# Project Report

To find out non-negative posts from the entire 4555 posts, I took the following steps:

1. Find out a roadmap of selecting non-negative posts
2. Build up a complaint word bank using "complaint1700.csv" file
3. Classify whether each post is potentially a negative post.
4. Run model to test the precision of my prediction.

## Find out a roadmap of selecting non-negative posts

In the beginning, the tweets are mostly negative. One way to distinguish non-negative tweets from all the tweets is that negative tweets may contain some symbolic complaining words such as "awful", "poor", "worst" etc., and non-negative ones rarely have those words. Given that sense, I started from extracting symbolic complaining words out of the "complaint1700.csv", the training dataset, in which every post is negative. From there, I could easily obtain all the words that potentially signal a negative post. By grouping up the words as a complaint word bank, I can then use the bank to evaluate the tweets in my unique dataset.

## Build up a complaint word bank using "complaint1700.csv" file

In this step, I extracted the reoccurring adjectives, nouns, adverbials and verbs from the training dataset; those words may potentially expressed anxiety, anger, frustration, sadness, towards the airlines, or signal the actions that could lead to those negative emotions. However, this word bank is not perfect due to the existing non-negative components. I eliminated those non-negative components to make sure the word bank is ready to use for distinguishing negative posts.

## Classify whether each post is potentially a negative post

Having the complaint word bank ready, I used the bank as a dictionary that was applied to my tweets' DTM building. In this step, I intended to see whether each post of my tweets contains the complaining words from the word bank. If yes, I set that post as 0, and no as 1. All the posts that marked as 1 were the potential non-negative posts.

## Run model to test the precision of my prediction

To test the precision of my prediction, I ran an SVM model. I set the dtm (produced from original tweets) as the dataset, potential negative and non-negative judgements on the tweets as Y (1 and 0s), and all other variables except Y as independent variables. Training data is 80% of the entire dataset, and the training samples are randomly selected. The model makes very precise predictions; the precision reached 0.97.

## Result and Take-aways

After all these steps, by manually evaluating the **106** non-negative tweets extracted from the original dataset, I got a precision of **0.82243**. The actual precision was lower than the one in R. This is partially because the model wasn't smart enough to identify there was a negative meaning of a sentence that had no complaining words. This flaw should be compensated by human beings who has better semantic analysis skills.

```r
#====================================================#
#===============CIS434 FINAL PROJECT================#
#================YUKUN GAO 31616027=================#
#====================================================#
rm(list=ls())

require("SnowballC")

## Loading required package: SnowballC

require("NLP")

## Loading required package: NLP

require("openNLP")

## Loading required package: openNLP

library(tm)
library(pROC)

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

library(DBI)
library(e1071)
library(readr)
#====================================================#
# Get data from SQL server and store as local file  #
#====================================================#
# driver <- dbDriver("MySQL")
# myhost <- "localhost"
# mydb   <- "studb"
# myacct <- "cis434"
# mypwd  <- "LLhtFPbdwiJans8F@S207"
#
# conn <- dbConnect(driver, host=myhost, dbname=mydb, myacct, mypwd)
#
# temp <- dbGetQuery(conn, "SELECT * FROM proj4final WHERE tag='uN#w4
v22?Trn'")
# dbDisconnect(conn)
#
```

```r
# write.csv(temp, '~/final_df.csv', row.names = F)

#====================================================#
#                 Read datasets                  #
#====================================================#
setwd("C:/Users/yukun/OneDrive/UR Graduate/Fall B/CIS434 Social Media
 Analytics/Final Project")
complaint <- read.csv('complaint1700.csv')
data <- read.csv('final_df.csv')
post <- data$tweet
tweet <- complaint$tweet
#====================================================#
#   Extract complaint words from complaint dataset  #
#        and write them to complaint wordbank      #
#====================================================#
sent_token_annotator <- Maxent_Sent_Token_Annotator()
a1 <- annotate(tweet, sent_token_annotator)

tweet <- as.String(tweet)

word_token_annotator <- Maxent_Word_Token_Annotator()
a2 <- annotate(tweet, word_token_annotator, a1)

pos_tag_annotator <- Maxent_POS_Tag_Annotator()
a3 <- annotate(tweet, pos_tag_annotator, a2)
a3_word <- subset(a3, type == "word")
tags <- sapply(a3_word$features, "[[", "POS")
#get relevant adjectives, noun, adv, and verbs from the complaint dat
aset
mypos_adj = a3_word[tags=="JJ" | tags == 'JJS' | tags == "JJR"]
mypos_nn = a3_word[tags=="NN" | tags =='NNS']
mypos_adv = a3_word[tags=="RB"]
mypos_vb = a3_word[tags=="VB" | tags == 'VBD' | tags == "VBG" | tags
== 'VBP']
adj <- tweet[mypos_adj]
nn <- tweet[mypos_nn]
adv <- tweet[mypos_adv]
vb <- tweet[mypos_vb]

adj_words <-Corpus(VectorSource(adj))
nn_words <- Corpus(VectorSource(nn))
adv_words <- Corpus(VectorSource(adv))
vb_words <- Corpus(VectorSource(vb))
```

```r
dtm_adj.full <- DocumentTermMatrix(adj_words, control=list(tolower=
T,removePunctuation=T, removeNumbers = T))
dtm_adj <- removeSparseTerms(dtm_adj.full, 0.999)
dtm_nn.full <- DocumentTermMatrix(nn_words, control=list(tolower=T,r
emovePunctuation=T, removeNumbers = T))
dtm_nn <- removeSparseTerms(dtm_nn.full, 0.999)
dtm_adv.full <- DocumentTermMatrix(adv_words, control=list(tolower=
T,removePunctuation=T, removeNumbers = T))
dtm_adv <- removeSparseTerms(dtm_adv.full, 0.999)
dtm_vb.full <- DocumentTermMatrix(vb_words, control=list(tolower=T,r
emovePunctuation=T, removeNumbers = T))
dtm_vb <- removeSparseTerms(dtm_vb.full, 0.999)
X_adj = as.matrix(dtm_adj)
X_nn = as.matrix(dtm_nn)
X_adv = as.matrix(dtm_adv)
X_vb = as.matrix(dtm_vb)
complaint_wordbank <- c(colnames(X_adj),colnames(X_nn),colnames(X_ad
v),colnames(X_vb))

#====================================================#
#          write complaint wordbank to txt          #
#====================================================#

fileConn1<-file("complaint.txt")
writeLines(complaint_wordbank, fileConn1)
close(fileConn1)

#==========================================================#
# classify complaining posts using the complaint wordbank #
#==========================================================#
#read in the edited wordbank (deleted non complaint words from the wo
rdbank)
complaint_wordbank <- scan('complaint_edited.txt', character(), quote
 = "",sep = "\n")
docs <- Corpus(VectorSource(post))
dtm_bank <- DocumentTermMatrix(docs, control = list(dictionary=compla
int_wordbank,
                                           tolower=T, removeNumbers=
T,removePunctuation=T))
X <- as.matrix(dtm_bank)
Y = as.numeric(rowSums(X) < 1) #distinguish whether a complaint word
exists in the post
data$prediction = Y
noncomplaintpost <- data[data$prediction == 1,]
noncomplaintpost <- noncomplaintpost[,c(1,5)] #returns a table that c
ontains non negative posts

#==========================================================#
#           run svm model to test for precision          #
#==========================================================#
stops = c('united','southwest','american','jetblue','virginamerica',
'delta','amp')
dtm_post <- DocumentTermMatrix(docs, control = list(tolower=T, remove
Numbers=T,removePunctuation=T,
                                                 stopwords = c(stops,s
topwords('english')),
                                                 stripWhitespace=T))
X.full <- as.matrix(dtm_post)
inter = intersect(colnames(X),colnames(X.full))
dtm_eval <- DocumentTermMatrix(docs, control = list(dictionary=inter,
                                                 tolower=T, removeNumb
ers=T,removePunctuation=T))
X_eval <- as.matrix(dtm_eval)

set.seed(1)
n=length(Y)
n1=round(n*0.8)
n2=n-n1
train=sample(1:n,n1)

Evaluation <- function(pred, true, class)
{
  tp <- sum( pred==class & true==class)
  fp <- sum( pred==class & true!=class)
  tn <- sum( pred!=class & true!=class)
  fn <- sum( pred!=class & true==class)
  precision <- tp/(tp+fp)
  precision
}

i = 1
for (i in 1:4555){
  if (Y[i] == 1) {
    Y[i] = 'non-negative'
  } else {
    Y[i] = 'negative'
  }
}
```

```
svm.model <- svm(factor(Y[train]) ~ ., data = X_eval[train,], kernel=
'linear')
```

```
## Warning in svm.default(x, y, scale = scale, ..., na.action = na.ac
tion):
## Variable(s) 'trap' and 'fool' and 'shittiest' and 'abysmal' and 's
houts'
## and 'greedy' constant. Cannot scale data.
```

```
pred <- predict(svm.model, X_eval[-train,])
pred.class <- as.numeric( pred == 'negative')
Y = as.numeric(rowSums(X) > 0)
Evaluation( pred.class, Y[-train], 1 ) #precision is 0.974753
```

```
## [1] 0.974753
```

```
write.csv(noncomplaintpost,'final.csv',row.names = F)

#==========================================================#
#                          END                             #
#==========================================================#
```