ISSN: 2473-3857
v2 n4022

An Approach to Harvesting, Cleaning, and Analyzing Data from Twitter Using R

Stephen Hill
hills@uncw.edu

Rebecca Scott
scottra@uncw.edu

Information Systems and Operations Management
University of North Carolina Wilmington
Wilmington, NC 28403 USA

Abstract

Using data from social media can be of great value to businesses and other interested parties. However, harvesting data from social media networks such as Twitter, cleaning the data, and analyzing the data can be difficult. In this article, a step-by-step approach to obtaining data via the Twitter application program interface (API) is described. Cleaning of the data and basic sentiment analysis are also described.

Keywords: Analytics, Social Media, Text Mining, Data Cleaning, Classification

1. INTRODUCTION

Social media has become nearly ubiquitous in modern society with users interacting with friends and celebrities, posting photos of their children, and sharing their opinions on a variety of topics. The ubiquity of social media has driven the rapid growth of social media networks. For example, two of the predominant social media networks in the United States, Facebook and Twitter, have grown steadily since their founding. Facebook reported over one billion daily active users in early 2016 (Company Info, n.d.) and Twitter reported over 300 million monthly active users (Company, n.d.). The sheer size of these networks and the information that their users are willing to share present rich opportunities for businesses to market to existing and potential customers and to accrue valuable customer information. In this article, the authors focus on harvesting tweets and related data from the Twitter social media network. The process for cleaning and preparing this data and approaches to basic analysis of the data are described.

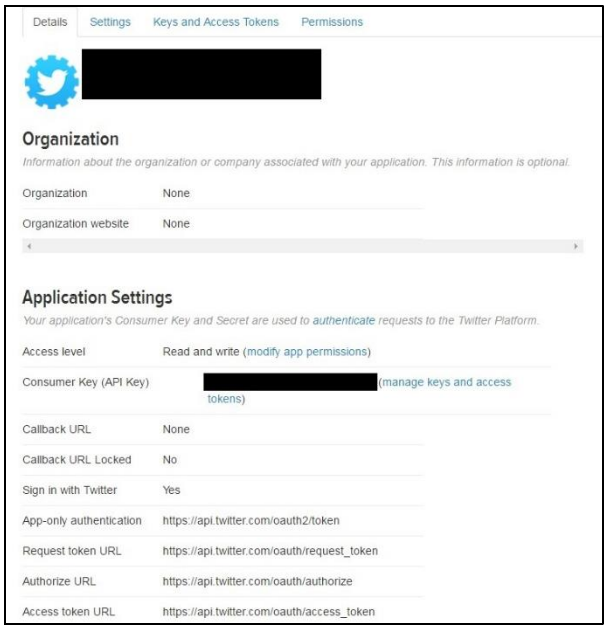
Twitter provides developers access to their network via its application program interface (API). Accessing, harvesting, and analyzing social media data via Twitter's API is not a new phenomenon. Doing these tasks in R, a popular open-source statistical programming software package, is also not new. However, providing a single reference via which an interested data analyst can be guided, in a step-by-step manner, through the process of Twitter data harvesting, cleaning, and analysis is valuable. This article attempts to present such a reference. The closest such reference is from Dannemann and Heimann (2014), but this reference has proven to be outdated, particularly in its description of access to the Twitter API. Other relevant examples include Breen (2011), Bryl (2014), and de Vries (2016). However, these examples either rely on out-of-date approaches to access the Twitter API or are incomplete in that they provide very limited examples of analysis.

2. COLLECTING TWITTER DATA

To retrieve data from Twitter a user must first gain access to the Twitter API. Access to the API will allow the user to collect archived tweets (limited to approximately seven to nine days of historical tweets, subject to API rate limits) or real-time tweets (subject to API rate limits). Twitter may change the process by which users are granted access to the API. The authors have verified that the process described in this article is valid as of late May 2016.

Accessing the Twitter API

In order to gain access to the Twitter API to collect data a user must first sign up for Twitter (Free, https://twitter.com/signup). A Twitter user can then register as a Twitter developer (Free, https://dev.twitter.com/). Twitter developer registration is a one-time task and does not need to be repeated. Once registered as a developer, the user should then create a Twitter application at https://apps.twitter.com/. Select "Create New App". The user will then be prompted to "Create an application".



[redacted]

Organization

Information about the organization or company associated with your application. This information is optional.

Organization	None
Organization website	None

Application Settings

Your application's Consumer Key and Secret are used to authenticate requests to the Twitter Platform.

Access level	Read and write (modify app permissions)
Consumer Key (API Key)	[redacted](manage keys and access tokens)
Callback URL	None
Callback URL Locked	No
Sign in with Twitter	Yes
App-only authentication	https://api.twitter.com/oauth2/token
Request token URL	https://api.twitter.com/oauth/request_token
Authorize URL	https://api.twitter.com/oauth/authorize
Access token URL	https://api.twitter.com/oauth/access_token

Figure 1: Twitter Application Information Screenshot

Enter the requested information. Note that it is acceptable to use a placeholder URL for the requested Website. Agree to the Developer Agreement, and a Twitter application has been developed. The user can then access their application by clicking on the application name. Figure 1 shows the application information for a sample application developed by this article's authors. Personally identifying and other confident information has been redacted from this figure.

To use the newly created application to access the Twitter API and collect tweets data, the user should now open R. For this article, the authors used R version 3.3 (R Development Core Team, 2016) and RStudio version 0.99.902 (RStudio Team, 2016). If the user does not have R and/or RStudio, they may be obtained from https://www.r-project.org/ and https://www.rstudio.com/products/rstudio/download/, respectively. Note that both software packages are available at no cost and in versions for Windows, Mac OS X, and Linux. When installing these software packages, be sure to install R first before installing RStudio. RStudio is optional but provides a useful development environment for R.

Several R packages are needed for this project. See Table 1 for a listing of the required packages. Each package should be installed via the R install.packages command if not done previously. The packages should then be loaded via the R library command.

Package
caret
caTools
dplyr
e1071
ggplot2
httr
plyr
qdap
rattle
RColorBrewer
RCurl
RJSONIO
ROAuth
rpart
rpart.plot
SnowballC
streamR
stringr
tm
twitteR
wordcloud

Table 1: Necessary R Packages

Appendix 1 contains the R script that is used to access the Twitter API. Each line in the script is numbered to enable easier referencing in the article. Line numbers should be removed when the script is used in R. Also, R's commenting sign, "#", is used to denote comments in the script.

To connect to the API, the user must first obtain the appropriate certificates for interaction with Twitter's servers. This is accomplished by line (1). Lines (2) to (4) are then used to prepare for Twitter API access authorization.

Next, the user should assign their Consumer Key and Consumer Secret to appropriately named R objects (Lines (5) and (6)). The user can obtain their Key and Secret from the "manage keys and access tokens" link on the Twitter Application information screen (see Figure 1). Note that the article authors' Key and Secret have been replaced by X's in the sample lines of code. The user should replace the X's with their own Key and Secret. Lines (7) and (8) are then used to establish the connection to the Twitter API.

The Twitter application access token and secret are then assigned to appropriately named R objects. As before, the author's token and secret have been replaced by X's. The user should replace the X's with their own token and secret values.

The Twitter API access authorization can be saved to a .Rdata file by line (11). The user can then, in the future, load the authorization via line (12) and gain reauthorization to access the API via line (13). If this is done, there is no need to run lines (1) to (11) for future instances of data harvesting. At this point, the user has gained access to the Twitter API and is ready to harvest data.

Collecting Archived Twitter Data
The searchTwitter function as shown in Line (14) was run on May 26th, 2016 to collect up to 25,000 tweets featuring the hashtag "#Warriors" and no older than May 19th, 2016. This code returned 25,000 tweets with the oldest tweet from May 23rd, 2016. The searchTwitter function can be used to gather tweets for a set of search terms (with terms separated by "+") and/or by geocoded location. A geocoded location is provided as a point specified by latitude and longitude and a radius (in miles or kilometers) from that point. For example, line (15) shows the same search as in line (14), but modified to only collect tweets generated within 50 miles of San Francisco, California.

Collecting Real-Time Twitter Data
The filterStream command that is part of the streamR package is used to collect real-time Twitter data (Barbera, 2016). The authors used the filterStream command to collect 30 minutes of tweets generated during the March 10th, 2016 Republican Presidential debate. A total of 79,456 tweets were collected. Line (16) shows the R code used to collect the tweets. The "timeout = 1800" portion of the code specifies the duration (in seconds) during which tweets should be collected.

3. PREPARING TWITTER DATA FOR ANALYSIS

Once tweet data has been harvested from Twitter, it must then be prepared for analysis. Before preparing the data it is prudent to save an archive of the raw data from Twitter. This is done via lines (17) and (18). Then, to enable easier re-use of the preparation and analysis R code, the tweets data is given a generic name of "some_tweets" in line (19). The tweet text is then isolated from the remainder of the tweet data and stored as "some_txt" via line (20).

Lines (21) through (27) prepare the text data by removing any "RT" and "@" portions for retweeted text, the "@" for tweets directed at a user , punctuation marks, numbers, web links, extra spaces, and non-graphical characters. These changes are done via R's gsub function and regular expressions (regex). Lines (28) through (30) convert all text to lower case. A function in line (29) from Sanchez (2012) is used to prevent R's tolower() function from producing errors that prevent the upper to lower case conversion from taking place. Lines (31) and (32) then remove the originally searched for term "gopdebate" from the text and also removes a related Twitter hashtag. Other frequently appearing, but uninformative text can be removed in a similar manner if desired. The table in Appendix 3 shows how a particular tweet is modified by the procedures in lines (21) to (32).

The text is then converted into a corpus (body of text) by line (33). The tm package is then used in line (34) to remove common words, known as stop words, which often have little analytical value. Examples of English stop words include "a", "is", and "the". With text data that is now cleaned, the user can proceed to analysis of the data.

ISSN: 2473-3857
v2 n4022

4. BASIC ANALYSIS OF TWITTER DATA

There are a variety of possible avenues for the analysis of Twitter data. In this section, a few basic approaches to analysis are presented. The first approach is a simple word cloud via the wordcloud R package. Line (36) shows the R code used to develop the word cloud shown in Figure 2. Note that the wordcloud function requires that a corpus be passed to it. Word clouds are often used to provide an appealing visualization of the frequency and importance of words that appear in a body of text (Heimerl, Lohmann, Lange, & Ertl, 2014). Lines (37) then converts the corpus to a data frame object. Line (38) assigns the cleaned (but not yet stemmed) text to a variable in the some_tweets data frame. This is done to facilitate analysis.

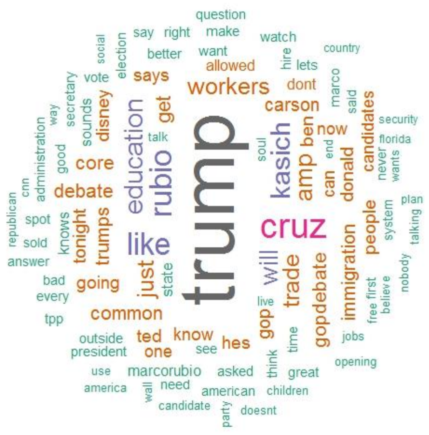


Figure 2: Republican Presidential Debate Twitter Wordcloud

A common analysis technique for text data is sentiment analysis. A significant portion of sentiment analysis work in the academic focuses on capturing the polarity (e.g., positive, neutral, or negative) of a text author's feelings, emotions, or thoughts (Pang and Lee, 2008). A simple way to conduct sentiment analysis is to consider each text document (each tweet in the case of this article) as a "bag of words" (Salton and McGill, 1986). Each word in the document is then matched against a dictionary of words that have been classified as either positive or negative. Positive words are assigned a sentiment of +1, negative words are assigned a sentiment of -1, and unmatched words are assigned a sentiment of 0. The sentiment of the document is then the sum of the sentiment of each word in the document. In this article, dictionaries of positive and negative sentiment words from Hu and Liu (2004) are used. Lines (39) and (40) read in these dictionaries.

A custom function from Bryl (2014) is used to score the sentiment of each tweet. Before using this function, lines (41) and (42) should be run to appropriately format the data for use in the function. The custom function is shown in line (43). The user may wish to save this function as a separate .R file and then source this file as needed (see line (44)). The sentiment scores are generated by line (45) and are then copied to the some_tweets data frame by line (46). The data frame is saved (as a CSV file) for backup purposes by line (47). The user can then analyze the sentiment scores. For example, line (48) shows the R code used to develop the histogram of tweet sentiment scores shown in Figure 3. Sentiment scores can be categorized as Negative, Positive, or Neutral by lines (49) to (51).

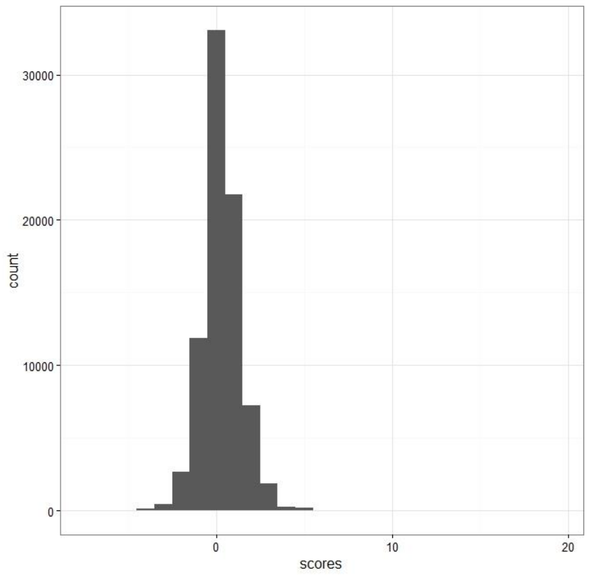


Figure 3: Republican Presidential Debate Twitter Sentiment Histogram

The final data manipulation that is described in this article is stemming. Lines (52) and (53) are used to stem the tweets. Stemming is used to reduce a word down to its root. Doing so reduces the number of unique terms (i.e., words) that are considered during analysis (Meyer, Hornik, & Feinerer, 2008). For example, the word "applied", when stemmed, becomes "appli". After stemming the tweet that was shown in the table in Appendix 3, becomes "anoth tonight".

Once stemming is complete, the frequencies of terms (words) in the text corpus is examined by line (54). If the user wishes to see terms that appear frequently, line (55) can be used. In this example, terms that appear 20 or more times across all of the tweets in the corpus are shown. Line (56) is used to eliminate terms from the corpus that appear infrequently. Setting the numerical value in this line to a number closer to 1 will result in more terms being retained. Smaller values will result in fewer terms being retained. Line (57) will display the number of terms remaining. In this example, 296 unique terms were retained. Lines (58) and (59) convert the remaining terms into a data frame and give each column in the data frame a name that corresponds to the terms. Line (60) transfers the sentiment score categories to the data frame created in lines (58) and (59).

The user can now consider a variety of analytics techniques with an objective to determine what terms that appear in a tweet are most predictive of whether or not the tweet is classified as "Negative" or "Not Negative". In this article, a classification tree is used. Classification trees rely on recursive portioning to predict the classification of entities in a dataset (Breiman, Friedman, Stone, & Olshen, 1984). In this article, a classification tree will be constructed that predicts whether a tweet is "Negative" or "Not Negative".

Before constructing the tree, the dataset is split. Lines (61) through (63) are used to split the data into a training set comprised of 70% of the data and a testing set with the remaining 30% of the data. The training set is then used for predictive model building while the testing set is set aside for evaluation of model quality. The splitting is performed via the sample.split function from the caret package.

The rpart package is then used to develop the classification tree. Line (64) generates the tree and line (65) displays the tree. The resulting classification tree is shown in Appendix 5. The tree classifies a tweet as either "Negative" (total sentiment less than zero) or "Not Negative" (total sentiment of zero or greater). To classify a tweet as negative or not negative, the user of the tree examines the tweet and begins at the top of the tree with the statement "disast >= 0.5". This statement is True (follow the Yes branch of the tree) if the tweet contains at least one word with the root "disast". For example, if the tweet contains the word "disaster", the user would follow the Yes branch of the tree and classify the tweet as "Negative". If the tweet does not contain a word with the root "disast" then the No branch is followed. This process is repeated until the tweet is classified.

Line (66) uses the classification tree to predict the classification of tweets in the testing dataset. Of the 23,836 tweets in the testing portion of the dataset, 20,311 were correctly classified by the tree and 3,525 were incorrectly classified. This gives the tree an out-of-sample accuracy of 85.2%. This accuracy compares favorably to a naïve (no information) accuracy of 81.0%. The accuracy calculations are performed in line (67) by the confusionMatrix function from the caret package.

5. CONCLUSIONS AND OPPORTUNITIES FOR FUTURE WORK

This article provides a convenient and easy to follow step-by-step approach to harvesting, cleaning, and analyzing data from Twitter. The approach accesses the Twitter API via the R statistical programming software. Archived tweets or streaming, real-time tweets are then collected. The tweets data is then cleaned and prepared for analysis. The article closes with a brief description of basic sentiment analysis of the data.

The step-by-step approach provided in this article is valuable to businesses and other interested parties. Analyzing social media, such as Twitter, allows businesses to evaluate customer impressions of their goods and services. This in turn can allow organizations to use social media as an effective customer service tool.

There are a variety of options available for this work to be expanded. For example, tweets could be collected that relate to a particular event (e.g., the Super Bowl, elections, etc.) and sentiment regarding these events could then be analyzed. Any number of text mining techniques and other analytical techniques for classification could also be applied.

6. REFERENCES

Barbera, P. (2016). *Introducing the streamR package*. *Pablobarbera.com*. Retrieved 30 May 2016, from http://pablobarbera.com/blog/archives/1.html.

Breen, J. (2011). *Mining Twitter for Airline Consumer Sentiment*. *Inside-r.org*. Retrieved 30 May 2016, from http://www.inside-r.org/howto/mining-twitter-airline-consumer-sentiment.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. CRC Press.

Bryl, S. (2014). *Twitter sentiment analysis with R*. *AnalyzeCore.com*. Retrieved 30 May 2016, from http://analyzecore.com/2014/04/28/twitter-sentiment-analysis/.

Company Info. (n.d.). Retrieved May 31, 2016, from http://newsroom.fb.com/company-info/.

Company. (n.d.). Retrieved May 31, 2016, from https://about.twitter.com/company.

Danneman, N., & Heimann, R. (2014). Social media mining with R. Packt Publishing Ltd.

de Vries, A. (2016). *Text Analysis 101: Sentiment Analysis in Tableau & R*. *The Information Lab*. Retrieved 30 May 2016, from http://www.theinformationlab.co.uk/2016/03/02/text-analysis-101-sentiment-analysis-in-tableau-r/.

Heimerl, F., Lohmann, S., Lange, S., & Ertl, T. (2014). Word cloud explorer: Text analytics based on word clouds. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on* (pp. 1833-1842). IEEE.

Hu, M., & Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 168-177). ACM.

Meyer, D., Hornik, K., & Feinerer, I. (2008). Text mining infrastructure in R. *Journal of statistical software*, 25(5), 1-54.

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2), 1-135.

R Development Core Team. (2014). R: A Language and Environment for Statistical Computing (Version 3.3) [Software]. Vienna, Austria: R Foundation for Statistical Computing. Available from http://www.R-project.org.

RStudio Team (2016). RStudio: Integrated Development for R (Version 0.99.902) [Software]. Boston: RStudio Inc.. Available from http://www.rstudio.com.

Salton, G. & McGill, M., editors (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill.

Sanchez, G. (2012). *Catching errors when using tolower*. *Gastonsanchez.com*. Retrieved 30 May 2016, from http://gastonsanchez.com/how-to/2012/05/29/Catching-errors-when-using-tolower/.

ISSN: 2473-3857
v2 n4022

Appendix 1 (Twitter API Access R Script)

The R script below is used to gain access to the Twitter API. The user must register as a Twitter develop and create a Twitter “application” before executing this code. Comments are embedded in the script and are indicated by R’s commenting sign #. Each line in the script is numbered to enable easier referencing in the article. The line numbers should be excluded when the script is input in R.

```
(1)   download.file(url='http://curl.haxx.se/ca/cacert.pem',
      destfile='cacert.pem')
(2)   reqURL <- 'https://api.twitter.com/oauth/request_token'
(3)   accessURL <- 'https://api.twitter.com/oauth/access_token'
(4)   authURL <- 'https://api.twitter.com/oauth/authorize'
(5)   consumerKey <- ' XXXX ' #Replace X’s with your Consumer Key
(6)   consumerSecret <- 'XXXX' #Replace X’s with your Consumer Secret
(7)   Cred <- OAuthFactory$new(consumerKey=consumerKey,
                  consumerSecret=consumerSecret,
                  requestURL=reqURL,
                  accessURL=accessURL,
                  authURL=authURL)
(8)   Cred$handshake(cainfo = system.file('CurlSSL', 'cacert.pem', package
      = 'RCurl'))
(9)   access_token = 'XXXX' #Replace the X’s with your Access Token
(10)  access_secret= 'XXXX' #Replace the X’s with your Access Token
(11)  save(Cred, file='twitter authentication.Rdata')
(12)  load('twitter authentication.Rdata')
(13)  setup_twitter_oauth(consumerKey,consumerSecret,access_token,acce
      ss_secret)
```

ISSN: 2473-3857
v2 n4022

Appendix 2 (Twitter Archived and Streaming Tweets Collection Examples)

The R script below is used harvest tweets. Line (14) is used for archived tweets and line (15) is used for real-time, streaming tweets. As in the other appendices, comments are embedded in the script and are indicated by R's commenting sign #. Each line in the script is numbered to enable easier referencing in the article. The line numbers should be excluded when the script is input in R.

```
(14)  tweets = searchTwitter("#Warriors",n=25000, retryOnRateLimit=120,
      lang="en", since="2016-05-15", resultType="recent")
(15)  tweets = searchTwitter("#Warriors",n=25000, retryOnRateLimit=120,
      lang="en", geocode="37.7749,-122.4194,50 mi", since="2016-05-
      19", resultType="recent")
(16)  filterStream(file.name = "tweetsGOP.json",
      track = c("GOPDebate", "gopdebate", "GOPdebate"), language = "en",
      timeout = 1800, oauth = Cred)
```

Appendix 3 (Twitter Data Cleaning and Preparation for Analysis)

The R script below is used to perform cleaning of the tweets data. As in the other appendices, comments are embedded in the script and are indicated by R's commenting sign #. Each line in the script is numbered to enable easier referencing in the article. The line numbers should be excluded when the script is input in R.

```
(17)  tweet_archive = do.call("rbind", lapply(tweets, as.data.frame))
      #OR for streaming tweets
      tweets_archive <- parseTweets("tweetsGOP.json", simplify = FALSE)
(18)  write.csv(tweet_archive,file="tweets.csv")
(19)  some_tweets = tweets_archive
(20)  some_txt = sapply(some_tweets, function(x) x$getText())
(21)  some_txt = gsub("(RT|via)((?:\\b\\W*@\\w+)+)", "", some_txt)
(22)  some_txt = gsub("@\\w+", "", some_txt)
(23)  some_txt = gsub("[[:punct:]]", "", some_txt)
(24)  some_txt = gsub("[[:digit:]]", "", some_txt)
(25)  some_txt = gsub("http\\w+", "", some_txt)
(26)  some_txt = gsub("^\\s+|\\s+$", "", some_txt)
(27)  some_txt = gsub("[^[:graph:]]", " ", some_txt)
(28)  try.error = function(x)
(29)  {
              y = NA
              try_error = tryCatch(tolower(x), error=function(e) e)
              if (!inherits(try_error, "error"))
              y = tolower(x)
              return(y)
      }
(30)  some_txt = sapply(some_txt, try.error)
(31)  some_txt = gsub("gopdebate", "", some_txt)
(32)  some_txt = gsub("cnndebate", "", some_txt)
(33)  corpus = Corpus(VectorSource(some_txt))
(34)  corpus = tm_map(corpus, removeWords, stopwords("english"))
(35)  corpus = tm_map(corpus,PlainTextDocument)
```

Action	Tweet Text
Original Tweet	Another #GOPDebate tonight. https://t.co/TIEy5DwSzo
Punctuation Removed	Another GOPDebate tonight httpstcoTIEy5DwSzo
Numbers Removed	Another GOPDebate tonight httpstcoTIEyDwSzo"
Removed Web Links	Another GOPDebate tonight
Converted Text to Lower Case	another gopdebate tonight
Removed Specific Hashtags	another tonight

Appendix 4 (Basic Twitter Data Analysis)

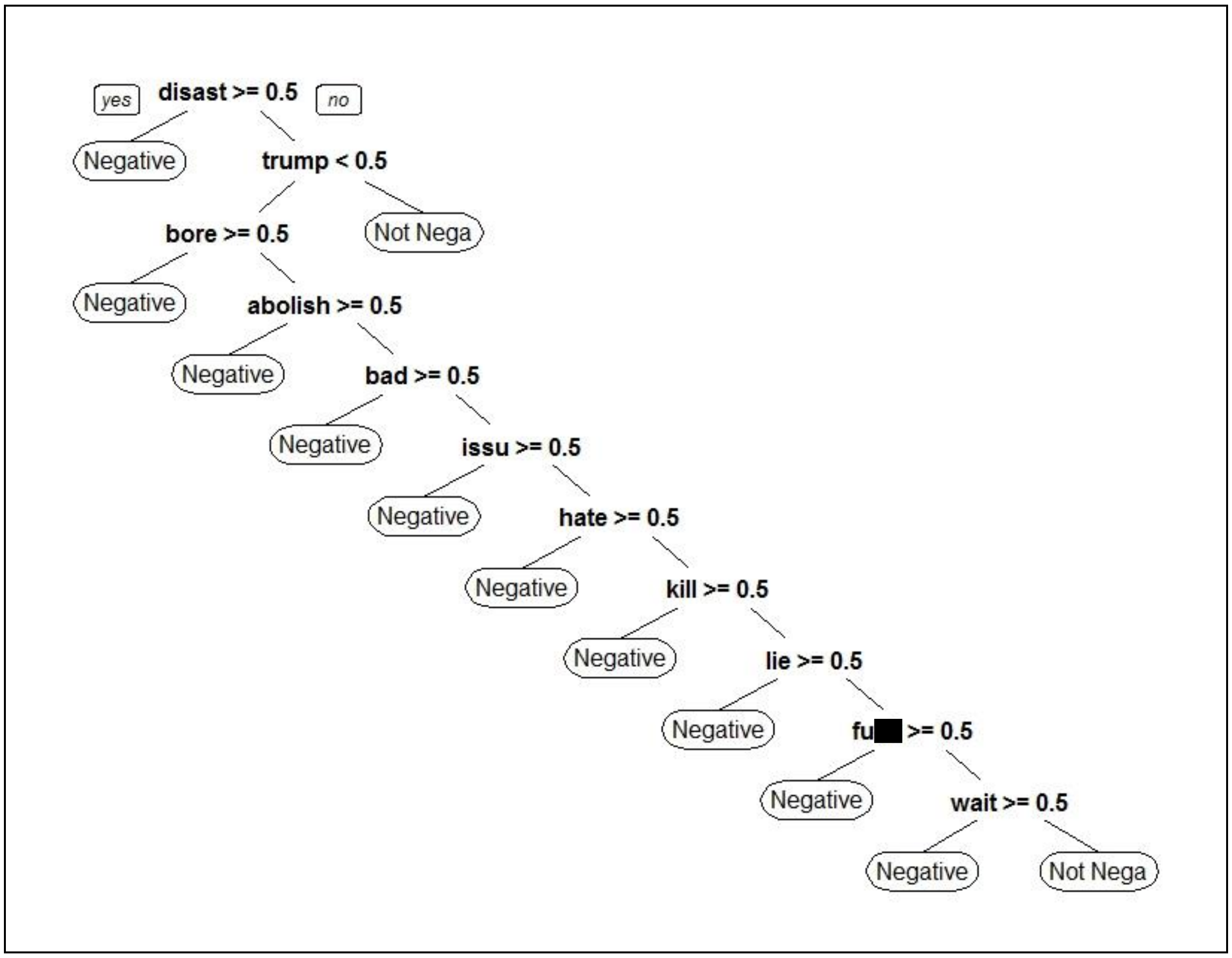
The R script below is used to perform basic analysis of the tweets. As in the other appendices, comments are embedded in the script and are indicated by R's commenting sign #. Each line in the script is numbered to enable easier referencing in the article. The line numbers should be excluded when the script is input in R.

```
(36)  wordcloud(corpus, scale=c(5,0.5), max.words=100,
       random.order=FALSE, rot.per=0.35,
       use.r.layout=FALSE, colors=brewer.pal(8, "Dark2"))
(37)  corpus_df = as.data.frame(corpus)
(38)  some_tweets$unstem = corpus_df$text
(39)  pos <- scan('C:/XXXX/positive-words.txt', what='character',
      comment.char=';') #Replace the X's with the correct path to this file
(40)  neg <- scan('C:/XXXX/negative-words.txt', what='character',
      comment.char=';') #Replace the X's with the correct path to this file
(41)  Dataset <- some_tweets$unstem
(42)  Dataset <- as.factor(Dataset)
(43)  score.sentiment <- function(sentences, pos.words, neg.words,
      .progress='none')
      {
       require(plyr)
      require(stringr)
      scores <- laply(sentences, function(sentence, pos.words, neg.words){
      sentence <- gsub('[[:punct:]]', "", sentence)
      sentence <- gsub('[[:cntrl:]]', "", sentence)
       sentence <- gsub('\\d+', "", sentence)
       sentence <- tolower(sentence)
      word.list <- str_split(sentence, '\\s+')
      words <- unlist(word.list)
      pos.matches <- match(words, pos.words)
      neg.matches <- match(words, neg.words)
      pos.matches <- !is.na(pos.matches)
      neg.matches <- !is.na(neg.matches)
      score <- sum(pos.matches) - sum(neg.matches)
      return(score)
      }, pos.words, neg.words, .progress=.progress)
      scores.df <- data.frame(score=scores, text=sentences)
      return(scores.df)
      }
(44)  source("scoresent.R")
(45)  scores <- score.sentiment(Dataset, pos.words, neg.words,
      .progress='text')
```

```
(46)  some_tweets$scores = scores$score
(47)  write.csv(some_tweets, file="tweetsandscores.csv",
      row.names=TRUE)
(48)  ggplot(some_tweets,aes(x=scores)) + geom_histogram(bins=27) +
      theme_bw()
(49)  some_tweets$scorescat[some_tweets$scores < 0] <- "Negative"
(50)  some_tweets$scorescat[some_tweets$scores >= 0] <- "Not Negative"
(51)  some_tweets$scorescat = as.factor(some_tweets$scorescat)
(52)  corpus = tm_map(corpus, stemDocument, language = "english")
(53)  corpus = tm_map(corpus,PlainTextDocument)
(54)  frequencies = DocumentTermMatrix(corpus)
(55)  findFreqTerms(frequencies, lowfreq=20)
(56)  sparse = removeSparseTerms(frequencies, 0.995)
(57)  sparse
(58)  tweetsSparse = as.data.frame(as.matrix(sparse))
(59)  colnames(tweetsSparse) = make.names(colnames(tweetsSparse))
(60)  tweetsSparse$scorescat = some_tweets$scorescat
(61)  split = sample.split(tweetsSparse$scorescat, SplitRatio = 0.7)
(62)  train = subset(tweetsSparse, split==TRUE)
(63)  test = subset(tweetsSparse, split==FALSE)
(64)  tweetTREE = rpart(scorescat ~ ., data=train, method="class")
(65)  prp(tweetTREE)
(66)  predictTREE = predict(tweetTREE, newdata=test, type="class")
(67)  confusionMatrix(predictTREE,test$scorescat)
```

ISSN: 2473-3857
v2 n4022

Appendix 5 (Classification Tree)


yes
disast >= 0.5
no
Negative
trump < 0.5
bore >= 0.5
Not Nega
Negative
abolish >= 0.5
Negative
bad >= 0.5
Negative
issu >= 0.5
Negative
hate >= 0.5
Negative
kill >= 0.5
Negative
lie >= 0.5
Negative
fu[redacted] >= 0.5
Negative
wait >= 0.5
Negative
Not Nega