Weekly Assignment 5

Loic Roldan Waals & Kimberly Cheng

Wednesday 25 April, 2018

# Question 1

In this question we just need the text part of last week’s assignment. Before we answer the sub-questions, we first import the data and make a *corpus* and *document term matrix* (DTM). these will then be used for the next questions.

Amongst the words that were removed were the search terms *artificial, intelligence* and *artificialintelligence*, and several usernames.

#--------------------------------------  
# Load Relevant Packages & Tweets  
#--------------------------------------  
library(readr)  
library(wordcloud)  
library(jsonlite)  
library(tm)  
library(SnowballC)  
  
AITweets <- read\_delim("AITweets.csv",   
 ">", escape\_double = FALSE, trim\_ws = TRUE)  
  
# Remove emoji's  
AITweets$text <- sapply(AITweets$text, function(row) iconv(row, "latin1", "ASCII", sub = ""))  
  
#--------------------------------------  
# Create Corpus and Clean Textual Data  
#--------------------------------------  
  
# Create Corpus  
vecdata <- VectorSource(AITweets$text)  
myCorpus <- VCorpus(vecdata)   
  
  
# Remove the links  
removeLinks <- function (x) {  
 gsub("http [^[:blank:]]+","",x)  
}  
myCorpus <- tm\_map (myCorpus, content\_transformer(removeLinks))  
  
# Only lower case  
tweets\_corpus\_clean <- tm\_map(myCorpus, content\_transformer(tolower))   
  
# Remove punctuation  
tweets\_corpus\_clean <- tm\_map(tweets\_corpus\_clean, removePunctuation)  
  
# Remove numbers  
tweets\_corpus\_clean <- tm\_map(tweets\_corpus\_clean, removeNumbers)   
  
# Remove stopwords  
tweets\_corpus\_clean <- tm\_map(tweets\_corpus\_clean, removeWords, stopwords("english"))  
tweets\_corpus\_clean <- tm\_map(tweets\_corpus\_clean, removeWords, c("userexperienceu", "artificial", "intelligence", "artificialintelligence", "murthaburk", "will", "dbrainio"))   
  
# Remove unnecessary white space  
tweets\_corpus\_clean <- tm\_map(tweets\_corpus\_clean, stripWhitespace)   
  
#stemming  
tweets\_corpus\_clean <- tm\_map(tweets\_corpus\_clean, stemDocument, language = "english")  
  
myCorpus <- tweets\_corpus\_clean  
  
#--------------------------------------  
# Create Term Frequency Matrix  
#--------------------------------------  
  
dtm <- TermDocumentMatrix(tweets\_corpus\_clean)  
dtm

## <<TermDocumentMatrix (terms: 2467, documents: 1021)>>  
## Non-/sparse entries: 9078/2509729  
## Sparsity : 100%  
## Maximal term length: 42  
## Weighting : term frequency (tf)

## Part (a)

Because euclidean distance is normally used to calculate distance with k-means clustering (<https://www.mailman.columbia.edu/research/population-health-methods/cluster-analysis-using-k-means>) and because the dataset does not suffer too much from high dimensionality, euclidean distance was used to measure the distance.

We first remove the spare items to allow for a more easily understandable analysis. Then we plot a cluster dendrogram to see if we can see any natural clusters forming. Again, the euclidean distance metric was used. It appears that 3 groups have formed, these were labeled with the red boxes.

The first cluster appears to discuss machine learning and the new capabilities of robots (thanks to artificial intelligence). The second cluster is solely defined by the word *human*. This cluster could be encoding the comparison between humans and AI (e.g. when they become smarter than us, if they can mimic our behavior). The final cluster on the right seems to be about the (potential) behavior of AI, but this is highly speculative.

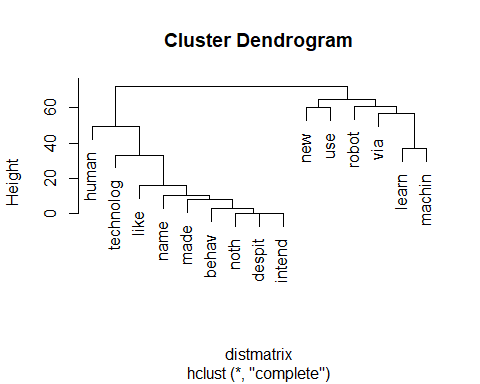
Finally the elbow method is used to determine whether our choice of 3 clusters seems reasonable. Plotting the within groups sum of squares against the number of clusters shows different values every time, but the large drop is always between 2 and 5 clusters. This means our choice of k=3 is reasonable.

We finally finish with a cluster plot that shows that two components explain 67.9% of the point variability. Additionally, there is not much overlap between the clusters. A decision tree will also be created to show how each cluster can be attained.

set.seed(156)  
  
#--------------------------------------  
# Load Relevant Packages  
#--------------------------------------  
library(cluster)  
  
# Remove sparse words  
dtm2 <- removeSparseTerms(dtm, sparse = 0.95)  
dtm2

## <<TermDocumentMatrix (terms: 15, documents: 1021)>>  
## Non-/sparse entries: 1193/14122  
## Sparsity : 92%  
## Maximal term length: 9  
## Weighting : term frequency (tf)

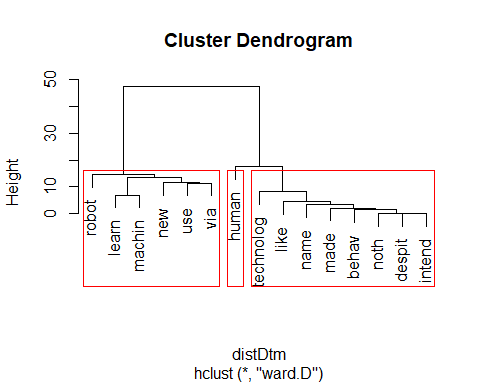
dtm2 <- as.matrix(dtm2)  
distmatrix <- dist(scale(dtm2))  
  
#--------------------------------------  
# Plot Dendrogram  
#--------------------------------------  
fit <- hclust(distmatrix)  
  
plot(fit)



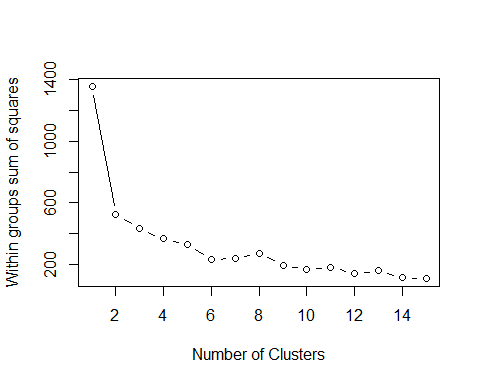
clusters = 3  
  
  
#--------------------------------------  
# K-Means Clustering  
#--------------------------------------  
distDtm <- dist(dtm2 , method = "euclidean")  
  
groups <- hclust(distDtm, method = "ward.D")  
groups

##   
## Call:  
## hclust(d = distDtm, method = "ward.D")  
##   
## Cluster method : ward.D   
## Distance : euclidean   
## Number of objects: 15

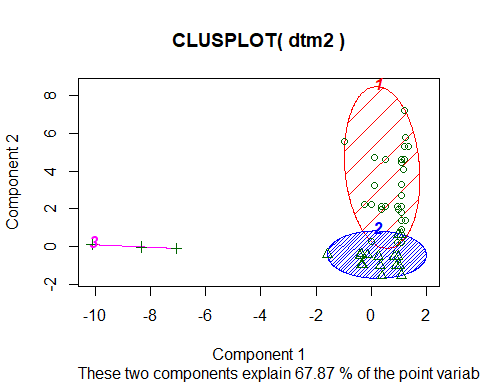
plot(groups)  
rect.hclust(groups, k=clusters)



dtm2 <- t(dtm2)  
  
  
#--------------------------------------  
# Elbow Method  
#--------------------------------------  
# Using elbow method to determine number of clusters  
x <- data.frame(dtm2)  
wss <- (nrow(x)-1)\*sum(apply(x,2,var))  
for (i in 2:15) wss[i] <- sum(kmeans(x, centers = i)$withinss)  
  
plot(1:15, wss, type="b", xlab="Number of Clusters", ylab="Within groups sum of squares")

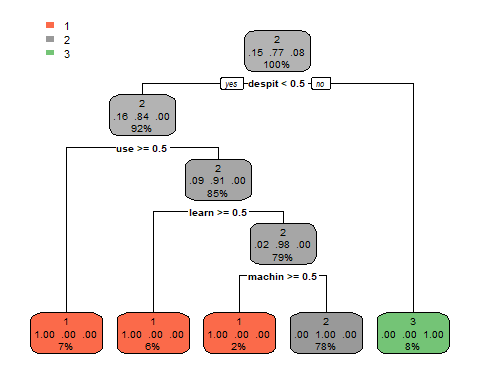


#--------------------------------------  
# Cluster Plot Principle Component Analysis  
#--------------------------------------  
rsltKmeans <- kmeans(dtm2, clusters)  
clusplot(dtm2, rsltKmeans$cluster, color = TRUE, shade = TRUE, labels = 4, lines = 0)



It seems like the model checks whether the stem word *despit* is used, in which case it belongs to the third cluster.Then it checks Whether any of the words *use, learn* or *machin* are being used, if any of them are used, it belongs to the first cluster. Anything else and it belongs to the second cluster, the one that solely contained the word *human*.

#--------------------------------------  
# Load Relevant Packages  
#--------------------------------------  
library(rpart)  
library(rpart.plot)  
  
#--------------------------------------  
# Plot Decision Tree  
#--------------------------------------  
  
#add clusters to the existing dataframe  
dfcluster <- data.frame(dtm2, k = as.factor(rsltKmeans$cluster))  
  
#train the tree  
tree <- rpart(k ~ ., data = dfcluster, method = "class")  
rpart.plot(tree)



## Part (b)

We will now proceed with the sentiment analysis for twitter. We first import the file and clean the text in the same way as in **Part (a)**. After we have obtained the DTM we append the manual sentiment labels to create a data frame that contains the usage of the terms and the sentiment as columns. Here we find the first variable where we have a choice, *sparsity*. Using a trail an error method, we found that the optimal value for this is 0.96.

We then apply ten fold cross validation. Here we find the second variable we can tweak, the threshold of the prediction for the Naive Bayes model. Again, using the trail and error method we found a optimal threshold of 0.7. Finally to evaluate the performance we use accuracy and a confusion matrix. The code below was used:

#load libraries  
  
#--------------------------------------  
# Load Relevant Packages  
#--------------------------------------  
library("e1071")  
library("plyr")  
library(naivebayes)  
library(caret)  
library(pROC)  
  
# Set the values for the sparsity and the threshold for NB (easier tweaking).  
spar = 0.96  
nbt = 0.7  
  
#--------------------------------------  
# Convert to Corpus and Clean Text Data  
#--------------------------------------  
  
# Import the dataset and create corpus  
AITweets <- read\_delim("C:/Users/Loic RW/Google Drive/Big Data and Business Analytics/Assignments/Assignment 5/AITweets\_sentiment.txt", ">", escape\_double = FALSE, trim\_ws = TRUE)  
  
AITweets$sentiment <- as.factor(AITweets$sentiment)  
  
# Remove emoji's  
AITweets$text <- sapply(AITweets$text, function(row) iconv(row, "latin1", "ASCII", sub = ""))  
  
# Create corpus  
vecdata <- VectorSource(AITweets$text)  
myCorpus <- VCorpus(vecdata)   
  
# Remove the links  
removeLinks <- function (x) {  
 gsub("http [^[:blank:]]+","",x)  
}  
myCorpus <- tm\_map (myCorpus, content\_transformer(removeLinks))  
  
# Only lower case  
tweets\_corpus\_clean <- tm\_map(myCorpus, content\_transformer(tolower))   
  
# Remove punctuation  
tweets\_corpus\_clean <- tm\_map(tweets\_corpus\_clean, removePunctuation)  
  
# Remove numbers  
tweets\_corpus\_clean <- tm\_map(tweets\_corpus\_clean, removeNumbers)   
  
# Remove stopwords  
tweets\_corpus\_clean <- tm\_map(tweets\_corpus\_clean, removeWords, stopwords("english"))  
tweets\_corpus\_clean <- tm\_map(tweets\_corpus\_clean, removeWords,   
 c("userexperienceu", "artificial", "intelligence", "artificialintelligence",   
 "murthaburk", "will", "dbrainio"))   
  
# Remove unnecessary white space  
tweets\_corpus\_clean <- tm\_map(tweets\_corpus\_clean, stripWhitespace)   
  
# Stemming  
tweets\_corpus\_clean <- tm\_map(tweets\_corpus\_clean, stemDocument, language = "english")  
  
myCorpus <- tweets\_corpus\_clean  
  
#--------------------------------------  
# Make Term Frequency Matrix  
#--------------------------------------  
  
# Make term frequency matrix  
dtm <- TermDocumentMatrix(tweets\_corpus\_clean)  
dtm2 <- removeSparseTerms(dtm, sparse = spar)  
dtm2 <- t(as.matrix(dtm2))  
  
# Turn the dtm into a data frame  
dfsentiment <- data.frame(dtm2)  
dfsentiment$target <- as.factor(AITweets$sentiment)  
  
# Delete any potential missing values due to improper manual labeling  
dfsentiment <- dfsentiment[complete.cases(dfsentiment),]  
  
# Randomize the data  
dfsentiment <- dfsentiment[sample(1:nrow(dfsentiment)), ]  
  
#--------------------------------------  
# 10-Fold Cross Validation & Naive Bayes Model  
#--------------------------------------  
  
# Create 10 equally sized folds  
nFolds <- 10  
folds <- cut(seq(1, nrow(dfsentiment)),  
 breaks = nFolds,  
 labels = FALSE)  
  
# Vectors to store the results initialized with zeros  
pdNB <- NULL  
actual <- NULL  
  
# Define the model  
mdl <- as.factor(target) ~ .  
  
# Perform 10 fold cross validation  
for(i in 1:nFolds){  
  
 #Segment the data by fold using which-function  
 testIndexes <- which(folds == i, arr.ind = FALSE)  
 testData <- dfsentiment[testIndexes, ]  
 trainData <- dfsentiment[-testIndexes, ]  
   
 #train the model  
 rsltNB <- naiveBayes(mdl, data = trainData)  
   
 #Make predictions on the test set  
 NBClass <- predict(rsltNB, testData, type="class", threshold = nbt)  
   
 #save the results to calculate accuracy later on  
 pdNB <- c(as.character(pdNB), as.character(NBClass))  
 actual <- c(actual, as.character(testData$target))  
}   
  
meanAccNB = mean(pdNB == actual)  
meanAccNB

## [1] 0.6967509

actual <- as.factor(actual)  
pdNB <- as.factor(pdNB)  
  
confusionMatrix(pdNB, actual, positive = "1")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction -1 0 1  
## -1 0 4 4  
## 0 11 144 40  
## 1 1 24 49  
##   
## Overall Statistics  
##   
## Accuracy : 0.6968   
## 95% CI : (0.6389, 0.7503)  
## No Information Rate : 0.6209   
## P-Value [Acc > NIR] : 0.005062   
##   
## Kappa : 0.3574   
## Mcnemar's Test P-Value : 0.028418   
##   
## Statistics by Class:  
##   
## Class: -1 Class: 0 Class: 1  
## Sensitivity 0.00000 0.8372 0.5269  
## Specificity 0.96981 0.5143 0.8641  
## Pos Pred Value 0.00000 0.7385 0.6622  
## Neg Pred Value 0.95539 0.6585 0.7833  
## Prevalence 0.04332 0.6209 0.3357  
## Detection Rate 0.00000 0.5199 0.1769  
## Detection Prevalence 0.02888 0.7040 0.2671  
## Balanced Accuracy 0.48491 0.6757 0.6955

As can be seen the accuracy of the model is 69.8%, simply choosing the most common factor would result in an accuracy of 62.1%. This shows that our model does not show great performance, but it is better than simply predicting neutral all the time. In the confusion matrix we see that the model classifies 0 most of the time correctly (83.7%). The sensitivity class 1 and -1 is poor, with 52.7% and 0% respectively.

# Question 2

First we collect all the data with *newsapi*, these are then put into data frames and saved for later use.

## Part (a)

#--------------------------------------  
# Load Relevant Files and Packages  
#--------------------------------------  
load(file = "cars.rda")  
  
# Read lines of Positive and Negative Words  
positive\_words <- readLines("positive\_words.txt")  
negative\_words <- readLines("negative\_words.txt")   
  
library(stringi) #to more cleanly transform to lower  
  
  
#--------------------------------------  
# Prepare Corpus and Clean Textual Data  
#--------------------------------------  
  
# Remove links Function   
removeLinks <- function (x) {  
 gsub("http [^[:blank:]]+","",x)  
}  
  
# Create a function to convert all strings to UTF   
convertUTF <- function(d) {   
 d$description <- iconv(d$description, "latin1", "ASCII")  
 d$description <- iconv(d$description, "ASCII", "UTF-8")  
 # Extract only complete cases  
 d <- d[complete.cases(d), ]  
}  
  
# Create a function to Clean each Corpus   
clean\_corpus <- function(x) {   
   
 x <- convertUTF(x)  
   
 vecData <- VectorSource(x$description)  
 myCorpus <- VCorpus(vecData)  
   
 # Remove Links  
 myCorpus <- tm\_map(myCorpus, content\_transformer(removeLinks))  
   
 # Convert words to lowercase  
 myCorpus <- tm\_map(myCorpus, content\_transformer(stri\_trans\_tolower))  
   
 # Remove punctuation  
 myCorpus <- tm\_map(myCorpus, removePunctuation)  
   
 # Remove numbers  
 myCorpus <- tm\_map(myCorpus, removeNumbers)  
   
 # Remove stop-words  
 myCorpus <- tm\_map(myCorpus, removeWords, stopwords("english"))  
   
 # Remove extra white spaces  
 myCorpus <- tm\_map(myCorpus, stripWhitespace)  
   
 # Stemming  
 myCorpus<- tm\_map(myCorpus, stemDocument, language = "english")  
   
}  
  
# Run function to clean corpus  
myCorpus.Volkswagen <- clean\_corpus(df.Volkswagen)  
myCorpus.Toyota <- clean\_corpus(df.Toyota)  
myCorpus.Honda <- clean\_corpus(df.Honda)  
myCorpus.BMW <- clean\_corpus(df.BMW)  
myCorpus.Ford <- clean\_corpus(df.Ford)  
myCorpus.MercedesBenz <- clean\_corpus(df.MercedesBenz)  
myCorpus.Volvo <- clean\_corpus(df.Volvo)  
myCorpus.Tesla <- clean\_corpus(df.Tesla)

#--------------------------------------  
# Score Each Corpus  
#--------------------------------------  
  
# Create a function to Score each Corpus  
  
score\_corpus <- function(y) {   
 # Initiate the Score Topic  
 scoreTopic <- 0  
   
 #Start a loop over the documents  
 for(ifold in 1: length(y)) {  
 terms <- unlist(strsplit(y[[ifold]]$content, " "))  
 pos\_matches <- sum(terms %in% positive\_words)  
 neg\_matches <- sum(terms %in% negative\_words)  
 scoreTopic[ifold] <- pos\_matches - neg\_matches  
 } # End of the for loop  
   
 df <- data.frame(text = sapply(y, paste, collapse = " "), stringsAsFactors = FALSE)  
 df$score\_topic <- scoreTopic  
}   
  
score.Volkswagen <- score\_corpus(myCorpus.Volkswagen)  
score.Toyota <- score\_corpus(myCorpus.Toyota)  
score.Honda <- score\_corpus(myCorpus.Honda)  
score.BMW <- score\_corpus(myCorpus.BMW)  
score.Ford <- score\_corpus(myCorpus.Ford)  
score.MercedesBenz <- score\_corpus(myCorpus.MercedesBenz)  
score.Volvo <- score\_corpus(myCorpus.Volvo)  
score.Tesla <- score\_corpus(myCorpus.Tesla)

# Merge all scores to plot  
df <- cbind(score.Tesla, score.BMW, score.Ford, score.Honda,   
 score.Toyota, score.MercedesBenz, score.Volkswagen, score.Volvo)  
  
#--------------------------------------  
# Evaluate Sentiment  
#--------------------------------------  
  
# Total Sum of Scores of Topics (Positive Words - Negative Words) / Mean of Scores of TOpics  
sums <- data.frame(Sums = colSums(df, na.rm = TRUE))  
sums

Sums

score.Tesla -13 score.BMW -20 score.Ford 64 score.Honda 27 score.Toyota 78 score.MercedesBenz 108 score.Volkswagen 37 score.Volvo -32

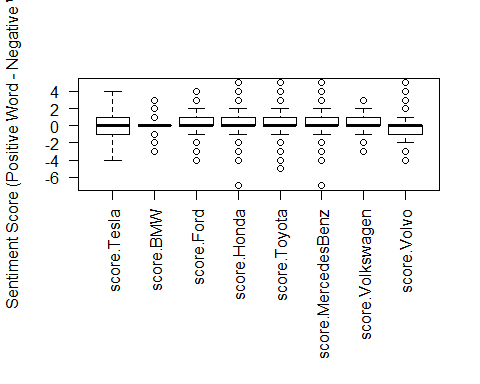
means <- data.frame(Means = colMeans(df, na.rm = TRUE))  
means

Means

score.Tesla -0.02233677 score.BMW -0.03436426 score.Ford 0.10996564 score.Honda 0.04639175 score.Toyota 0.13402062 score.MercedesBenz 0.18556701 score.Volkswagen 0.06357388 score.Volvo -0.05498282

When evaluating which brand has the most positive sentiment, we initially evaluated the sum of the sentiment where *sentiment = positive words - negative words*. The highest in this data is Mercedes Benz with a sentiment score from the whole corpus of 108. This means that Mercedes Benz has the most positive terms relative to negative terms out of all brands. When evaluating the average sentiment score per document, it is seen that again Mercedes Benz has the largest score with an average 0.18 score out of all documents within the corpus. In order to get better visualization of the distribution of these sentiment scores, a box plot was plotted.

#--------------------------------------  
# Plot Boxplot  
#--------------------------------------  
par(mar = c(10,4,4,2) + 0.1)  
boxplot(df, ylab = "Sentiment Score (Positive Word - Negative Word)", las = 2)

 From the box plot, it is visible that the median sentiment score for the documents within each corpus all fall around the same range ~0. The distribution of sentiment scores appear to be largely skewed with only Tesla showing a relatively normal distribution. Ford, Honda, Toyota, Mercedes-Benz, and Volkswagen are all more negatively-skewed, while Volvo is positively-skewed. Additionally, BMW appears to have the most consistent score.

## Part (b)

Here we simply chose a value of 0 for the cutoff. This means that a document with a sentiment score higher than 0 will be labeled as 1, while the ones with a sentiment score of less than 0 will be labeled as -1, the rest is labeled as 0.

#--------------------------------------  
# Prepare Term Frequency Matix into Data Frame  
#--------------------------------------  
  
# Create a Function to convert the corpus to a data frame   
# with the calculated sentiment scores  
  
convert2DF <- function(a, b) {   
 df <- cbind(data.frame(text = sapply(a, paste, collapse = " "),   
 stringsAsFactors = FALSE), score\_topic = b)  
}  
  
# Run the functions for each brand  
df.Volkswagen.Corp <- convert2DF(myCorpus.Volkswagen, score.Volkswagen)  
df.Toyota.Corp <- convert2DF(myCorpus.Toyota, score.Toyota)  
df.Honda.Corp <- convert2DF(myCorpus.Honda, score.Honda)  
df.BMW.Corp <- convert2DF(myCorpus.BMW, score.BMW)  
df.MercedesBenz.Corp <- convert2DF(myCorpus.MercedesBenz, score.MercedesBenz)  
df.Ford.Corp <- convert2DF(myCorpus.Ford, score.Ford)  
df.Volvo.Corp <- convert2DF(myCorpus.Volvo, score.Volvo)  
df.Tesla.Corp <- convert2DF(myCorpus.Tesla, score.Tesla)  
  
  
# Merge all dataframes  
df <- rbind(df.Volkswagen.Corp, df.Toyota.Corp,   
 df.Honda.Corp, df.Ford.Corp, df.BMW.Corp,   
 df.MercedesBenz.Corp, df.Tesla.Corp, df.Volvo.Corp)  
  
# Classify Sentiment based on cutoffs  
# Cutoffs ( > 1 is "1", < 1 is "-1", and 0 is "0")  
df$sentiment[df$score\_topic > 0] <- 1  
df$sentiment[df$score\_topic < 0] <- -1  
df$sentiment[df$score\_topic == 0] <- 0  
  
# Convert Back to Corpus  
vecData <- VectorSource(df$text)  
myCorpus <- VCorpus(vecData)  
  
# Convert to Document Term Matrix -   
dtm <- DocumentTermMatrix(myCorpus)  
  
# Remove Sparse Terms  
dtm2 <- removeSparseTerms(dtm, sparse = 0.99)  
dtm2 <- as.matrix(dtm2)  
  
# Create Data Frame for Predictions  
df.news <- data.frame(cbind(dtm2, sentiment = df$sentiment))  
df.news$sentiment <- as.factor(df.news$sentiment)  
  
#----------------------------------  
# 10-Fold Cross Validation - Prep  
#----------------------------------  
  
# Prepare Cross Validation  
# 1 - Randomize the order of the observations  
df.news <- df.news[sample(1:nrow(df.news)), ]  
  
# 2 - Create 10 equally sized folds - (10 = K)  
n.Folds <- 10  
Folds<- cut(seq(1, nrow(df.news)),   
 breaks = n.Folds,   
 labels = FALSE)

#----------------------------------  
# Naive Bayes Classifier   
#----------------------------------  
  
library(caret)  
library(e1071)  
  
# Define the Model  
  
mdl.news <- sentiment ~ .   
  
# Create Empty Vectors to Collect Results  
accuracy.NB <- rep(NA, n.Folds)  
accuracy.SVM <- rep(NA, n.Folds)  
  
# Perform the 10-fold cross validation for Naive Bayes  
#Set a threshold   
nbt = 0.5  
  
for(i in 1: n.Folds) {   
   
 testIndexes <- which(Folds == i, arr.ind = TRUE)  
 df.news.test <- df.news[testIndexes, ]  
 df.news.train <- df.news[-testIndexes, ]  
   
 #Fit NB Model  
 rslt.NB <- naiveBayes(mdl.news, data = df.news.train)  
 #Fit SVM Model  
 rslt.SVM <- svm(mdl.news, data = df.news.train, type = "C-classification", probability = TRUE)  
   
 #Predicted Classification Test Set  
 pred.NB <- predict(rslt.NB, df.news.test, type = "class", threshold = nbt)  
 pred.SVM <- predict(rslt.SVM, df.news.test, type = "response")  
   
 #Accuracy  
 accuracy.NB[i] <- mean(pred.NB == df.news.test$sentiment)  
 accuracy.SVM[i] <- mean(pred.SVM == df.news.test$sentiment)  
   
}  
  
# Calculate Results for Naive Bayes Model: Accuracy  
JSONAccNB <- mean(accuracy.NB)  
JSONAccNB

## [1] 0.5899393

JSONAccSVM <- mean(accuracy.SVM)  
JSONAccSVM

## [1] 0.7192802

In this data, SVM has the highest accuracy in predicting the sentiment for the news articles based on textual data with a mean accuracy of mean(accuracy.SVM) over a Naive Bayes which has a mean accuracy of mean(accuracy.SVM).