Hashtag EVO 2016 Process

Emeka Nwosu

August 4, 2016

This is just a follow up to my Hashtag EVO2016 project in which I acquired a large collection of tweets and performed some analyses on the data. Here I will present The R code used for the project that I left out and talk about how the whole process went. There are many tutorials on how to use the TwitteR package and make word clouds, so I'm no trying to make another one of those. Instead I am showing the problems and difficulties I encountered applying a applying a "Twitter analysis" to the type of data I wanted to work with.

### Load Packages

library(twitteR) ##acess Twitter  
library(wordcloud) ##make wordloud  
library(tm) ##text mining  
library(ggplot2)  
library(data.table)

* twitteR: Interacting with Twitter's API
* wordcloud: Making wordclouds
* tm: Text mining functions
* ggplot2: Graphics
* data.table: data tables are faster than base R's data frames. This isn't a requirement, but it probably saved me some time.

### Setup AUthorization for Twitter API

twit <- setup\_twitter\_oauth(consumer\_key = "(insert consumer\_key here)",   
 consumer\_secret = "(insert consumer\_secret here)",   
 access\_token = "(insert access\_token here",   
 access\_secret = "(insert acess\_secret here)")

There are plenty of tutorials on setting this up. A quick google search will lead you somewhere with better information than I can give. I only know enough to get it to work. I do not understand all the ins and outs.

### Getting the Tweets

##Getting Tweets Day 1 (wasn't able to acess tweets because they were 10 days old by the   
##time of attempt)  
tweets1 <- searchTwitter("#EVO2016", n=21589, since = "2016-7-15", until = "2016-7-16")  
tweets.meta1 <- twListToDF(tweets1) ##metadata for tweets  
tweets.text1 <- sapply(tweets1, function(x) x$getText())  
##approximately 21589 tweets  
  
##Getting Tweets day 2  
tweets2 <- searchTwitter("#EVO2016", n=39271, since = "2016-7-16", until = "2016-7-17")  
tweets.meta2 <- twListToDF(tweets2) ##metadata for tweets  
tweets.text2 <- sapply(tweets2, function(x) x$getText())  
##approximately 39271 tweets  
  
## Getting Tweets day 3  
tweets3 <- searchTwitter("#EVO2016", n=76350, since = "2016-7-17", until = "2016-7-18")  
tweets.meta3 <- twListToDF(tweets3) ##metadata for tweets  
tweets.text3 <- sapply(tweets3, function(x) x$getText())  
##approximately 137455 tweets

This was the most difficult part. I did not anticipate how much data I was trying to access nor how long it would take. I seriously cannot explain how frustrated I was getting this to work. I don't even know where to start with this ordeal. I guess I'll try to go in chronological order. I may start rambling for a bit here, but that will give you an idea of how I felt.

I assumed there would be a lot of data. So instead of downloading all the tweets at once, I tried doing it day by day (Friday's tweets, then Saturday's, then Sundays). Even so, it still took a very long time and I was not sure if It was still working or not. I frequently got "rate limited," which is something Twitter's API does when you are making too many requests. R then shows a counter starting at about 120 retry attempts. It will count down, but then sometimes go back up. I waited for many hours (like 8 hours in some cases) and it would often get stuck without me knowing. Even stopping the process would take hours because it was working with so much data.

When the stopped process would complete stopping it would say something like "could not acquire the requested number of tweets. Returned X amount." When you use the "searchTwitter" function, you request a number of tweets. In this case, I didn't know how many tweets there would be so I initially just put a bunch of 9's. So it returned the number of tweets it could acquire, but would not actually save them to my computer. So for each day, I had to make an initial attempt with "n=999999", wait many hours and eventually stop the process. Then I would do it again with the number returned so it would naturally stop the process instead of continuing to search for a number of tweets that don't exist.

This whole thing took so long that I discovered there's a restriction on accessing tweets based on hashtag. Once I figured out how to do this, I tried getting tweets from day 1, but the number dropped from 21,000 to about 5,000. Then it would not give me any tweets at all. Then I tried testing it with day 2 and that number was dropping. Tweets 10 days old cannot be accessed with "searchTwitter". Because of that, I was not able to use tweets from day 1 in my project.

### Saving the Tweets

save(tweets2, file = "tweets2.RData") ##Raw twiiter data  
write(tweets.text2, "tweets2.txt") ##list of tweets  
write.csv(tweets.meta2, "tweets2.csv") #data frame of tweets  
  
save(tweets3, file = "tweets3.RData") ##Raw twiiter data  
write(tweets.text3, "tweets3.txt") ##list of tweets  
write.csv(tweets.meta3, "tweets3.csv") #data frame of tweets  
  
  
##Loading Saved Tweet data frames  
tweets2 <- read.csv("C:/Users/Emeka/Desktop/tweets2.csv", comment.char="#")  
tweets3 <- read.csv("C:/Users/Emeka/Desktop/tweets3.csv", comment.char="#")  
tweets.meta <- data.table(rbind(tweets2, tweets3))  
##Sometimes the "created" variable converts to factor instead date.  
##Convert back to date-time with this code  
tweets.meta$created <- as.POSIXct(strptime(as.character(tweets.meta$created),'%Y-%m-%d %H:%M:%S'))  
rm(tweets2, tweets3)  
  
  
write.csv(tweets.meta, "tweets.csv") #data frame of tweets  
  
tweets.text <- c(tweets.text2, tweets.text3)

I initially saved the data just for the purpose of formality for reproducibility, but after seeing my ability to access these tweets dwindle away each second, it was absolutely necessary it was imperative to save what data I had. This process also took a while, but was very quick relative to the time it took to download it.

I was not sure what type of structure the twitter data come in from "searchTwitter." I saved it as an R script, but I don't think it worked correctly. To be safe, I saved the list of tweets after using the "gettext" function as a txt file. It worked well, but I had a few problems reusing it. It didn't separate the text of the tweets properly. The metadata is already in data frame format so I simply saved that as a csv which took almost no time at all. I can also get the text from the tweets in their original form from the data frame so that's really all that is needed.

### Volume of Tweets

##Number of Tweets  
length(tweets.text)  
##115584 (plus the 21589 missing tweets from friday = grand total of 137173)  
  
  
##Number tweets by day  
21589 from friday  
length(tweets.text2)  
39234  
length(tweets.text3)  
76350  
  
  
##Time Chart  
  
ggplot(data = tweets.meta, aes(x = created)) +  
 geom\_histogram(aes(fill = ..count..)) +  
 theme(legend.position = "none") +  
 xlab("Time (Eastern Standard)") + ylab("Number of tweets") +   
 scale\_fill\_gradient(low = "midnightblue", high = "red") +  
 scale\_y\_continuous(expand = c(0, 500))

The only problem that occurred here is the format of the timestamps in the metadata data frame. When you initially use twListToDF on the data from searchTwitter, it saves the "created" variable as a "'POSIXct' 'POSIXt'". However, when you save the data frame and reload it, it comes back as a factor variable. The code to convert it back to its original format must be done every time the data is loaded into R. The data need to be in this format for the histogram of time stamps to work.

### Word Cloud

tweets <- tweets.meta$text  
tweets.text <- gettext(tweets)  
  
  
tweets.text <- tolower(tweets.text)  
# Replace blank space ("rt")  
tweets.text <- gsub("rt", "", tweets.text)  
# Replace @UserName  
tweets.text <- gsub("@\\w+", "", tweets.text)  
# Remove punctuation  
tweets.text <- gsub("[[:punct:]]", "", tweets.text)  
# Remove links  
tweets.text <- gsub("http\\w+", "", tweets.text)  
# Remove tabs  
tweets.text <- gsub("[ |\t]{2,}", "", tweets.text)  
# Remove blank spaces at the beginning  
tweets.text <- gsub("^ ", "", tweets.text)  
# Remove blank spaces at the end  
tweets.text <- gsub(" $", "", tweets.text)  
  
  
#create corpus  
tweets.text.corpus <- Corpus(VectorSource(tweets.text))  
#clean up by removing stop words  
tweets.text.corpus <- tm\_map(tweets.text.corpus, function(x)removeWords(x,stopwords()))  
  
###Optional Create Term Document Matrix  
##tweets.tdm <- TermDocumentMatrix(tweets.text.corpus, control = list(minWordLength = 1000))  
##tdm.matrix <- as.matrix(tweets.tdm)  
##word.count <- sort(rowSums(tdm.matrix), decreasing = TRUE)  
  
  
  
##WOrd Coud  
wordcloud(tweets.text.corpus,min.freq = 100000, scale=c(7,0.5),colors=brewer.pal(8, "Dark2"), random.color= TRUE, random.order = FALSE, max.words = 100)  
  
#####EVO Top 8#####  
  
##Top 8 most favorited tweets  
head(select(tweets.meta, text, favoriteCount, screenName)[order(-favoriteCount)], n = 8)   
  
##Top 8 most retweeted tweets  
head(select(tweets.meta, text, retweetCount, screenName)[order(-retweetCount)], n = 1)   
  
##Users who tweeted the most  
head(summary(as.factor(tweets.meta$screenName)), n = 8)

I have never made a nice looking word cloud like I see in other people's tutorials. I thought it was because I never have enough data, but this time, it was because I had too much data. The term document matrix just outright failed. I think it was because I converted text from the metadata instead of just straight from the searchTwitter data. I had no other choice because I wasn't able to (not willing to) go through the process of downloading that data again. Next year, I'll do it right and I'll save the data frame of the termDocumentMatrix like I did with the metadata for reproducibility.

### The Rest

##Top 8 most favorited tweets  
head(select(tweets.meta, text, favoriteCount, screenName)[order(-favoriteCount)], n = 8)   
  
##Top 8 most retweeted tweets  
head(select(tweets.meta, text, retweetCount, screenName)[order(-retweetCount)], n = 1)   
  
##Users who tweeted the most  
head(summary(as.factor(tweets.meta$screenName)), n = 8)

Nothing to say here really. It was just a matter of organizing and subsetting the data.