Sentiment Analysis in R: Part 1) Structured Data

By Anahita Sanandaji (please do not share without permission)

IMPORTANT NOTE: *Please consider this is not a political assignment. We are just using famous tweeter datasets from two former presidents. These files are publicly available and are widely used as classical examples in teaching sentiment analysis. Therefore, we are not advocating/pushing any political agenda in this assignment.*

Step 0: start by reading packages (you may need to install them). Do not forget to set your directory.

```
#============================================
# Install and Load Packages
#============================================
install.packages("twitteR")
install.packages("plyr")
install.packages("stringr")
install.packages("tm")
install.packages("wordcloud")
install.packages("Snowballc")

library(twitteR)
library(plyr)
library(stringr)
library(tm)
library(Snowballc)
library(wordcloud)
```

Step 1: Read lexicons (positive and negative words lists) and run the functions:

```
#============================================
# Step 1: Read the lexicons (list of negative and positive words):
#============================================


#Or more complex lists from files:
positiveWords= readLines("positiveWords.txt")
negativeWords= readLines("negativeWords.txt") # you can also use single ' insetad of "
```

Step 2: Run **generalCleaning** function (Run all lines of code together)

```
##============================================================================
# Step 2: Run generalCleaning function (Run all lines of code together)
#============================================================================

# build a generalCleaning function to implement what we want to consider "genera
#   pass it some data, it will "clean" the data according to our rules, then ret
generalCleaning <- function(dataToClean)
{
  dataToReturn <- gsub(" ?(f|ht)(tp)(s?)(://)(.*)[.|/](.*)", "", dataToClean) #
  dataToReturn <- gsub("<.*?>", "", dataToReturn) # remove HTML
  dataToReturn <- gsub("[^[:graph:]]", " ", dataToReturn) # anything not alphabe
  dataToReturn <- gsub("[[:punct:]]", "", dataToReturn) # remove punctuation
  dataToReturn <- gsub("[[:cntrl:]]", " ", dataToReturn) # remove control charac
  dataToReturn <- gsub("\\d+", "", dataToReturn) # remove numbers
  dataToReturn <- gsub("[[:space:]]", " ", dataToReturn) #remove "white space" c

  #example for later in the activity
  dataToReturn = gsub("äóî", "", dataToReturn) #remove specific characters

  dataToReturn <- tolower(dataToReturn) # notice how we make it lowercase before
  return(dataToReturn)
}
```

Step 3: Run **calculateSentiment** function (Run all lines of code together)

```
#=========================================================================
# build a calculateSentiment function which will require 5 lists of data (dataToScore, dataToDisplay,
#   positiveListOfWords, negativeListOfWords); this function will count all instances of positive matches
#   and subtract all instances of negative matches to suggest a sentiment score for the data:
calculateSentiment <- function(dataToScore, dataToDisplay, positiveListOfWords, negativeListOfWords, nameOfDataset)
{
  # for every row of data in dataToScore we will calculate the sentiment score, then build a list of scores
  #   to list next to dataToDisplay:
  listOfScores <- laply(dataToScore, function(singleRowOfData, positiveListOfWords, negativeListOfWords) {
    words = unlist(str_split(singleRowOfData, '\\s+')) #generates a list of all words in the row of data
    # next, generate a list indicating "true" for every word in the list of "words" that is also in
    #   positiveListOfWords, otherwise indicates "false":
    positiveMatches <- is.element(words, positiveListOfWords)
    negativeMatches <- is.element(words, negativeListOfWords) #same idea, but for negativeListOfWords
    # sum will count up all instances of "true" as 1, so in this case, we are counting the positiveMatches
    #   then subtracting from negativeMatches:
    scoreForSingleRow <- sum(positiveMatches) - sum(negativeMatches)
    return(scoreForSingleRow)
  }, positiveListOfWords, negativeListOfWords, .progress="text" )

  # remove emojis, pictures, videos, etc from our output (notice, dataToDisplay did not run through our
  #   general cleaning)
  dataToDisplay = gsub("[^[:graph:]]", " ", dataToDisplay)

  # create a dataframe which will be the required data structure used to create a CSV (comma separated value) file;
  #   basically, return three columns, "which dataset, sentiment and text" with data listed under each:
  dataToReturn <- data.frame("whichDataset"=nameOfDataset, sentiment=listOfScores, text=dataToDisplay)
  return(dataToReturn)
}
```

Step 4: Read .csv files (given to you)

```
##=========================================================
# Step 4: Read .csv files
#=========================================================

myTweets1 <- read.csv("barackobama.csv", sep=",")
myTweets2 <- read.csv("realdonaldtrump.csv", sep=",")

# look at data
str(myTweets1)
summary(myTweets1)
```

Step 5: Add dataset names and combine

```
##=========================================================
# Step 5: Add dataset names and combine
#=========================================================

myTweets1$whichDataSet <- "Obama"
myTweets2$whichDataSet <- "Trump"

#combin: # pay attention to erros like incorrect names if
myAllTweets <- rbind(myTweets1, myTweets2)
```

Step 6: Do cleaning and sentiment analysis by calling the generalCleaning and calculateSentiment functions:

```
##=========================================================================
### Step 6: Do cleaning and sentiment analysis by calling
### the generalCleaning and calculateSentiment functions:
#=========================================================================

myDataBeforeCleaning <- myAllTweets$text

# 1) clean data by calling the function
myDataAfterCleaning <- generalCleaning(myAllTweets$text) # make sure to run the functions before running this or you will get error

#-------------------------------------------------------------------------
# 2) Do the actual sentiment analysis by calling the function
scores <- calculateSentiment(myDataAfterCleaning, myDataBeforeCleaning, positiveWords, negativeWords, myAllTweets$whichDataSet)
calculateSentiment(myData)
```

Step 7: Check results of Sentiment Analysis and Write to .csv Results file:

```
##=============================================================================
# Step 7: Check results of Sentiment Analysis and Write to .csv Results file:
#=============================================================================

#1) Quick Check of Results
sum(scores$sentiment)
scores[which.max(scores$sentiment), ]
scores[which.min(scores$sentiment), ]
#-----------------------------------------------------------------------------

#write into .csv file: see more help(write.csv)
write.csv(scores, "anahitas_Results.csv", row.names = FALSE)
```

Step 8: Perform some more analysis and create wordcloud

```
##=============================================================================
# Step 8: Check results of Sentiment Analysis and Write to .csv Results file:
#=============================================================================

## Function: remove stopwords, stem document, build TDM Matrix
buildTDM <- function(inputData)
  {
  myCorpusData <- VCorpus(VectorSource((inputData)))
  myCorpusData <- tm_map(myCorpusData,removeWords, stopwords("english"))
  myCorpusData <- tm_map(myCorpusData,stemDocument)
  return (TermDocumentMatrix(myCorpusData))
}

myTDMData <- buildTDM(myDataAfterCleaning)
#-----------------------------------------------------------------------------
### Analysis:

##Frequency

findFreqTerms(myTDMData,lowfreq = 10, highfreq = Inf)
findFreqTerms(myTDMData, 1000, Inf)

##Associations

findAssocs(x=myTDMData, term = "obama",corlimit = 0.4)
```

```
##--- Visualization-----------------------------------------------------------
freqOfTerms <- sort(rowSums(as.matrix(myTDMData)),decreasing = TRUE)
head(freqOfTerms)

## 1)barplot
barplot(freqOfTerms[1:10],col ="lightblue",las = 2)

# get list of words from heading
namesFromData <- names(freqOfTerms)

# 2) WordCloud: show the importance of words based on frequency with a word cloud
wordcloud(namesFromData, freqOfTerms, min.freq = 10, max.words = 75,
          colors=brewer.pal(8, "Dark2"))
```