus i.e., punctuations, numbers, remove stopwords and the noisy data which does not help you in making decisions
4. now convert the document as a stem document
5. make a document term matrix of stemmed data
6. Partition the data ,corpus, documentTermMatrix into 1:3 Ratio for training and testing purpose
7. Apply naïve Bayesian classifier on Training Data.
8. Apply resultant classifier from trained on Test Data
9. Find the Accuracy of Classifier using confusion Matrix

* **## libraries**
* library(tm)
* library(RTextTools)
* library(e1071)
* library(dplyr)
* library(caret)
* **## reading the file**
* setwd("C:/Users/ghari/Desktop/TermPaper")
* movie<-read.csv("movie.csv",header = T,stringsAsFactors = F)
* summary(movie)
* **## randomizing the data set**
* set.seed(1)
* movie<-movie[sample(nrow(movie)),]
* movie<-movie[sample(nrow(movie)),]
* movie$label=as.factor(movie$label)
* **##converting corpus**
* moviec<-Corpus(VectorSource(movie$review))
* moviec
* inspect(moviec[1:3])
* length(moviec)
* **######cleaning the DATA**
* doc<-tm\_map(moviec,removeNumbers)
* doc<-tm\_map(doc,removeWords,stopwords("english"))
* doc<-tm\_map(doc,removePunctuation)
* doc<-tm\_map(doc,stripWhitespace)
* doc<-tm\_map(doc,stemDocument,language="english")
* doc<-tm\_map(doc,tolower)
* doc<-tm\_map(doc,PlainTextDocument )
* moviec.clean<-doc
* **##document term matrix**
* moviedtm<-DocumentTermMatrix(doc)
* dim(moviedtm)
* **##Partitioning the Data**
* movie.train <- movie[1:1500,]
* movie.test <- movie[1501:2000,]
* moviedtm.train <- moviedtm[1:1500,]
* moviedtm.test <- moviedtm[1501:2000,]
* moviec.clean.train <- moviec.clean[1:1500]
* moviec.clean.test <- moviec.clean[1501:2000]
* **## filtering the data set**
* dim(moviedtm.train)
* fivefreq <- findFreqTerms(moviedtm.train, 95)
* length((fivefreq))
* **## document term matrices of training and test data**
* dtm.train.fw <- DocumentTermMatrix(moviec.clean.train, control=list(dictionary = fivefreq))
* dim(dtm.train.fw)
* dtm.test.fw <- DocumentTermMatrix(moviec.clean.test, control=list(dictionary = fivefreq))
* dim(dtm.test.fw)
* convert\_count <- function(x) {
* y <- ifelse(x > 0, 1,0)
* y <- factor(y, levels=c(0,1), labels=c("No", "Yes"))
* y
* }
* trainNB <- apply(dtm.train.fw, 2, convert\_count)
* testNB <- apply(dtm.test.fw, 2, convert\_count)
* **## training naive bayesian classifier**
* system.time( classifier <- naiveBayes(trainNB, movie.train$label) )
* system.time( pred <- predict(classifier, newdata=testNB) )
* table("Predictions"= pred, "Actual" = movie.test$label )
* confmatrix <- confusionMatrix(pred, movie.test$label)
* confmatrix

**RESULT**

* Naïve Bayesian classifier achieved 78.8 percent accuracy

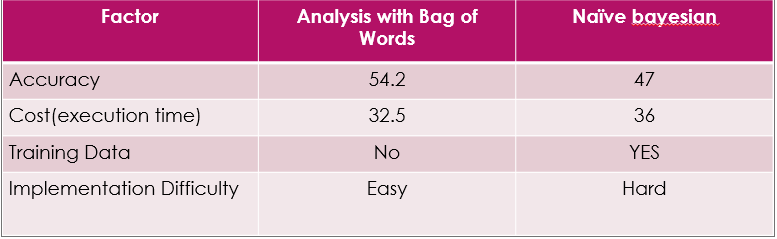


**CODE**

1. library(tm)
2. library(wordcloud)
3. library(caret)
4. library(RTextTools)
5. library(e1071)
6. library(dplyr)
7. setwd("C:/Users/ghari/Desktop/TermPaper")
8. neg<-read.csv("neg.csv",header = F,stringsAsFactors = T)
9. negc<-Corpus(VectorSource(neg))
10. negc<-tm\_map(negc,stemDocument,language="english")
11. negdtm<-DocumentTermMatrix(negc)
12. negwords<-findFreqTerms(negdtm)
13. str(negwords)
14. pos<-read.csv("pos.csv",header = F,stringsAsFactors = T)
15. posc<-Corpus(VectorSource(pos))
16. posc<-tm\_map(posc,stemDocument,language="english")
17. posdtm<-DocumentTermMatrix(posc)
18. poswords<-findFreqTerms(posdtm)
19. str(poswords)
20. movie<-read.csv("movie.csv",header = T,stringsAsFactors = F)
21. summary(movie)
22. moviec<-Corpus(VectorSource(movie$review)) #converting corpus
23. dim(moviec)
24. length(moviec)
25. doc<-tm\_map(moviec,removeNumbers)
26. doc<-tm\_map(doc,removeWords,stopwords("english"))
27. doc<-tm\_map(doc,removePunctuation)
28. doc<-tm\_map(doc,stripWhitespace)
29. tempp<-doc
30. doc<-tm\_map(doc,stemDocument,language="english")
31. moviedtm<-DocumentTermMatrix(doc)
32. dim(moviedtm)
33. temp<-moviedtm
34. rr<-removeSparseTerms(moviedtm,0.8)
35. dim(rr)
36. mean\_movie<-sort(colMeans(as.matrix(rr),na.rm = TRUE),decreasing = T)
37. mean\_movie[1:20]
38. barplot(mean\_movie[1:20],xlab="top 20 words",ylab="frequency",las=3,ylim=c(0,3))
39. wordcloud(names(mean\_movie[1:100]), mean\_movie[1:100], min.freq = 0,scale=c(2,1)
40. ,max.words=200, random.order=FALSE, rot.per=0.80,
41. colors=brewer.pal(8, "Dark2"))
42. unwanted<-c("film","movie")
43. unwanted<-tm\_map(tempp,removeWords,unwanted)
44. unwanted<-tm\_map(unwanted,stemDocument,language="english")
45. moviedtm<-DocumentTermMatrix(unwanted)
46. dim(moviedtm)
47. rr<-removeSparseTerms(moviedtm,0.8)
48. dim(rr)
49. mean\_movie<-sort(colMeans(as.matrix(rr),na.rm = TRUE),decreasing = T)
50. mean\_movie[1:20]
51. barplot(mean\_movie[1:20],xlab="top 20 words",ylab="frequency",las=3,ylim=c(0,3))
52. wordcloud(names(mean\_movie[1:100]), mean\_movie[1:100], min.freq = 0,scale=c(2,1)
53. ,max.words=210, random.order=FALSE, rot.per=0.80,
54. colors=brewer.pal(8, "Dark2"))
55. posdtm<-DocumentTermMatrix(doc, control=list(dictionary=poswords))
56. dim(posdtm)
57. posreviewrow<-rowSums(as.matrix(posdtm),na.rm = TRUE)
58. dim(posreviewrow)
59. negdtm<-DocumentTermMatrix(doc, control=list(dictionary=negwords))
60. dim(negdtm)
61. negreviewrow<-rowSums(as.matrix(negdtm),na.rm = TRUE)
62. x<-length(negreviewrow)
63. positive<-0
64. negative<-0
65. neutral<-0
66. hari<-function()
67. {
68. negrate<-colMeans( as.matrix(negdtm),na.rm = TRUE)
69. d<-length(negrate)
70. negt<-mean(negrate[1:1280])
71. posrate<-colMeans( as.matrix(posdtm),na.rm = TRUE)
72. e<-length(posrate)
73. post<-mean(posrate[1:e])
74. movierating<-post-negt
75. movierating<-movierating\*100
76. x<-length(negreviewrow)
77. positive<-0
78. negative<-0
79. neutral<-0
80. for( i in 1:x)
81. {
82. if( (posreviewrow[i]-negreviewrow[i])==4)
83. { neutral<-neutral+1
84. }
85. if( (posreviewrow[i]-negreviewrow[i])>4)
86. {positive<-positive+1 }
87. else
88. {
89. if( (posreviewrow[i]-negreviewrow[i])<4)
90. { negative<-negative+1} }}
91. cat("\nthe number of positive review ", positive)
92. cat("\nthe number of negative review",negative)
93. cat("\nthe number of neutral reviews",neutral)
94. cat("\nthe movie rating is ", movierating)
95. }
96. hari()
97. moviedtm<-DocumentTermMatrix(doc)
98. dim(moviedtm)
99. movie.train <- movie[1:1500,]
100. movie.test <- movie[1501:2000,]
101. moviedtm.train <- moviedtm[1:1500,]
102. moviedtm.test <- moviedtm[1501:2000,]
103. moviec.clean.train <- moviec.clean[1:1500]
104. moviec.clean.test <- moviec.clean[1501:2000]
105. dim(moviedtm.train)
106. fivefreq <- findFreqTerms(moviedtm.train, 95)
107. length((fivefreq))
108. dtm.train.fw <- DocumentTermMatrix(moviec.clean.train, control=list(dictionary = fivefreq))
109. dim(dtm.train.fw)
110. dtm.test.fw <- DocumentTermMatrix(moviec.clean.test, control=list(dictionary = fivefreq))
111. dim(dtm.test.fw)
112. convert\_count <- function(x) {
113. y <- ifelse(x > 0, 1,0)
114. y <- factor(y, levels=c(0,1), labels=c("No", "Yes"))
115. y
116. }
117. trainNB <- apply(dtm.train.fw, 2, convert\_count)
118. testNB <- apply(dtm.test.fw, 2, convert\_count)
119. system.time( classifier <- naiveBayes(trainNB, movie.train$label, laplace = 1) )
120. system.time( pred <- predict(classifier, newdata=testNB) )
121. table("Predictions"= pred, "Actual" = movie.test$label )
122. confmatrix <- confusionMatrix(pred, movie.test$label)
123. confmatrix

**4.RESULTS**

**4.1 COMPARISON**



**4.2 ANALYSIS**

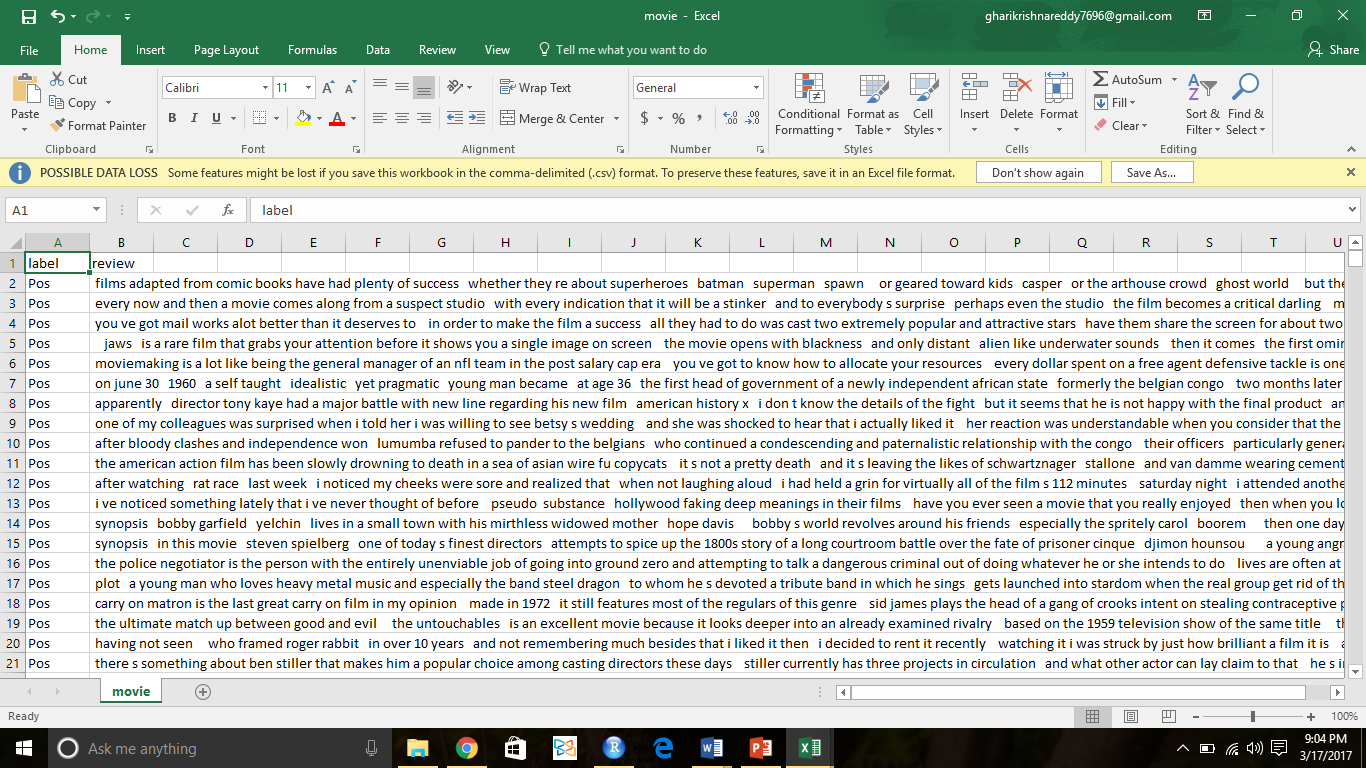
* The data set which I was used contains 2000 tuples totally
* It contains 1000 pos reviews and 1000 neg reviews
* The bag of Words techniques achieves 54.6 accuracy
* The Naïve Bayesian achieved 47 percent accuracy

**5. CONCLUSION**

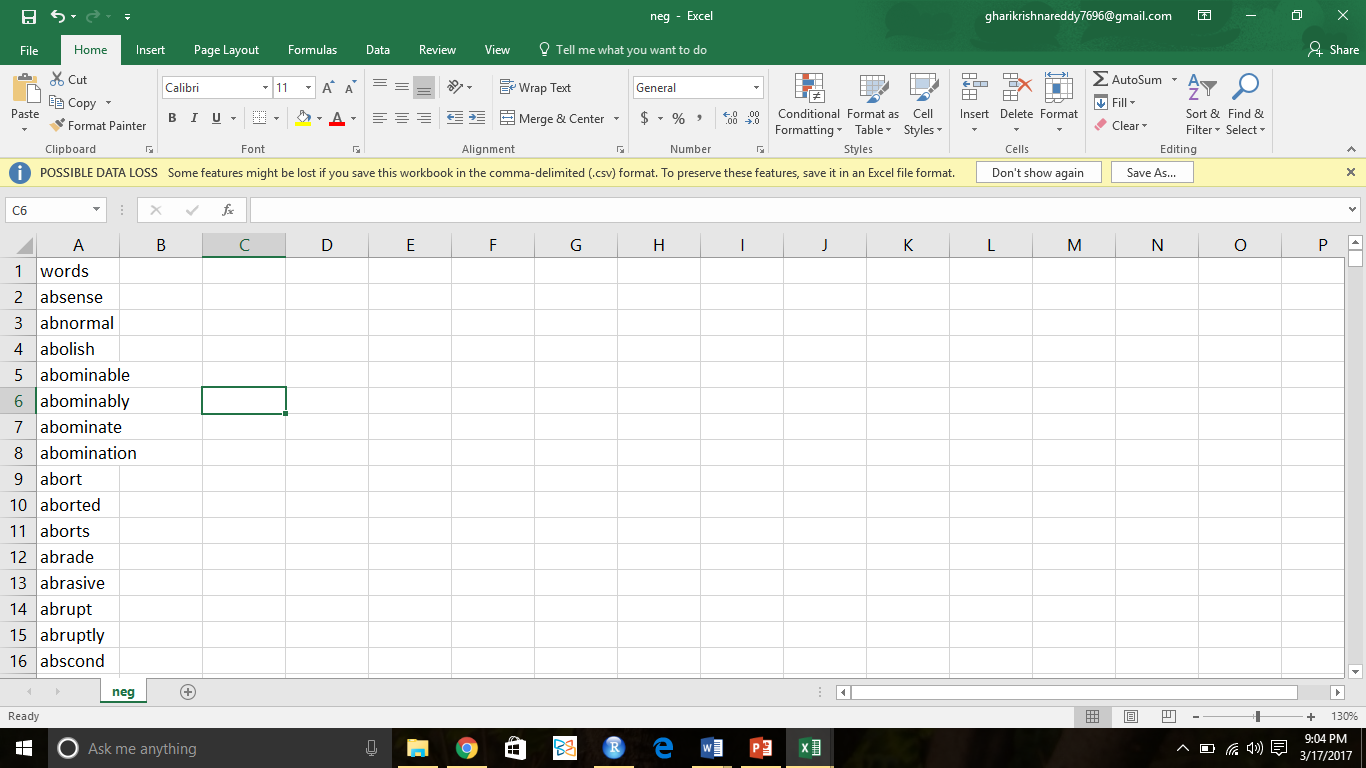
Naïve Bayesian can perform more efficient with larger training datasets than ‘bag of Words’. If you don’t have training data to train the algorithm then go to Bag of Words approach**.**

**DATA SETS USED IN ANALYSIS**

**MOVIE.CSV**



**NEG.CSV**



**POS.CSV**

