British Broadcasting Corporation (BBC) News Articles Text Classification

Solon Licas

Summary:

The use of text categorization has been widely studied and implemented from spam identification, documentation categorization, medical diagnosis. This project will try to implement some of those studies to make proper classification on a set of documents using different classifiers or combinations of each. We will use f measure as key metrics in evaluating the results.

Data:

The original source data is from the public data set on BBC news articles used in a publication by D. Greene and P. Cunningham. “Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering”, Proc. ICML 2006.

Methodology:

1. Check dataset if there are missing values, check the data types, and number of columns.
2. Make exploratory analysis by checking the distribution of the data.
3. Convert text to corpus and document term matrix.
4. Divide the dataset to training and testing sets.
5. Implement classifiers and make predictions.
6. Evaluate the results using key metrics.
7. Provide conclusion.

LITERATURE REVIEW:

1. A Survey of Text Classification Algorithms by Charu C. Aggarwal, ChengXiang Zhai A survey of a wide variety of text classification algorithms.
2. Naive Bayes and Text Classification by Sebastian Raschka Introduces the theory of naive Bayes classifiers and the basic concepts of text classification and implement those concepts to train a naive Bayes spam filter and apply naive Bayes to song classification based on lyrics.
3. Ensemble Methods: An Empirical Study by David Opitz, Richard Maclin The paper evaluates bagging and boosting methods on 23 data sets using both neural networks and decision trees as our classification algorithm.
4. Soft-Supervised Learning for Text Classification by Amarnag Subramanya & Jeff Bilmes The paper proposes a new graph-based semi-supervised learning (SSL) algorithm and demonstrate its application to document categorization.
5. An Improved Random Forest Classifier for Text Categorization by Baoxun Xu, Xiufeng Guo, Jiefeng Cheng The paper proposes an improved random forest algorithm for classifying text data. The algorithm is particularly designed for analyzing very high dimensional data with multiple classes.
6. Text Classification by Augmenting Bag of Words (BOW) Representation with Co-occurrence Feature by Soumya George K, Shibily Joseph The paper proposes a way to find co-occurrence feature from anchor text of wikipedia pages, proposes a way to incorporate co-occurrence feature to BOW model. The method is analyzed to know how it performs in task of text classification.

DATA DESCRIPTION:

The source data is from the public data set on BBC news articles used in a publication by D. Greene and P. Cunningham. “Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering”, Proc. ICML 2006.

Consists of 2225 documents from the BBC news website corresponding to stories in five topical areas from 2004-2005.

There are five categories: Business, Entertainment, Politics, Sports, Technology

There are no missing values in any of the two columns: Category and Text

Category column has 5 classes.

Text Columns are title and body of the articles concatenated.

RETRIEVAL OF DATASET:

#OPENING DATA  
  
raw\_data<- read.csv("bbc-text.csv",header=T)  
df<- raw\_data  
  
#head(df)  
str(df)

## 'data.frame': 2225 obs. of 2 variables:  
## $ category: Factor w/ 5 levels "business","entertainment",..: 5 1 4 4 2 3 3 4 4 2 ...  
## $ text : Factor w/ 2126 levels "$1m payoff for former shell boss shell is to pay $1m (Â£522 000) to the ex-finance chief who stepped down from "| \_\_truncated\_\_,..: 1856 2097 1798 2112 1379 897 180 830 2068 1089 ...

#identify missing values  
  
table(is.na(df$category))

##   
## FALSE   
## 2225

table(is.na(df$text))

##   
## FALSE   
## 2225

prop.table(table(df$category))

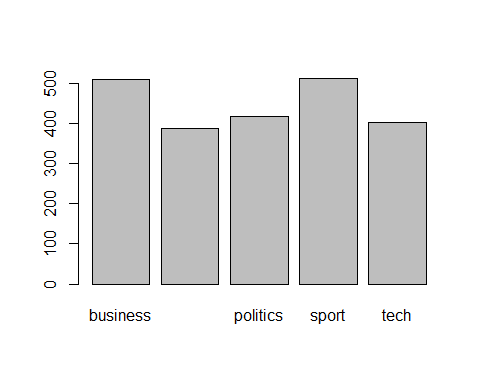
##   
## business entertainment politics sport tech   
## 0.2292135 0.1734831 0.1874157 0.2296629 0.1802247

EXPLORATORY ANALYSIS OF DATASET:

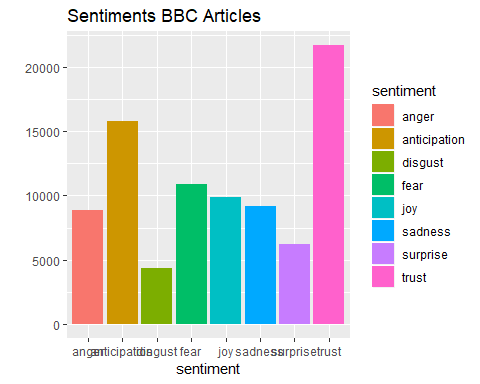
table(df$category)

##   
## business entertainment politics sport tech   
## 510 386 417 511 401

barplot(table(df$category))  
  
#counts for each category is about close to each other, no need to balance the categories.  
  
#sentiment analysis  
  
#install.packages("RSentiment")  
library(RSentiment)  
#install.packages("syuzhet")  
library(syuzhet)  
library(ggplot2)



df$text= as.character(df$text)  
d<-get\_nrc\_sentiment(df$text)  
td<-data.frame(t(d))  
td\_new <- data.frame(rowSums(td[1:2225]))  
names(td\_new)[1] <- "count"  
td\_new <- cbind("sentiment" = rownames(td\_new), td\_new)  
rownames(td\_new) <- NULL  
td\_new2<-td\_new[1:8,]  
  
qplot(sentiment, data=td\_new2, weight=count, geom="bar",fill=sentiment)+ggtitle("Sentiments BBC Articles")



#Creating wordcloud for ALL Categories  
  
#install.packages("NLP")  
library(NLP)

##   
## Attaching package: 'NLP'

## The following object is masked from 'package:ggplot2':  
##   
## annotate

#install.packages("RColorBrewer")  
library(RColorBrewer)  
#install.packages("tm")  
library(tm)  
#install.packages("wordcloud")  
library(wordcloud)  
  
  
docs <- Corpus(VectorSource(t(df$text)))  
#inspect(docs[1:3])  
  
docs <- tm\_map(docs, removeNumbers)

## Warning in tm\_map.SimpleCorpus(docs, removeNumbers): transformation drops  
## documents

docs <- tm\_map(docs, removeWords, stopwords("english"))

## Warning in tm\_map.SimpleCorpus(docs, removeWords, stopwords("english")):  
## transformation drops documents

docs <- tm\_map(docs, removePunctuation)

## Warning in tm\_map.SimpleCorpus(docs, removePunctuation): transformation  
## drops documents

docs <- tm\_map(docs, tolower)

## Warning in tm\_map.SimpleCorpus(docs, tolower): transformation drops  
## documents

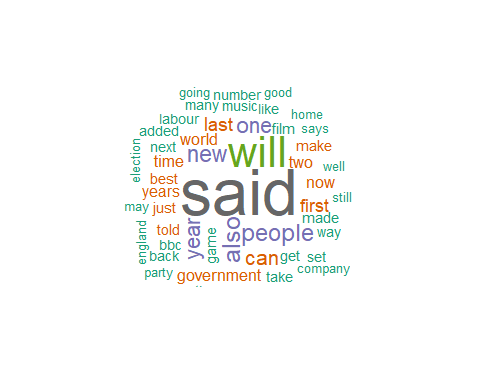
docs <- tm\_map(docs, stripWhitespace)

## Warning in tm\_map.SimpleCorpus(docs, stripWhitespace): transformation drops  
## documents

#inspect(docs[1:3])  
  
dtm <- TermDocumentMatrix(docs)  
#inspect(dtm)  
m <- as.matrix(dtm)  
v <- sort(rowSums(m),decreasing=TRUE)  
d <- data.frame(word = names(v),freq=v)  
  
dftot<-head(d,10)  
dftot

## word freq  
## said said 7254  
## will will 4472  
## also also 2156  
## people people 2044  
## new new 1970  
## year year 1860  
## one one 1814  
## can can 1668  
## last last 1381  
## first first 1282

wordcloud(words = d$word, freq = d$freq, min.freq = 1,  
 max.words=50, random.order=FALSE, rot.per=0.1,   
 colors=brewer.pal(8, "Dark2"))



##Creating word cloud for each category  
  
df.business <- df[df$category == "business",]  
df.entertainment <- df[df$category == "entertainment",]  
df.politics <- df[df$category == "politics",]  
df.sport <- df[df$category == "sport",]  
df.tech <- df[df$category == "tech",]  
  
  
#Wordcloud for Business  
docs.bus <- Corpus(VectorSource(t(df.business$text)))  
#inspect(docs.bus[1:3])  
  
docs.bus <- tm\_map(docs.bus, removeNumbers)

## Warning in tm\_map.SimpleCorpus(docs.bus, removeNumbers): transformation  
## drops documents

docs.bus <- tm\_map(docs.bus, removeWords, stopwords("english"))

## Warning in tm\_map.SimpleCorpus(docs.bus, removeWords,  
## stopwords("english")): transformation drops documents

docs.bus <- tm\_map(docs.bus, removePunctuation)

## Warning in tm\_map.SimpleCorpus(docs.bus, removePunctuation): transformation  
## drops documents

docs.bus <- tm\_map(docs.bus, tolower)

## Warning in tm\_map.SimpleCorpus(docs.bus, tolower): transformation drops  
## documents

docs.bus <- tm\_map(docs.bus, removeWords, c("said","also","however","will","last","just","can"))

## Warning in tm\_map.SimpleCorpus(docs.bus, removeWords, c("said", "also", :  
## transformation drops documents

docs.bus <- tm\_map(docs.bus, stripWhitespace)

## Warning in tm\_map.SimpleCorpus(docs.bus, stripWhitespace): transformation  
## drops documents

#inspect(docs.bus[1:3])  
  
dtm.bus <- TermDocumentMatrix(docs.bus)  
#inspect(dtm.bus)  
m.bus <- as.matrix(dtm.bus)  
v.bus <- sort(rowSums(m.bus),decreasing=TRUE)  
d.bus <- data.frame(word = names(v.bus),freq=v.bus)  
  
topword.bus<-head(d.bus,10)  
topword.bus

## word freq  
## year year 647  
## market market 425  
## new new 416  
## company company 415  
## growth growth 384  
## firm firm 362  
## economy economy 359  
## government government 340  
## bank bank 335  
## sales sales 316

wordcloud(words = d.bus$word, freq = d.bus$freq, min.freq = 1,  
 max.words=50,scale=c(3,0.2), random.order=FALSE, rot.per=0.1,   
 colors=brewer.pal(8, "Dark2"))



#Wordcloud for Entertainment  
docs.ent <- Corpus(VectorSource(t(df.entertainment$text)))  
#inspect(docs.bus[1:3])  
  
docs.ent <- tm\_map(docs.ent, removeNumbers)

## Warning in tm\_map.SimpleCorpus(docs.ent, removeNumbers): transformation  
## drops documents

docs.ent <- tm\_map(docs.ent, removeWords, stopwords("english"))

## Warning in tm\_map.SimpleCorpus(docs.ent, removeWords,  
## stopwords("english")): transformation drops documents

docs.ent <- tm\_map(docs.ent, removePunctuation)

## Warning in tm\_map.SimpleCorpus(docs.ent, removePunctuation): transformation  
## drops documents

docs.ent <- tm\_map(docs.ent, tolower)

## Warning in tm\_map.SimpleCorpus(docs.ent, tolower): transformation drops  
## documents

docs.ent <- tm\_map(docs.ent, removeWords, c("said","also","however","will","last","just","can"))

## Warning in tm\_map.SimpleCorpus(docs.ent, removeWords, c("said", "also", :  
## transformation drops documents

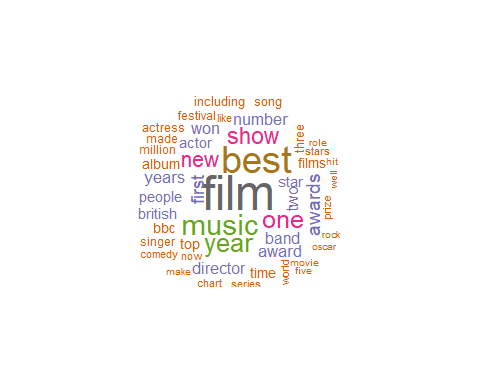
docs.ent <- tm\_map(docs.ent, stripWhitespace)

## Warning in tm\_map.SimpleCorpus(docs.ent, stripWhitespace): transformation  
## drops documents

#inspect(docs.bus[1:3])  
  
dtm.ent <- TermDocumentMatrix(docs.ent)  
#inspect(dtm.ent)  
m.ent <- as.matrix(dtm.ent)  
v.ent <- sort(rowSums(m.ent),decreasing=TRUE)  
d.ent <- data.frame(word = names(v.ent),freq=v.ent)  
  
topword.ent<-head(d.ent,10)  
topword.ent

## word freq  
## film film 753  
## best best 591  
## music music 435  
## year year 378  
## one one 372  
## show show 328  
## new new 322  
## awards awards 273  
## first first 251  
## award award 233

wordcloud(words = d.ent$word, freq = d.ent$freq, min.freq = 1,  
 max.words=50,scale=c(3,0.2), random.order=FALSE, rot.per=0.1,   
 colors=brewer.pal(8, "Dark2"))



#Wordcloud for Politics  
docs.pol <- Corpus(VectorSource(t(df.politics$text)))  
#inspect(docs.pol[1:3])  
  
docs.pol <- tm\_map(docs.pol, removeNumbers)

## Warning in tm\_map.SimpleCorpus(docs.pol, removeNumbers): transformation  
## drops documents

docs.pol <- tm\_map(docs.pol, removeWords, stopwords("english"))

## Warning in tm\_map.SimpleCorpus(docs.pol, removeWords,  
## stopwords("english")): transformation drops documents

docs.pol <- tm\_map(docs.pol, removePunctuation)

## Warning in tm\_map.SimpleCorpus(docs.pol, removePunctuation): transformation  
## drops documents

docs.pol <- tm\_map(docs.pol, tolower)

## Warning in tm\_map.SimpleCorpus(docs.pol, tolower): transformation drops  
## documents

docs.pol <- tm\_map(docs.pol, removeWords, c("said","also","however","will","last","just","can"))

## Warning in tm\_map.SimpleCorpus(docs.pol, removeWords, c("said", "also", :  
## transformation drops documents

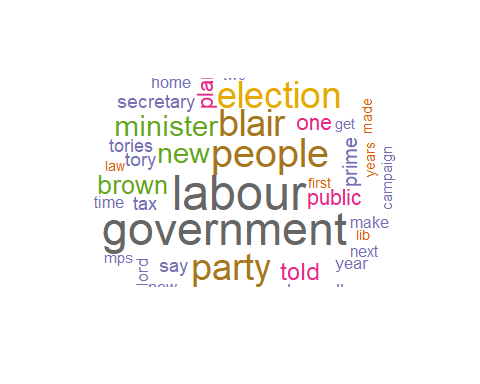
docs.pol <- tm\_map(docs.pol, stripWhitespace)

## Warning in tm\_map.SimpleCorpus(docs.pol, stripWhitespace): transformation  
## drops documents

#inspect(docs.pol[1:3])  
  
dtm.pol <- TermDocumentMatrix(docs.pol)  
#inspect(dtm.ent)  
m.pol <- as.matrix(dtm.pol)  
v.pol <- sort(rowSums(m.pol),decreasing=TRUE)  
d.pol <- data.frame(word = names(v.pol),freq=v.pol)  
  
topword.pol<-head(d.pol,10)  
topword.pol

## word freq  
## labour labour 760  
## government government 730  
## people people 623  
## party party 575  
## blair blair 572  
## election election 569  
## minister minister 433  
## new new 430  
## brown brown 384  
## told told 359

wordcloud(words = d.pol$word, freq = d.pol$freq, min.freq = 1,  
 max.words=50,scale=c(3,0.2), random.order=FALSE, rot.per=0.1,   
 colors=brewer.pal(8, "Dark2"))



#Wordcloud for Sports  
docs.spo <- Corpus(VectorSource(t(df.sport$text)))  
#inspect(docs.spo[1:3])  
  
docs.spo <- tm\_map(docs.spo, removeNumbers)

## Warning in tm\_map.SimpleCorpus(docs.spo, removeNumbers): transformation  
## drops documents

docs.spo <- tm\_map(docs.spo, removeWords, stopwords("english"))

## Warning in tm\_map.SimpleCorpus(docs.spo, removeWords,  
## stopwords("english")): transformation drops documents

docs.spo <- tm\_map(docs.spo, removePunctuation)

## Warning in tm\_map.SimpleCorpus(docs.spo, removePunctuation): transformation  
## drops documents

docs.spo <- tm\_map(docs.spo, tolower)

## Warning in tm\_map.SimpleCorpus(docs.spo, tolower): transformation drops  
## documents

docs.spo <- tm\_map(docs.spo, removeWords, c("said","also","however","will","last","just","can"))

## Warning in tm\_map.SimpleCorpus(docs.spo, removeWords, c("said", "also", :  
## transformation drops documents

docs.spo <- tm\_map(docs.spo, stripWhitespace)

## Warning in tm\_map.SimpleCorpus(docs.spo, stripWhitespace): transformation  
## drops documents

#inspect(docs.spo[1:3])  
  
dtm.spo <- TermDocumentMatrix(docs.spo)  
#inspect(dtm.ent)  
m.spo <- as.matrix(dtm.spo)  
v.spo <- sort(rowSums(m.spo),decreasing=TRUE)  
d.spo <- data.frame(word = names(v.spo),freq=v.spo)  
  
topword.spo<-head(d.spo,10)  
topword.spo

## word freq  
## game game 478  
## england england 459  
## first first 437  
## win win 416  
## world world 379  
## one one 365  
## two two 352  
## time time 329  
## back back 321  
## players players 307

wordcloud(words = d.spo$word, freq = d.spo$freq, min.freq = 1,  
 max.words=40,scale=c(3,0.2), random.order=FALSE, rot.per=0.1,   
 colors=brewer.pal(8, "Dark2"))



#Wordcloud for Technology  
docs.tec <- Corpus(VectorSource(t(df.tech$text)))  
#inspect(docs.tec[1:3])  
  
docs.tec <- tm\_map(docs.tec, removeNumbers)

## Warning in tm\_map.SimpleCorpus(docs.tec, removeNumbers): transformation  
## drops documents

docs.tec <- tm\_map(docs.tec, removeWords, stopwords("english"))

## Warning in tm\_map.SimpleCorpus(docs.tec, removeWords,  
## stopwords("english")): transformation drops documents

docs.tec <- tm\_map(docs.tec, removePunctuation)

## Warning in tm\_map.SimpleCorpus(docs.tec, removePunctuation): transformation  
## drops documents

docs.tec <- tm\_map(docs.tec, tolower)

## Warning in tm\_map.SimpleCorpus(docs.tec, tolower): transformation drops  
## documents

docs.tec <- tm\_map(docs.tec, removeWords, c("said","also","however","will","last","just","can"))

## Warning in tm\_map.SimpleCorpus(docs.tec, removeWords, c("said", "also", :  
## transformation drops documents

docs.tec <- tm\_map(docs.tec, stripWhitespace)

## Warning in tm\_map.SimpleCorpus(docs.tec, stripWhitespace): transformation  
## drops documents

#inspect(docs.pol[1:3])  
  
dtm.tec <- TermDocumentMatrix(docs.tec)  
#inspect(dtm.tec)  
m.tec <- as.matrix(dtm.tec)  
v.tec <- sort(rowSums(m.tec),decreasing=TRUE)  
d.tec <- data.frame(word = names(v.tec),freq=v.tec)  
  
topword.tec<-head(d.tec,10)  
topword.tec

## word freq  
## people people 960  
## new new 517  
## one one 509  
## technology technology 504  
## mobile mobile 467  
## users users 407  
## games games 400  
## music music 385  
## use use 379  
## digital digital 373

wordcloud(words = d.tec$word, freq = d.tec$freq, min.freq = 1,  
 max.words=40,scale=c(4,0.3), random.order=FALSE, rot.per=0.1,   
 colors=brewer.pal(8, "Dark2"))



PREDICTIVE MODELLING:

We will be using Naive Bayes, Decision Tree, Random Forest and an Ensemble of classifiers using majority vote to get predictive results. We will be splitting the dataset in 70% train and 30% test set. We will make sure that proportion between sets are maintained by using stratified partitioning.

#split dataset, corpus, dtm to training and testing set 70/30 stratified split  
  
  
#install.packages("caret")  
library(caret)

## Loading required package: lattice

library(tm)  
  
#Corpus and DTM for df dataframe  
df.corpus <- Corpus(VectorSource(df$text))  
#inspect(df.corpus[1:1])  
df.corpus<-tm\_map(df.corpus,tolower)

## Warning in tm\_map.SimpleCorpus(df.corpus, tolower): transformation drops  
## documents

df.corpus<-tm\_map(df.corpus,removeNumbers)

## Warning in tm\_map.SimpleCorpus(df.corpus, removeNumbers): transformation  
## drops documents

df.corpus<-tm\_map(df.corpus,removePunctuation)

## Warning in tm\_map.SimpleCorpus(df.corpus, removePunctuation):  
## transformation drops documents

df.corpus<-tm\_map(df.corpus,removeWords,stopwords("english"))

## Warning in tm\_map.SimpleCorpus(df.corpus, removeWords,  
## stopwords("english")): transformation drops documents

df.corpus<-tm\_map(df.corpus,stripWhitespace)

## Warning in tm\_map.SimpleCorpus(df.corpus, stripWhitespace): transformation  
## drops documents

#inspect(df.corpus[1:1])  
  
  
#document term matrix for df  
df.dtm <- DocumentTermMatrix(df.corpus)  
#inspect(df.dtm[1:2,1:10])  
  
  
  
#Partitin df to 70/30 split (Train/Test)  
set.seed(1)  
index = createDataPartition(df$category, times=1, p=0.7, list = FALSE)  
index[1:10]

## [1] 1 3 5 6 7 8 9 10 11 14

#DATASET for training and testing  
df.train <- df[index,]  
df.test <- df[-index,]  
dim(df.train)

## [1] 1559 2

dim(df.test)

## [1] 666 2

dim(df)

## [1] 2225 2

prop.table(table(df$category))

##   
## business entertainment politics sport tech   
## 0.2292135 0.1734831 0.1874157 0.2296629 0.1802247

#check proportion for train and test  
prop.table(table(df.train$category))

##   
## business entertainment politics sport tech   
## 0.2289929 0.1738294 0.1872996 0.2296344 0.1802437

prop.table(table(df.test$category))

##   
## business entertainment politics sport tech   
## 0.2297297 0.1726727 0.1876877 0.2297297 0.1801802

1. Naive Bayes Classifier

# Naive Bayes   
  
#install.packages("tm")  
library (tm)  
  
# Create Corpus and DTM for training and testing sets   
#Corpus for df.train  
df.train.corpus <- Corpus(VectorSource(df.train$text))  
#inspect(df.train.corpus[1:1])  
df.train.corpus<-tm\_map(df.train.corpus,tolower)

## Warning in tm\_map.SimpleCorpus(df.train.corpus, tolower): transformation  
## drops documents

df.train.corpus<-tm\_map(df.train.corpus,removeNumbers)

## Warning in tm\_map.SimpleCorpus(df.train.corpus, removeNumbers):  
## transformation drops documents

df.train.corpus<-tm\_map(df.train.corpus,removePunctuation)

## Warning in tm\_map.SimpleCorpus(df.train.corpus, removePunctuation):  
## transformation drops documents

df.train.corpus<-tm\_map(df.train.corpus,removeWords,stopwords("english"))

## Warning in tm\_map.SimpleCorpus(df.train.corpus, removeWords,  
## stopwords("english")): transformation drops documents

df.train.corpus<-tm\_map(df.train.corpus,stripWhitespace)

## Warning in tm\_map.SimpleCorpus(df.train.corpus, stripWhitespace):  
## transformation drops documents

#inspect(df.train.corpus[1:1])  
  
  
#document term matrix for df.train  
df.train.dtm <- DocumentTermMatrix(df.train.corpus)  
#inspect(df.train.dtm[1:2,1:10])  
  
  
#Corpus for df.test  
df.test.corpus <- Corpus(VectorSource(df.test$text))  
#inspect(df.test.corpus[1:1])  
df.test.corpus<-tm\_map(df.test.corpus,tolower)

## Warning in tm\_map.SimpleCorpus(df.test.corpus, tolower): transformation  
## drops documents

df.test.corpus<-tm\_map(df.test.corpus,removeNumbers)

## Warning in tm\_map.SimpleCorpus(df.test.corpus, removeNumbers):  
## transformation drops documents

df.test.corpus<-tm\_map(df.test.corpus,removePunctuation)

## Warning in tm\_map.SimpleCorpus(df.test.corpus, removePunctuation):  
## transformation drops documents

df.test.corpus<-tm\_map(df.test.corpus,removeWords,stopwords("english"))

## Warning in tm\_map.SimpleCorpus(df.test.corpus, removeWords,  
## stopwords("english")): transformation drops documents

df.test.corpus<-tm\_map(df.test.corpus,stripWhitespace)

## Warning in tm\_map.SimpleCorpus(df.test.corpus, stripWhitespace):  
## transformation drops documents

#inspect(df.test.corpus[1:1])  
  
  
#document term matrix for df.train  
df.test.dtm <- DocumentTermMatrix(df.test.corpus)  
#inspect(df.dtm[1:2,1:10])  
  
  
  
  
#Find Frequent Terms  
frequent\_words <- findFreqTerms(x=df.train.dtm,lowfreq = 10)  
str(frequent\_words)

## chr [1:5251] "according" "adam" "added" "advertising" "adverts" ...

#Reduce document term matrix with only frequent words  
df.train.dtm.mod <- DocumentTermMatrix(df.train.corpus, control=list(dictionary=frequent\_words))  
str(df.train.dtm.mod)

## List of 6  
## $ i : int [1:192608] 1 1 1 1 1 1 1 1 1 1 ...  
## $ j : int [1:192608] 1 2 3 4 5 6 7 8 9 10 ...  
## $ v : num [1:192608] 1 1 1 2 1 1 1 1 3 1 ...  
## $ nrow : int 1559  
## $ ncol : int 5251  
## $ dimnames:List of 2  
## ..$ Docs : chr [1:1559] "1" "2" "3" "4" ...  
## ..$ Terms: chr [1:5251] "according" "adam" "added" "advertising" ...  
## - attr(\*, "class")= chr [1:2] "DocumentTermMatrix" "simple\_triplet\_matrix"  
## - attr(\*, "weighting")= chr [1:2] "term frequency" "tf"

df.test.dtm.mod <- DocumentTermMatrix(df.test.corpus, control=list(dictionary=frequent\_words))  
str(df.test.dtm.mod)

## List of 6  
## $ i : int [1:78955] 1 1 1 1 1 1 1 1 1 1 ...  
## $ j : int [1:78955] 1 2 3 4 5 6 7 8 9 10 ...  
## $ v : num [1:78955] 5 1 1 2 1 1 1 1 1 1 ...  
## $ nrow : int 666  
## $ ncol : int 5251  
## $ dimnames:List of 2  
## ..$ Docs : chr [1:666] "1" "2" "3" "4" ...  
## ..$ Terms: chr [1:5251] "accounting" "accused" "admission" "admitted" ...  
## - attr(\*, "class")= chr [1:2] "DocumentTermMatrix" "simple\_triplet\_matrix"  
## - attr(\*, "weighting")= chr [1:2] "term frequency" "tf"

#Function to convert word frequncies if 1 or greater to YES(presence) or zero to NO (absence)  
yes\_no <- function(x) {  
 y <- ifelse(x>0, 1, 0)  
 y <- factor (y, levels= c(0, 1), labels = c("No","Yes"))  
 y  
}   
  
  
#Convert document term matrices to Yes or NO(categorical) to be used in Naive Bayes Classifier   
df.train.dtm.mod.yn <- apply(df.train.dtm.mod,2,yes\_no)  
#str(df.train.dtm.mod.yn)  
dim(df.train.dtm.mod.yn)

## [1] 1559 5251

df.test.dtm.mod.yn <- apply(df.test.dtm.mod, 2, yes\_no)  
#str(df.test.dtm.mod.yn)  
dim(df.test.dtm.mod.yn)

## [1] 666 5251

#install naiveBayes algorithm classifier  
#install.packages("e1071")  
library(e1071)  
  
model\_nb <- naiveBayes(df.train.dtm.mod.yn,df.train$category, laplace = 1)  
class(model\_nb)

## [1] "naiveBayes"

#str(model\_nb)  
predict.nb <-predict(model\_nb,newdata = df.test.dtm.mod.yn)  
#str(predict.nb)  
  
#confusion matrix   
cm.nb <- table(predict.nb,df.test$category)  
  
#Manual computation for accuracy, recall, precision and fmeasure (Key Metrics)  
cm.nb <- as.matrix( table(predict.nb,df.test$category))  
n.nb <- sum(cm.nb) # sum of instances  
nc.nb <- nrow(cm.nb) #number of classes/categories  
diag.nb <- diag(cm.nb) # number of correctly classified instances per class  
rowsums.nb <- apply(cm.nb, 1, sum) #number of predictions per class   
colsums.nb <- apply(cm.nb, 2, sum) #number of instances per class  
p.nb <- rowsums.nb/n.nb # distribution of instances over predicted classes  
q.nb <- colsums.nb/n.nb # distribution of instances over actual classes  
  
#computation for Accuracy  
accuracy.nb <- sum(diag.nb)/n.nb  
  
#computation for precison, recall, and fmeasure per class  
  
precision.nb <- diag.nb/rowsums.nb  
recall.nb <- diag.nb /colsums.nb  
fm.nb <- 2\*precision.nb \* recall.nb /(precision.nb + recall.nb)  
  
#computation for precison, recall, and fmeasure per class as Macro-averaged Metrics  
macroPrecision.nb <- mean(precision.nb)  
macroRecall.nb <- mean(recall.nb)  
macroFmeasure.nb <- mean(fm.nb)  
  
keyMetrics.nb <- data.frame(macroPrecision.nb,macroRecall.nb,macroFmeasure.nb, accuracy.nb)  
  
  
#prop.table(table(predict.nb,df.test$category))  
  
#presenting in cross Table for naive bayes tree  
#install.packages("gmodels")  
library(gmodels)  
  
#CrossTable(predict.nb,df.test$category, prop.chisq = F, prop.t = F, dnn=c("Predicted","Actual"))  
  
confusionMatrix(predict.nb,df.test$category, positive = "1")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction business entertainment politics sport tech  
## business 149 2 8 1 4  
## entertainment 0 107 3 1 4  
## politics 1 2 113 0 1  
## sport 0 0 1 151 0  
## tech 3 4 0 0 111  
##   
## Overall Statistics  
##   
## Accuracy : 0.9474   
## 95% CI : (0.9277, 0.9631)  
## No Information Rate : 0.2297   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.934   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: business Class: entertainment Class: politics  
## Sensitivity 0.9739 0.9304 0.9040  
## Specificity 0.9708 0.9855 0.9926  
## Pos Pred Value 0.9085 0.9304 0.9658  
## Neg Pred Value 0.9920 0.9855 0.9781  
## Prevalence 0.2297 0.1727 0.1877  
## Detection Rate 0.2237 0.1607 0.1697  
## Detection Prevalence 0.2462 0.1727 0.1757  
## Balanced Accuracy 0.9723 0.9580 0.9483  
## Class: sport Class: tech  
## Sensitivity 0.9869 0.9250  
## Specificity 0.9981 0.9872  
## Pos Pred Value 0.9934 0.9407  
## Neg Pred Value 0.9961 0.9836  
## Prevalence 0.2297 0.1802  
## Detection Rate 0.2267 0.1667  
## Detection Prevalence 0.2282 0.1772  
## Balanced Accuracy 0.9925 0.9561

#key metrics, sensitivity = recall, Pos pred value = precision  
  
  
#cannot perforn wilcox.test since values are categorical

1. Decision Tree

# DECISION TREE  
  
#install.packages("rpart")  
library(rpart)  
  
#remove sparse terms  
df.dtm2 <- df.dtm  
inspect(df.dtm2)

## <<DocumentTermMatrix (documents: 2225, terms: 29989)>>  
## Non-/sparse entries: 335142/66390383  
## Sparsity : 99%  
## Maximal term length: 31  
## Weighting : term frequency (tf)  
## Sample :  
## Terms  
## Docs also can first last new one people said will year  
## 1515 6 3 12 0 0 5 3 3 2 0  
## 1605 4 7 0 5 8 11 11 10 14 0  
## 1616 2 2 3 25 1 7 11 1 18 3  
## 1928 1 0 10 0 0 6 0 0 1 0  
## 228 1 1 14 0 34 2 0 2 0 1  
## 409 4 16 2 3 18 16 25 2 28 1  
## 483 0 19 2 1 2 7 15 2 5 3  
## 678 3 15 5 5 2 7 26 11 40 8  
## 866 3 8 2 1 4 3 18 2 0 2  
## 881 1 8 4 0 10 6 7 1 11 0

df.dtm2.RemSparse <-removeSparseTerms(df.dtm2, 0.995)  
  
#convert to dataframe  
df.dtm2.RemSparse.df <- as.data.frame(as.matrix(df.dtm2.RemSparse))  
colnames(df.dtm2.RemSparse.df) <- make.names(colnames(df.dtm2.RemSparse.df))  
  
#add column category to dataframe and populate it from df column category  
df.dtm2.RemSparse.df$category <- df$category  
  
#partition dataframe to 70/30 (train/test)  
set.seed(1)  
index = createDataPartition(df.dtm2.RemSparse.df$category, times=1, p=0.7, list = FALSE)  
index[1:10]

## [1] 1 3 5 6 7 8 9 10 11 14

#DATASET for training and testing  
df.train2 <- df.dtm2.RemSparse.df[index,]  
df.test2 <- df.dtm2.RemSparse.df[-index,]  
  
#check dimensions and proportion of train and test dataframe  
dim(df.train2)

## [1] 1559 4780

dim(df.test2)

## [1] 666 4780

#prop.table(table(df.train2$category))  
#prop.table(table(df.test2$category))  
  
  
#create model  
model\_tree <- rpart(category~., data=df.train2,method = "class", minbucket=50)  
#prp(model\_tree)  
  
#plotting decision tree  
#install.packages("rpart.plot")  
library(rpart.plot)  
  
  
predict.dtree <- predict(model\_tree, newdata = df.test2, type = "class")  
  
  
table(predict.dtree,df.test2$category)

##   
## predict.dtree business entertainment politics sport tech  
## business 136 25 33 28 30  
## entertainment 2 75 0 0 28  
## politics 6 3 85 0 1  
## sport 7 12 7 125 8  
## tech 2 0 0 0 53

#prop.table(table(predict.dtree,df.test2$category))  
  
  
  
#confusion matrix   
cm.dt <- table(predict.dtree,df.test2$category)  
  
#Manual computation for accuracy, recall, precision and fmeasure (Key Metrics)  
cm.dt <- as.matrix( table(predict.dtree,df.test2$category))  
n.dt <- sum(cm.dt) # sum of instances  
nc.dt <- nrow(cm.dt) #number of classes/categories  
diag.dt <- diag(cm.dt) # number of correctly classified instances per class  
rowsums.dt <- apply(cm.dt, 1, sum) #number of predictions per class   
colsums.dt <- apply(cm.dt, 2, sum) #number of instances per class  
p.nb <- rowsums.dt/n.dt # distribution of instances over predicted classes  
q.nb <- colsums.dt/n.dt # distribution of instances over actual classes  
  
#computation for overall Accuracy  
accuracy.dt <- sum(diag.dt)/n.dt  
  
#computation for precison, recall, and fmeasure per class  
  
precision.dt <- diag.dt/rowsums.dt  
recall.dt <- diag.dt /colsums.dt  
fm.dt <- 2\*precision.dt \* recall.dt /(precision.dt + recall.dt)  
  
#computation for precison, recall, and fmeasure per class as Macro-averaged Metrics  
macroPrecision.dt <- mean(precision.dt)  
macroRecall.dt <- mean(recall.dt)  
macroFmeasure.dt <- mean(fm.dt)  
  
keyMetrics.dt <- data.frame(macroPrecision.dt,macroRecall.dt,macroFmeasure.dt, accuracy.dt)  
  
  
#presenting in cross Table for decision tree  
#install.packages("gmodels")  
library(gmodels)  
  
#CrossTable(predict.dtree,df.test2$category, prop.chisq = F, prop.t = F, dnn=c("Predicted","Actual"))  
  
confusionMatrix(predict.dtree,df.test2$category, positive = "1")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction business entertainment politics sport tech  
## business 136 25 33 28 30  
## entertainment 2 75 0 0 28  
## politics 6 3 85 0 1  
## sport 7 12 7 125 8  
## tech 2 0 0 0 53  
##   
## Overall Statistics  
##   
## Accuracy : 0.7117   
## 95% CI : (0.6757, 0.7459)  
## No Information Rate : 0.2297   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6348   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: business Class: entertainment Class: politics  
## Sensitivity 0.8889 0.6522 0.6800  
## Specificity 0.7739 0.9456 0.9815  
## Pos Pred Value 0.5397 0.7143 0.8947  
## Neg Pred Value 0.9589 0.9287 0.9299  
## Prevalence 0.2297 0.1727 0.1877  
## Detection Rate 0.2042 0.1126 0.1276  
## Detection Prevalence 0.3784 0.1577 0.1426  
## Balanced Accuracy 0.8314 0.7989 0.8308  
## Class: sport Class: tech  
## Sensitivity 0.8170 0.44167  
## Specificity 0.9337 0.99634  
## Pos Pred Value 0.7862 0.96364  
## Neg Pred Value 0.9448 0.89034  
## Prevalence 0.2297 0.18018  
## Detection Rate 0.1877 0.07958  
## Detection Prevalence 0.2387 0.08258  
## Balanced Accuracy 0.8754 0.71900

1. Random Forest

# RANDOM FOREST  
  
#install.packages("randomForest")  
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

df.train3 <-df.train2  
df.test3 <-df.test2  
  
#model\_rforest <- randomForest(category~.,data = df.train3, ntree=10, nodesize=5)   
model\_rforest <- randomForest(category~.,data = df.train3, ntree=50, nodesize=10) #might be the best option  
#model\_rforest <- randomForest(category~.,data = df.train3, ntree=100, nodesize=10)   
  
predict.rforest <- predict(model\_rforest, newdata = df.test3)  
  
table(predict.rforest,df.test3$category)

##   
## predict.rforest business entertainment politics sport tech  
## business 145 2 6 0 4  
## entertainment 1 103 0 1 3  
## politics 3 1 117 0 1  
## sport 1 6 2 152 1  
## tech 3 3 0 0 111

#prop.table(table(predict.rforest,df.test3$category))  
  
  
cm.rf <- table(predict.rforest,df.test3$category)  
  
#Manual computation for accuracy, recall, precision and fmeasure (Key Metrics)  
cm.rf <- as.matrix( table(predict.rforest,df.test3$category))  
n.rf <- sum(cm.rf) # sum of instances  
nc.rf <- nrow(cm.rf) #number of classes/categories  
diag.rf <- diag(cm.rf) # number of correctly classified instances per class  
rowsums.rf <- apply(cm.rf, 1, sum) #number of predictions per class   
colsums.rf <- apply(cm.rf, 2, sum) #number of instances per class  
p.rf <- rowsums.rf/n.rf # distribution of instances over predicted classes  
q.rf <- colsums.rf/n.rf # distribution of instances over actual classes  
  
#computation for overall Accuracy  
accuracy.rf <- sum(diag.rf)/n.rf  
  
#computation for precison, recall, and fmeasure per class  
  
precision.rf <- diag.rf/rowsums.rf  
recall.rf <- diag.rf /colsums.rf  
fm.rf <- 2\*precision.rf \* recall.rf /(precision.rf + recall.rf)  
  
#computation for precison, recall, and fmeasure per class as Macro-averaged Metrics  
macroPrecision.rf <- mean(precision.rf)  
macroRecall.rf <- mean(recall.rf)  
macroFmeasure.rf <- mean(fm.rf)  
  
keyMetrics.rf <- data.frame(macroPrecision.rf,macroRecall.rf,macroFmeasure.rf, accuracy.rf)  
  
  
#CrossTable(predict.rforest,df.test3$category, prop.chisq = F, prop.t = F, dnn=c("Predicted","Actual"))  
  
confusionMatrix(predict.rforest,df.test3$category, positive = "1")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction business entertainment politics sport tech  
## business 145 2 6 0 4  
## entertainment 1 103 0 1 3  
## politics 3 1 117 0 1  
## sport 1 6 2 152 1  
## tech 3 3 0 0 111  
##   
## Overall Statistics  
##   
## Accuracy : 0.9429   
## 95% CI : (0.9225, 0.9593)  
## No Information Rate : 0.2297   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9283   
## Mcnemar's Test P-Value : 0.3538   
##   
## Statistics by Class:  
##   
## Class: business Class: entertainment Class: politics  
## Sensitivity 0.9477 0.8957 0.9360  
## Specificity 0.9766 0.9909 0.9908  
## Pos Pred Value 0.9236 0.9537 0.9590  
## Neg Pred Value 0.9843 0.9785 0.9853  
## Prevalence 0.2297 0.1727 0.1877  
## Detection Rate 0.2177 0.1547 0.1757  
## Detection Prevalence 0.2357 0.1622 0.1832  
## Balanced Accuracy 0.9622 0.9433 0.9634  
## Class: sport Class: tech  
## Sensitivity 0.9935 0.9250  
## Specificity 0.9805 0.9890  
## Pos Pred Value 0.9383 0.9487  
## Neg Pred Value 0.9980 0.9836  
## Prevalence 0.2297 0.1802  
## Detection Rate 0.2282 0.1667  
## Detection Prevalence 0.2432 0.1757  
## Balanced Accuracy 0.9870 0.9570

1. Ensemble learning by Majority Vote (Naive Bayes, Decision Tree, Random Forest)

# Ensemble1 of models (NB,DTree, RandomForest) by majority vote  
  
df.ensemble <- df.test  
df.ensemble$NBpredict <- as.factor(predict.nb)  
df.ensemble$DTreepredict <- as.factor(predict.dtree)  
df.ensemble$RFpredict <- as.factor(predict.rforest)  
  
count\_predict <- df.ensemble[,-c(1:2)]  
  
   
#category count: business, entertainment, politics ,sport,tech   
count\_business <- count\_predict  
count\_business$Countbus <- rowSums(count\_business=="business")  
count\_business<-count\_business[,-c(1:3)]  
  
  
count\_entertainment <- count\_predict  
count\_entertainment$CountEnt <- rowSums(count\_entertainment=="entertainment")  
count\_entertainment <- count\_entertainment[,-c(1:3)]  
  
count\_politics <- count\_predict  
count\_politics$CountPol <- rowSums(count\_politics=="politics")  
count\_politics <- count\_politics[,-c(1:3)]  
  
count\_sport <- count\_predict  
count\_sport$CountSports <- rowSums(count\_sport=="sport")  
count\_sport <- count\_sport[,-c(1:3)]  
  
count\_tech <- count\_predict  
count\_tech$CountTech <- rowSums(count\_tech=="tech")  
count\_tech <- count\_tech[,-c(1:3)]  
  
  
#Put all counts together  
   
count\_max <- data.frame("business"=count\_business, "entertainment"=count\_entertainment,  
 "politics"=count\_politics, "sport"=count\_sport,  
 "tech"=count\_tech)  
#get prediction for ensemble by using highest count as prediction  
count\_max$predict <- colnames(count\_max)[apply(count\_max,1,which.max)]  
  
table(count\_max$predict,df.ensemble$category)

##   
## business entertainment politics sport tech  
## business 150 7 8 1 6  
## entertainment 0 107 0 1 5  
## politics 1 0 116 0 1  
## sport 0 1 1 151 1  
## tech 2 0 0 0 107

#prop.table(table(count\_max$predict,df.ensemble$category))  
cm.en <- table(count\_max$predict,df.ensemble$category)  
  
#Manual computation for accuracy, recall, precision and fmeasure (Key Metrics)  
cm.en <- as.matrix(table(count\_max$predict,df.ensemble$category))  
n.en <- sum(cm.en) # sum of instances  
nc.en <- nrow(cm.en) #number of classes/categories  
diag.en <- diag(cm.en) # number of correctly classified instances per class  
rowsums.en <- apply(cm.en, 1, sum) #number of predictions per class   
colsums.en <- apply(cm.en, 2, sum) #number of instances per class  
p.en <- rowsums.en/n.en # distribution of instances over predicted classes  
q.en <- colsums.en/n.en # distribution of instances over actual classes  
  
#computation for overall Accuracy  
accuracy.en <- sum(diag.en)/n.en  
  
#computation for precison, recall, and fmeasure per class  
  
precision.en <- diag.en/rowsums.en  
recall.en <- diag.en /colsums.en  
fm.en <- 2\*precision.en \* recall.en /(precision.en + recall.en)  
  
#computation for precison, recall, and fmeasure per class as Macro-averaged Metrics  
macroPrecision.en <- mean(precision.en)  
macroRecall.en <- mean(recall.en)  
macroFmeasure.en <- mean(fm.en)  
  
keyMetrics.en <- data.frame(macroPrecision.en,macroRecall.en,macroFmeasure.en, accuracy.en)  
  
  
  
#CrossTable(count\_max$predict,df.ensemble$category, prop.chisq = F, prop.t = F, dnn=c("Predicted","Actual"))  
  
count\_max$predict <- as.factor(count\_max$predict)  
  
confusionMatrix(count\_max$predict,df.ensemble$category, positive = "1")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction business entertainment politics sport tech  
## business 150 7 8 1 6  
## entertainment 0 107 0 1 5  
## politics 1 0 116 0 1  
## sport 0 1 1 151 1  
## tech 2 0 0 0 107  
##   
## Overall Statistics  
##   
## Accuracy : 0.9474   
## 95% CI : (0.9277, 0.9631)  
## No Information Rate : 0.2297   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9339   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: business Class: entertainment Class: politics  
## Sensitivity 0.9804 0.9304 0.9280  
## Specificity 0.9571 0.9891 0.9963  
## Pos Pred Value 0.8721 0.9469 0.9831  
## Neg Pred Value 0.9939 0.9855 0.9836  
## Prevalence 0.2297 0.1727 0.1877  
## Detection Rate 0.2252 0.1607 0.1742  
## Detection Prevalence 0.2583 0.1697 0.1772  
## Balanced Accuracy 0.9688 0.9598 0.9622  
## Class: sport Class: tech  
## Sensitivity 0.9869 0.8917  
## Specificity 0.9942 0.9963  
## Pos Pred Value 0.9805 0.9817  
## Neg Pred Value 0.9961 0.9767  
## Prevalence 0.2297 0.1802  
## Detection Rate 0.2267 0.1607  
## Detection Prevalence 0.2312 0.1637  
## Balanced Accuracy 0.9905 0.9440

1. Ensemble learning by Majority Vote (Naive Bayes, Random Forest)

# Ensemble2 of models (NB, RandomForest) by majority vote  
  
df.ensemble2 <- df.test  
df.ensemble2$NBpredict <- as.factor(predict.nb)  
df.ensemble2$RFpredict <- as.factor(predict.rforest)  
  
count\_predict2 <- df.ensemble2[,-c(1:2)]  
  
#category count: business, entertainment, politics ,sport,tech   
count\_business2 <- count\_predict2  
count\_business2$Countbus <- rowSums(count\_business2=="business")  
count\_business2<-count\_business2[,-c(1:2)]  
  
count\_entertainment2 <- count\_predict2  
count\_entertainment2$CountEnt <- rowSums(count\_entertainment2=="entertainment")  
count\_entertainment2 <- count\_entertainment2[,-c(1:2)]  
  
count\_politics2 <- count\_predict2  
count\_politics2$CountPol <- rowSums(count\_politics2=="politics")  
count\_politics2 <- count\_politics2[,-c(1:2)]  
  
count\_sport2 <- count\_predict2  
count\_sport2$CountSports <- rowSums(count\_sport2=="sport")  
count\_sport2 <- count\_sport2[,-c(1:2)]  
  
count\_tech2 <- count\_predict2  
count\_tech2$CountTech <- rowSums(count\_tech2=="tech")  
count\_tech2 <- count\_tech2[,-c(1:2)]  
  
#Put all counts together  
  
count\_max2 <- data.frame("business"=count\_business2, "entertainment"=count\_entertainment2,  
 "politics"=count\_politics2, "sport"=count\_sport2,  
 "tech"=count\_tech2)  
  
#get prediction for ensemble by using highest count as prediction  
count\_max2$predict <- colnames(count\_max2)[apply(count\_max2,1,which.max)]  
  
table(count\_max2$predict,df.ensemble2$category)

##   
## business entertainment politics sport tech  
## business 150 2 11 1 6  
## entertainment 0 110 3 1 5  
## politics 1 1 110 0 1  
## sport 0 2 1 151 1  
## tech 2 0 0 0 107

#prop.table(table(count\_max2$predict,df.ensemble2$category))  
  
  
  
cm.en2 <- table(count\_max2$predict, df.ensemble2$category)  
  
#Manual computation for accuracy, recall, precision and fmeasure (Key Metrics)  
cm.en2 <- as.matrix(table(count\_max2$predict, df.ensemble2$category))  
n.en2 <- sum(cm.en2) # sum of instances  
nc.en2 <- nrow(cm.en2) #number of classes/categories  
diag.en2 <- diag(cm.en2) # number of correctly classified instances per class  
rowsums.en2 <- apply(cm.en2, 1, sum) #number of predictions per class   
colsums.en2 <- apply(cm.en2, 2, sum) #number of instances per class  
p.en <- rowsums.en2/n.en2 # distribution of instances over predicted classes  
q.en <- colsums.en2/n.en2 # distribution of instances over actual classes  
  
#computation for overall Accuracy  
accuracy.en2 <- sum(diag.en2)/n.en2  
  
#computation for precison, recall, and fmeasure per class  
  
precision.en2 <- diag.en2/rowsums.en2  
recall.en2 <- diag.en2 /colsums.en2  
fm.en2 <- 2\*precision.en2 \* recall.en2 /(precision.en2 + recall.en2)  
  
#computation for precison, recall, and fmeasure per class as Macro-averaged Metrics  
macroPrecision.en2 <- mean(precision.en2)  
macroRecall.en2 <- mean(recall.en2)  
macroFmeasure.en2 <- mean(fm.en2)  
  
keyMetrics.en2 <- data.frame(macroPrecision.en2,macroRecall.en2,macroFmeasure.en2, accuracy.en2)  
  
  
#CrossTable(count\_max2$predict,df.ensemble2$category, prop.chisq = F, prop.t = F, dnn=c("Predicted","Actual"))  
  
count\_max2$predict <- as.factor(count\_max2$predict)  
  
confusionMatrix(count\_max2$predict,df.ensemble2$category, positive = "1")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction business entertainment politics sport tech  
## business 150 2 11 1 6  
## entertainment 0 110 3 1 5  
## politics 1 1 110 0 1  
## sport 0 2 1 151 1  
## tech 2 0 0 0 107  
##   
## Overall Statistics  
##   
## Accuracy : 0.9429   
## 95% CI : (0.9225, 0.9593)  
## No Information Rate : 0.2297   
## P-Value [Acc > NIR] : < 2e-16   
##   
## Kappa : 0.9283   
## Mcnemar's Test P-Value : 0.01205   
##   
## Statistics by Class:  
##   
## Class: business Class: entertainment Class: politics  
## Sensitivity 0.9804 0.9565 0.8800  
## Specificity 0.9610 0.9837 0.9945  
## Pos Pred Value 0.8824 0.9244 0.9735  
## Neg Pred Value 0.9940 0.9909 0.9729  
## Prevalence 0.2297 0.1727 0.1877  
## Detection Rate 0.2252 0.1652 0.1652  
## Detection Prevalence 0.2553 0.1787 0.1697  
## Balanced Accuracy 0.9707 0.9701 0.9372  
## Class: sport Class: tech  
## Sensitivity 0.9869 0.8917  
## Specificity 0.9922 0.9963  
## Pos Pred Value 0.9742 0.9817  
## Neg Pred Value 0.9961 0.9767  
## Prevalence 0.2297 0.1802  
## Detection Rate 0.2267 0.1607  
## Detection Prevalence 0.2327 0.1637  
## Balanced Accuracy 0.9896 0.9440

SUMMARY OF THE KEY METRICS IN DECIDNG THE BEST PREDICTIVE MODEL FOR OUR PROJECT:

#Naive bayes  
keyMetrics.nb

## macroPrecision.nb macroRecall.nb macroFmeasure.nb accuracy.nb  
## 1 0.9477765 0.9440438 0.9454638 0.9474474

#Decision Tree  
keyMetrics.dt

## macroPrecision.dt macroRecall.dt macroFmeasure.dt accuracy.dt  
## 1 0.779701 0.6959446 0.7066293 0.7117117

#Random Forest  
keyMetrics.rf

## macroPrecision.rf macroRecall.rf macroFmeasure.rf accuracy.rf  
## 1 0.9446553 0.9395657 0.9416815 0.9429429

#Ensenble by majority vote (NaiveBayes, Decision Tree, Random Forest)  
keyMetrics.en

## macroPrecision.en macroRecall.en macroFmeasure.en accuracy.en  
## 1 0.9528435 0.9434843 0.9469234 0.9474474

#Ensenble by majority vote (NaiveBayes, RandoM Forest)  
keyMetrics.en2

## macroPrecision.en2 macroRecall.en2 macroFmeasure.en2 accuracy.en2  
## 1 0.9472038 0.9391017 0.9416701 0.9429429

RESULTS AND CONCLUSION:

We used the average of precision and recall for each category to compute the f measure. With decision tree getting the lowest score for both fmeasure and accuracy. Naive Bayes gets highest score for individual classifiers with Random Forest not far behind. The ensemble classifier (NaiveBayes and Random Forest) surprisingly scored even lower than Random Forest alone. It was the combination of the three individual classifiers that garnered the highest f measure.

It is therefore an ensemble of the three individual classifiers that produced the highest score based on our designated key metrics.

Github link:

<https://github.com/Solonski/ckme136>