Sentiment Text Mining

Sam Vuong, Raymond (shanhua) Huang, Carmon Ho, Kyle Murphy

July 2020

## Introduction

The goal of this project is to be able to interpret the feelings of travellers on Twitter in February 2015. The success of this project will be determined by how well we can analyze the sentiments that were submitted to Twitter by various travellers in that period. This model can be used by airline companies who want to know the satisfaction of their customers’ travel experiences.

We will employ text mining techniques to analyze and process tweet text messages and use Random Forest classification algorithm to model and predict the sentiments tweet text messages.

Our application on ShinyApps can be accessed at: <https://skvuong.shinyapps.io/project/>

Our application code can be found in GitHub at: <https://github.com/skvuong/sentimentTextMining>

## Install and load the required packages

The following libraries are used:

if(!require(caTools))   
 install.packages("caTools")  
if(!require(dplyr))   
 install.packages("dplyr")  
if(!require(forcats))   
 install.packages("forcats")  
if(!require(ggplot2))   
 install.packages("ggplot2")  
if(!require(ggthemes))   
 install.packages("ggthemes")  
if(!require(lubridate))   
 install.packages("lubridate")  
if(!require(mgsub))   
 install.packages("mgsub")  
if(!require(RColorBrewer))   
 install.packages("RColorBrewer")  
if(!require(randomForest))   
 install.packages("randomForest")  
if(!require(scales))   
 install.packages("scales")  
if(!require(stringr))   
 install.packages("stringr")  
if(!require(SnowballC))   
 install.packages("SnowballC")  
if(!require(stopwords))   
 install.packages("stopwords")  
if(!require(syuzhet))   
 install.packages("syuzhet")  
if(!require(tm))   
 install.packages("tm")  
if(!require(wordcloud))  
 install.packages("wordcloud")  
if(!require(wordcloud2))  
 install.packages("wordcloud2")  
  
library(caTools) #data wrangling  
library(dplyr) #data manipulation  
library(forcats) #ggplot frequency  
library(ggplot2) #visualizations  
library(ggthemes) #visualizations  
library(lubridate)  
library(mgsub)  
library(RColorBrewer) #color palettes  
library(randomForest) #randomforest  
library(scales)  
library(stringr)  
library(syuzhet)  
library(SnowballC) #text stemming  
library(stopwords) #stop-words removal  
library(tm) #text mining  
library(wordcloud) #word-cloud generator  
library(wordcloud2) #word-cloud generator

## Loading Data

tweets <- read.csv("tweets.csv",header = TRUE)

## Data Exploration Analysis

#Check dataset size and column names  
dim(tweets)

## [1] 14640 15

names(tweets)

## [1] "tweet\_id" "airline\_sentiment"   
## [3] "airline\_sentiment\_confidence" "negativereason"   
## [5] "negativereason\_confidence" "airline"   
## [7] "airline\_sentiment\_gold" "name"   
## [9] "negativereason\_gold" "retweet\_count"   
## [11] "text" "tweet\_coord"   
## [13] "tweet\_created" "tweet\_location"   
## [15] "user\_timezone"

#How many unique tweets are in the dataset?  
length(unique(tweets$tweet\_id))

## [1] 14485

#How many users have tweeted?  
length(unique(tweets$name))

## [1] 7701

#Average characters per tweet?  
mean(str\_count(tweets$text))

## [1] 104.1185

#Average words per tweet?  
mean(str\_count(tweets$text, '\\s+')+1)

## [1] 17.65396

#Average tweet characters per sentiment?  
tweets$numofcharsintweet <- str\_count(tweets$text)  
aggregate(tweets$numofcharsintweet, by=list(SentID=tweets$airline\_sentiment), FUN= mean)

## SentID x  
## 1 negative 114.09370  
## 2 neutral 87.78574  
## 3 positive 86.79433

#Average tweet words per sentiment?  
tweets$numofwordsintweet <- str\_count(tweets$text, '\\s+')+1  
aggregate(tweets$numofwordsintweet, by=list(SentID=tweets$airline\_sentiment), FUN= mean)

## SentID x  
## 1 negative 19.68675  
## 2 neutral 14.40884  
## 3 positive 14.01439

#What are the types of sentiments?  
unique(tweets$airline\_sentiment)

## [1] neutral positive negative  
## Levels: negative neutral positive

#Count of sentiments?  
tweets %>% group\_by(airline\_sentiment) %>% summarize(n=n())

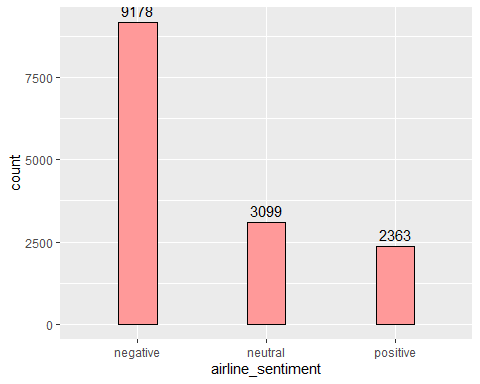
## # A tibble: 3 x 2  
## airline\_sentiment n  
## <fct> <int>  
## 1 negative 9178  
## 2 neutral 3099  
## 3 positive 2363

#Count of Negative Reasons?  
tweets %>% group\_by(negativereason) %>% summarize(n=n())

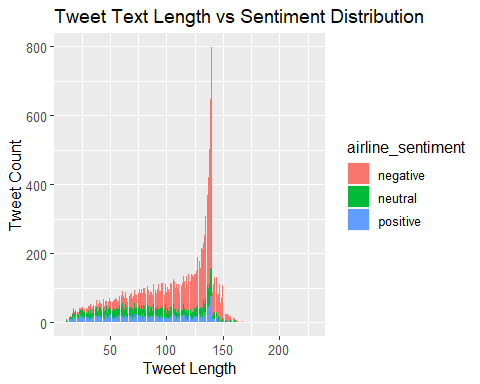
## # A tibble: 11 x 2  
## negativereason n  
## <fct> <int>  
## 1 "" 5462  
## 2 "Bad Flight" 580  
## 3 "Can't Tell" 1190  
## 4 "Cancelled Flight" 847  
## 5 "Customer Service Issue" 2910  
## 6 "Damaged Luggage" 74  
## 7 "Flight Attendant Complaints" 481  
## 8 "Flight Booking Problems" 529  
## 9 "Late Flight" 1665  
## 10 "longlines" 178  
## 11 "Lost Luggage" 724

## Data Visualization

#1. Tweet Sentiment Distribution  
ggplot(tweets, aes(x = airline\_sentiment)) +  
 geom\_bar(stat = "count", width=0.3, fill="#FF9999", colour="black") +  
 geom\_text(stat='count', aes(label=..count..), vjust=-0.5)



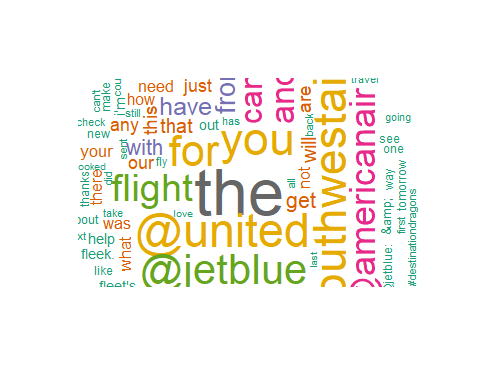
#2. Tweet Text Sentiment Scores  
#Note that get\_nrc\_sentiment() function runs very slow  
#It could be the number tweet text messages we have  
#  
#review <- as.character(tweets$text)  
#s <- get\_nrc\_sentiment(review)  
#review\_sentiment <- cbind(tweets$text,s)  
#barplot(colsums(s), col=rainbow(10),  
# ylab='count', main='Sentiment Scores for Tweets')  
  
#3. Tweet Text Length vs Sentiment Distribution  
tweets$length = nchar(as.character(tweets$text))  
ggplot(tweets, aes(x = length, fill = airline\_sentiment)) +   
 geom\_bar() +  
 scale\_x\_continuous() +  
 ggtitle("Tweet Text Length vs Sentiment Distribution") +  
 xlab("Tweet Length") +  
 ylab("Tweet Count") +  
 theme(text = element\_text(size=12))



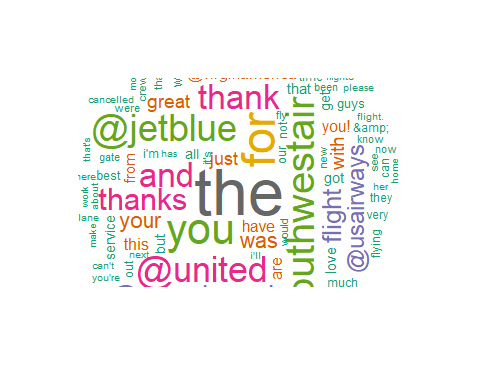
#4. Word-Cloud chart for Negative Sentiment Tweets (Top 100)  
special\_terms <- c("https\\S\*", "@\\S\*", "amp", "[\r\n]","[[:punct:]]","@united", "@VirginAmerica","@usairways","@americanair","@united","@jetblue")  
special\_text <- c("","","","","","","","","","","")  
negative <- tweets %>% filter(airline\_sentiment == "negative")  
negativetext <-negative$text  
#negativetext <- mgsub(negativetext,  
# as.character(special\_terms), as.character(special\_text))  
docsnegative <- Corpus(VectorSource(negativetext))  
dtmnegative <- TermDocumentMatrix(docsnegative)   
matrixnegative <- as.matrix(dtmnegative)   
wordsnegative <- sort(rowSums(matrixnegative),decreasing=TRUE)   
dfnegative <- data.frame(word = names(wordsnegative),freq=wordsnegative)  
set.seed(1234) # for reproducibility   
wordcloud(words = dfnegative$word, freq = dfnegative$freq, min.freq = 1,  
 max.words=100, random.order=FALSE, rot.per=0.35, colors=brewer.pal(8,"Dark2"))



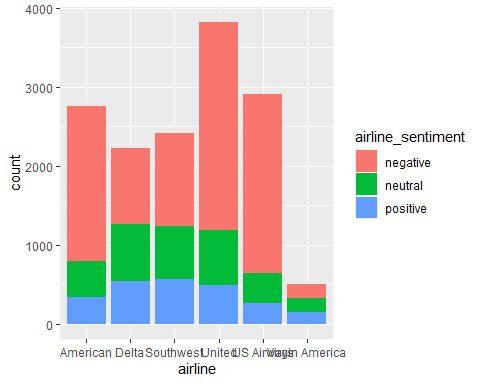
#5. Word-Cloud chart for Neutral Sentiment Tweets (Top 100)  
neutral <- tweets %>% filter(airline\_sentiment == "neutral")  
neutraltext <-neutral$text  
#neutraltext <- mgsub(negativetext,  
# as.character(special\_terms), as.character(special\_text))  
docsneutral <- Corpus(VectorSource(neutraltext))  
dtmneutral <- TermDocumentMatrix(docsneutral)   
matrixneutral <- as.matrix(dtmneutral)   
wordsneutral <- sort(rowSums(matrixneutral),decreasing=TRUE)   
dfneutral <- data.frame(word = names(wordsneutral),freq=wordsneutral)  
set.seed(1234) # for reproducibility   
wordcloud(words = dfneutral$word, freq = dfneutral$freq, min.freq = 1,  
 max.words=100, random.order=FALSE, rot.per=0.35, colors=brewer.pal(8,"Dark2"))



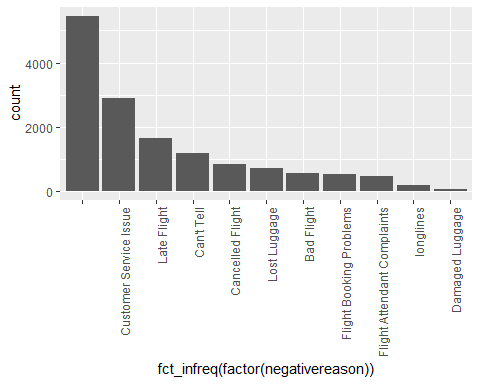
#6. Word-Cloud chart for Positive Sentiment Tweets (Top 100)  
positive <- tweets %>% filter(airline\_sentiment == "positive")  
positivetext <-positive$text  
#positivetext <- mgsub(negativetext,  
# as.character(special\_terms), as.character(special\_text))  
docspositive <- Corpus(VectorSource(positivetext))  
dtmpositive <- TermDocumentMatrix(docspositive)   
matrixpositive <- as.matrix(dtmpositive)   
wordspositive <- sort(rowSums(matrixpositive),decreasing=TRUE)   
dfpositive <- data.frame(word = names(wordspositive),freq=wordspositive)  
set.seed(1234) # for reproducibility   
wordcloud(words = dfpositive$word, freq = dfpositive$freq, min.freq = 1,  
 max.words=100, random.order=FALSE, rot.per=0.35, colors=brewer.pal(8,"Dark2"))



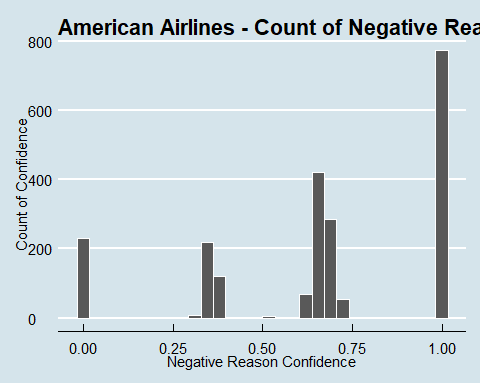
#7. Airlines Sentimental Distribution  
ggplot(tweets, aes(x = airline,fill = airline\_sentiment )) +   
 geom\_bar(stat = "count")



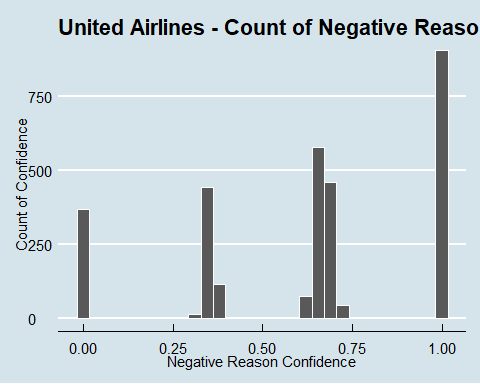
#8. Negative Tweet Reasons Distribution  
ggplot(tweets, aes(x = fct\_infreq(factor(negativereason)))) +   
 geom\_bar(stat = "count") +   
 theme(axis.text.x = element\_text(angle = 90, hjust = 1))



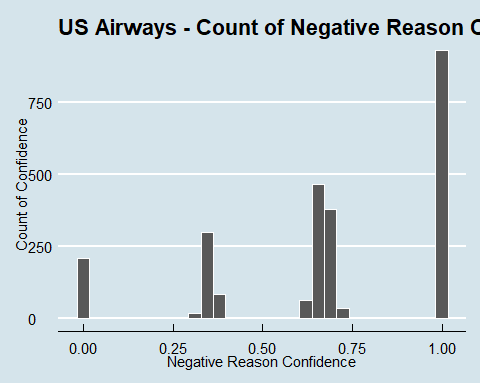
#9a. American Airline - Negative Reason confident Distribution  
American <- tweets %>% filter(airline == "American")  
American %>%   
 ggplot(aes(x= negativereason\_confidence)) +  
 geom\_histogram(color = "white") +   
 ggtitle("American Airlines - Count of Negative Reason Confidence") +  
 xlab("Negative Reason Confidence") +  
 ylab("Count of Confidence") +  
 scale\_y\_continuous(labels = comma) +   
 theme\_economist()



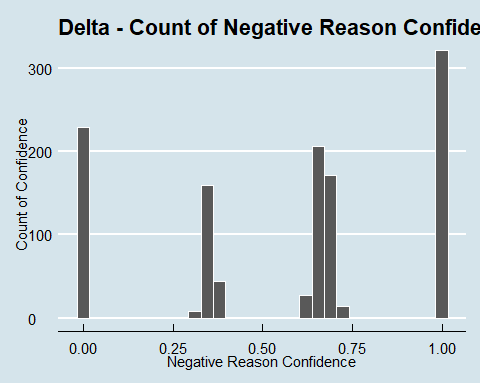
#9b. United Airline - Negative Reason confident Distribution  
United <- tweets %>% filter(airline == "United")  
United %>%  
 ggplot(aes(x= negativereason\_confidence)) +  
 geom\_histogram(color = "white") +   
 ggtitle("United Airlines - Count of Negative Reason Confidence") +  
 xlab("Negative Reason Confidence") +  
 ylab("Count of Confidence") +  
 scale\_y\_continuous(labels = comma) +   
 theme\_economist()



#9c. US Airways - Negative Reason confident Distribution  
USAirways <- tweets %>% filter(airline == "US Airways")  
USAirways %>%  
 ggplot(aes(x= negativereason\_confidence)) +  
 geom\_histogram(color = "white") +   
 ggtitle("US Airways - Count of Negative Reason Confidence") +  
 xlab("Negative Reason Confidence") +  
 ylab("Count of Confidence") +  
 scale\_y\_continuous(labels = comma) +   
 theme\_economist()



#9d. Delta Airline - Negative Reason confident Distribution  
Delta<- tweets %>% filter(airline == "Delta")  
Delta %>%  
 ggplot(aes(x= negativereason\_confidence)) +  
 geom\_histogram(color = "white") +   
 ggtitle("Delta - Count of Negative Reason Confidence") +  
 xlab("Negative Reason Confidence") +  
 ylab("Count of Confidence") +  
 scale\_y\_continuous(labels = comma) +   
 theme\_economist()



## Function: text\_to\_Corpus\_document

This function converts a text vector to a Corpus document object and perform common text tokenization, special characters/words removal and stemming

text\_to\_Corpus\_document <- function(text\_data)  
{  
 corpus\_docs <- Corpus(VectorSource(text\_data))  
   
 toSpace <- content\_transformer(function (x , pattern ) gsub(pattern, " ", x))  
 corpus\_docs <- tm\_map(corpus\_docs, toSpace, "/")  
 corpus\_docs <- tm\_map(corpus\_docs, toSpace, "@")  
 corpus\_docs <- tm\_map(corpus\_docs, toSpace, "\\|")  
   
 airline\_stopwords <- c("flight","usairway","americanair","southwestair",  
 "jetblu","virginamerica","united","delta")  
   
 corpus\_docs <- tm\_map(corpus\_docs, content\_transformer(tolower))  
 corpus\_docs <- tm\_map(corpus\_docs, removePunctuation)  
 corpus\_docs <- tm\_map(corpus\_docs, removeNumbers)  
 corpus\_docs <- tm\_map(corpus\_docs, removeWords, stopwords("english"))  
 corpus\_docs <- tm\_map(corpus\_docs, removeWords, airline\_stopwords)  
   
 corpus\_docs <- tm\_map(corpus\_docs, stripWhitespace)  
 corpus\_docs <- tm\_map(corpus\_docs, stemDocument)  
   
 return(corpus\_docs)  
}

## Data Preparation for Modeling

# Get tweet text messages and sentiment classification from tweets dataframe  
text\_data <- tweets$text  
sentiment <- tweets$airline\_sentiment  
  
# Convert text messages to Corpus document  
docs <- text\_to\_Corpus\_document(text\_data)  
  
# Convert to Document-Term Matrix  
# Remove Sparse Terms  
dtm = DocumentTermMatrix(docs)  
dtm = removeSparseTerms(dtm,sparse = 0.99)  
  
# Convert matrix to dataframe  
model\_data = as.data.frame(as.matrix(dtm))  
  
# Convert sentiment classification to factor  
sentiment <- as.character(sentiment)  
sentiment <- as.factor(sentiment)  
  
# Add sentiment classification to dataframe  
model\_data$sentiment = sentiment

## Create a Random Forest model

# Use 80/20 split for train/test sets  
set.seed(123)  
split = sample.split(sentiment,SplitRatio = 0.80)  
train = subset(model\_data,split = TRUE)  
test = subset(model\_data,split = FALSE)  
  
# Create a Random Forest model with ntree = 10  
predict\_rf = randomForest(x = train[,-length(train)], y = train$sentiment, ntree=10)  
summary(predict\_rf)

## Length Class Mode   
## call 4 -none- call   
## type 1 -none- character  
## predicted 14640 factor numeric   
## err.rate 40 -none- numeric   
## confusion 12 -none- numeric   
## votes 43920 matrix numeric   
## oob.times 14640 -none- numeric   
## classes 3 -none- character  
## importance 169 -none- numeric   
## importanceSD 0 -none- NULL   
## localImportance 0 -none- NULL   
## proximity 0 -none- NULL   
## ntree 1 -none- numeric   
## mtry 1 -none- numeric   
## forest 14 -none- list   
## y 14640 factor numeric   
## test 0 -none- NULL   
## inbag 0 -none- NULL

## Model Evaluation

# Prediction on test set  
y\_pred = predict(predict\_rf, newdata = test[,-length(test)])  
  
# Confusion matrix  
confusion\_matrix = table(test$sentiment, y\_pred)  
confusion\_matrix

## y\_pred  
## negative neutral positive  
## negative 8611 469 98  
## neutral 693 2243 163  
## positive 395 288 1680

# Accuracy  
accuracy <- sum(diag(confusion\_matrix)) / sum(confusion\_matrix)  
accuracy

## [1] 0.8561475