Twitter Data Analysis

Jyotishka Datta

11/01/2018

Contents

0.1 Background

- This is a preliminary analysis of twitter data from the timeline of Little Rock Police Department (USername: LRpolice) to explore the patterns of the tweets shared on their timeline.
- The analysis was done using many R packages, the key ones are `TwitteR`, `topicmodels`, and `sentiment`. Most of the analysis here were done using the R codes given in these tutorials:

1. Twitter Data Analysis with R
2. Twitter Authentication.

I will provide you codes for coming with the final dataset where you should have sentiment score and time to analyze the trend of a specific user of keyword. However, the codes are relevant for repeating this analysis at a general level, if you are interested.

0.2 Twitter Authentication

The first step is authentication on Twitter. You will need a Twitter account for doing this as Twitter only allows real human users to download and analyse their data.

If none of the project members have a Twitter handle, let me know and I can supply you a downloaded dataset for any specific query or username.

Visit the second link above for authentication and follow the steps mentioned there:

Twitter Authentication.

```r
install.packages(c("devtools", "rjson", "bit64", "httr"))
#RESTART R session!
library(devtools)
install_github("twitteR", username="geoffjentry")
library(twitteR)

## Replace "YOUR_API_KEY" by the actual API key.
## Instructions on http://thinktostart.com/twitter-authentification-with-r/
```

```
api_key <-  "YOUR_API_KEY"
api_secret <- "YOUR_API_SECRET"
access_token <- "YOUR_ACCESS_TOKEN"
access_token_secret <- "YOUR_ACCESS_TOKEN_SECRET"

setup_twitter_oauth(api_key,api_secret)
```

0.3 Data Download and cleaning

- The two most useful functions for scraping Twitter data are `userTimeline` and `searchTwitter`.
- The Twitter API allows one to pull **3,200** tweets from any user's timeline or keyword. We start by extracting **3200** tweets with the keywords 'EASPORTS'. (I used this because EA Sports has recently received a lot of backlash in Social Media over their Star Wars Battlefront game, but you can use **any** keyword you want!)

```
library(twitteR)
tweets <- searchTwitter("EA+SPORTS", n = 3200)
(n.tweet <- length(tweets))
tweets.df <- twListToDF(tweets)
```

Now we can convert the extracted tweets into a dataframe and access individual tweets by row numbers:

```
tweets.df <- twListToDF(tweets) # Make a dataframe
save(tweets.df,file="easports_tweets.Rdata") # save data for lazy loading
```

If we save the tweets, we can also use lazy loading to do the analysis without extracting tweets every single time.

```
load(file="easports_tweets.Rdata") # Lazy loading
tweets.df[1, c("id", "created", "screenName", "replyToSN",
                 "favoriteCount", "retweetCount", "longitude",  "latitude", "text")]
```

```
##                    id             created screenName replyToSN
## 1 931339825092939778 2017-11-17 01:54:52    Emosabe      <NA>
##   favoriteCount retweetCount longitude latitude
## 1             0            0      <NA>     <NA>
##
## 1 I don t really buy much of EA s stuff because half of it is sports and the other half is garbage.
```

For example, we can print tweet number 1 and make text fit for slide width:

```
writeLines(strwrap(tweets.df$text[1], 60))
```

```
## I don t really buy much of EA s stuff because half of it is
## sports and the other half is garbage.  but I m definite
## https://t.co/0lZ0kKzdRl
```

0.4 Data Cleaning

- The next step involves data cleaning, which involves the following basic steps:
- Convert to lower case
- Remove URLs.
- Remove anything other than English letters or space

- Remove stopwords
- Remove extra whitespace
- I will paste my codes here but if you don't want to mess with this, it should be fine to use as is. You can also copy-paste all of this into a single program file in R and run that file to get cleaned data.
- I should also mention that data cleaning is a painful yet important step in natural lanuage processing. We usually keep cleaning until we get a somewhat noise free corpus, but it is subjective.

```r
## Text Cleaning
library(tm)
library(SnowballC)

# build a corpus, and specify the source to be character vectors
myCorpus <- Corpus(VectorSource(tweets.df$text))

# remove URLs
removeURL <- function(x) gsub("http[^[:space:]]*", "", x)
myCorpus <- tm_map(myCorpus, content_transformer(removeURL))
```

```
## Warning in tm_map.SimpleCorpus(myCorpus, content_transformer(removeURL)):
## transformation drops documents
```

```r
# remove anything other than English letters or space
removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]*", "", x)
myCorpus <- tm_map(myCorpus, content_transformer(removeNumPunct))
```

```
## Warning in tm_map.SimpleCorpus(myCorpus,
## content_transformer(removeNumPunct)): transformation drops documents
```

```r
# remove stopwords
myStopwords <- c(setdiff(stopwords('english'), c("r", "big")),
                 "use", "see", "used", "via", "amp")
myCorpus <- tm_map(myCorpus, removeWords, myStopwords)
```

```
## Warning in tm_map.SimpleCorpus(myCorpus, removeWords, myStopwords):
## transformation drops documents
```

```r
# remove extra whitespace
myCorpus <- tm_map(myCorpus, stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(myCorpus, stripWhitespace): transformation
## drops documents
```

```r
# convert to lower case
myCorpus <- tm_map(myCorpus, content_transformer(tolower))
```

```
## Warning in tm_map.SimpleCorpus(myCorpus, content_transformer(tolower)):
## transformation drops documents
```

```r
# keep a copy for stem completion later
myCorpusCopy <- myCorpus

myCorpus <- tm_map(myCorpus, stemDocument) # stem words
```

```
## Warning in tm_map.SimpleCorpus(myCorpus, stemDocument): transformation
## drops documents
```

```r
writeLines(strwrap(myCorpus[[190]]$content, 60))
```

```
## rt truongasm ea sport it game narrat voiceov it wasnt game
```

- The last step in this replacing similar words by one unique word, usually plural ('players', 'games') by singular nouns ('player', 'game') such that these words do not counted separately in our analysis. **This depends on your search query a lot**.

```r
replaceWord <- function(corpus, oldword, newword) {
  tm_map(corpus, content_transformer(gsub),
         pattern=oldword, replacement=newword)
}
myCorpus <- replaceWord(myCorpus, "players", "player")
```

```
## Warning in tm_map.SimpleCorpus(corpus, content_transformer(gsub), pattern =
## oldword, : transformation drops documents
```

```r
myCorpus <- replaceWord(myCorpus, "games", "game")
```

```
## Warning in tm_map.SimpleCorpus(corpus, content_transformer(gsub), pattern =
## oldword, : transformation drops documents
```

- Once we have a clean corpus, we create a term document matrix that gives the frequency of each term in a collection of documents (here tweets). This term-document matrix (or, TDM) is then analzed by suitable Statistical methods for finding associations or performing sentiment analyses.

```r
tdm <- TermDocumentMatrix(myCorpus,control = list(wordLengths = c(1, Inf)))
tdm
```

```
## <<TermDocumentMatrix (terms: 4365, documents: 3200)>>
## Non-/sparse entries: 40088/13927912
## Sparsity           : 100%
## Maximal term length: 28
## Weighting          : term frequency (tf)
```

```r
save(tdm,file="easports_tdm.Rdata")
```

- People usually save these data-sets (tweetsdf, tdm) for future use because extraction takes time and you don't want to waste a lot of time every time you analyze a specific set of tweets.

```r
rm(list=ls()) # clean your workspace
load(file="easports_tdm.Rdata") # Lazy loading
load(file="easports_tweets.Rdata") # Lazy loading
```

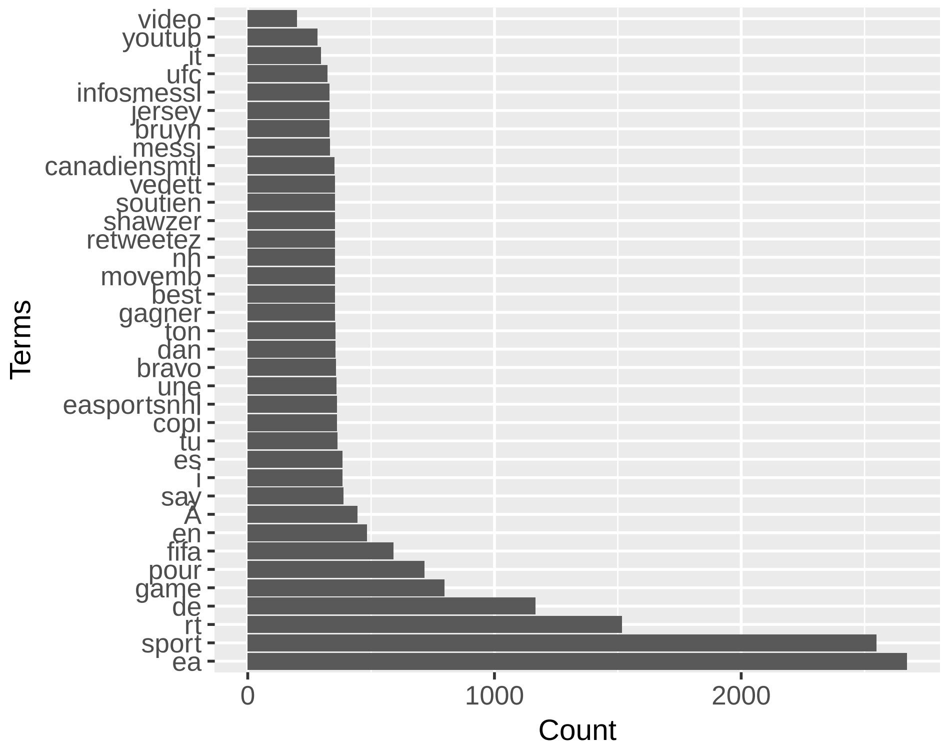
0.5 Frequent Terms

- Our first job is to print and visualize the frequent terms that gives an overall idea about the corpus.

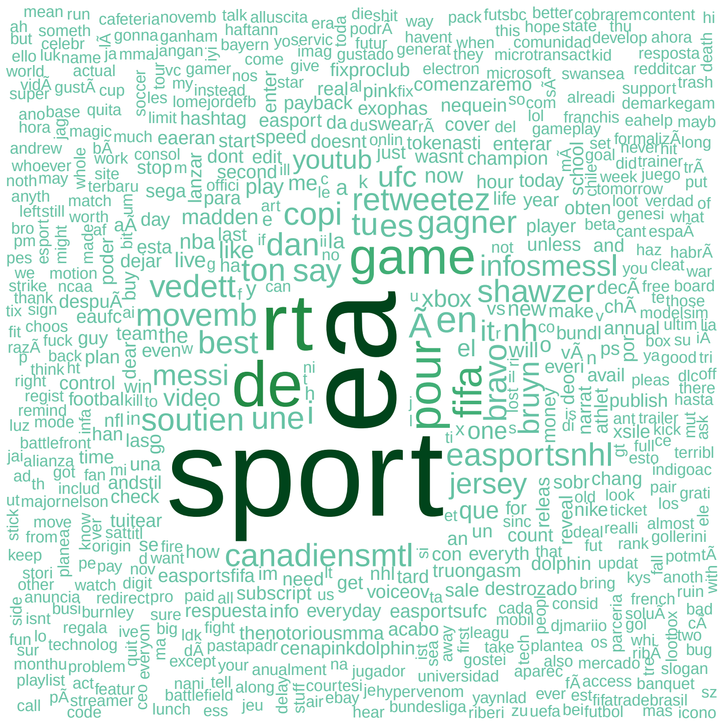
```r
library(tm)
# inspect frequent words
(freq.terms <- findFreqTerms(tdm, lowfreq = 200))
```

```
##  [1] "ea"          "i"           "sport"       "ufc"
##  [5] "video"       "youtub"      "game"        "rt"
##  [9] "it"          "best"        "say"         "fifa"
## [13] "copi"        "de"          "en"          "es"
## [17] "bravo"       "canadiensmtl" "dan"        "easportsnhl"
## [21] "gagner"      "movemb"      "nh"          "pour"
## [25] "retweetez"    "shawzer"      "soutien"      "ton"
## [29] "tu"           "une"          "vedett"       "Ã"
## [33] "bruyn"        "infosmessl"   "jersey"       "messi"
```

- The following figure is a barplot of the most frequent words in the last 3,200 tweets on President Trump's timeline.



- Wordcloud gives a prettier picture to look at (of course with the same story).



0.6 Association Mining

- One might be interested in looking at keywords associated with certain words of interest such as "game":

```r
# which words are associated with 'game'?
findAssocs(tdm, "game", 0.2)
```

```
## $game
##        it  voiceov     wasnt    narrat truongasm   instead    consid
##      0.52     0.49      0.49      0.48      0.48      0.25      0.24
## microsoft    those     worth      deal     digit    includ    rocket
##      0.24     0.24      0.24      0.23      0.23      0.22      0.22
##  tomorrow subscript   destini    unicom
##      0.21     0.21      0.20      0.20
```

0.7 Sentiment Analysis

- How are the sentiments associated with the extracted tweets. As the following table shows, they might be more positive than negative!

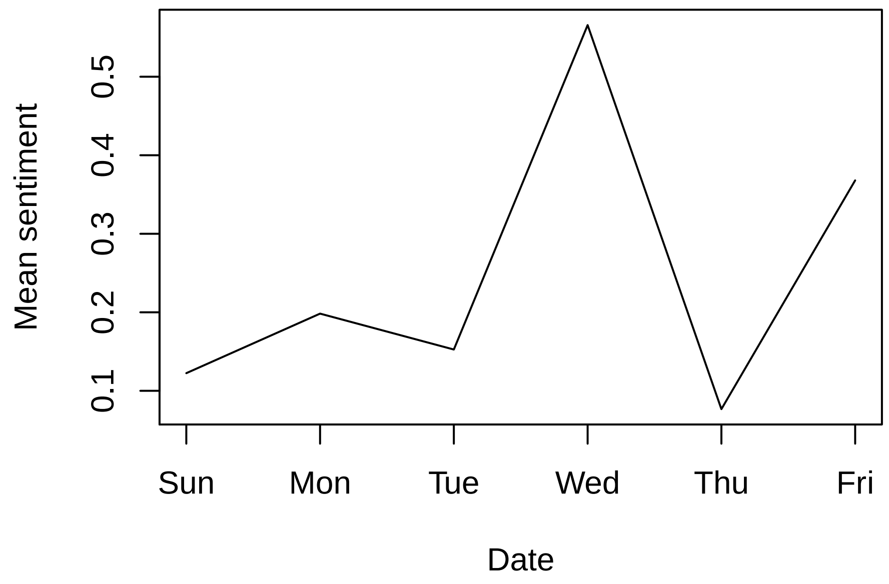
```r
# library(sentiment)
# sentiments <- sentiment(tweets.df$text)
# table(sentiments$polarity)
library(SentimentAnalysis)
sentiments <- analyzeSentiment(tweets.df$text)
sentiments$polarity <-convertToDirection(sentiments$SentimentQDAP)
table(sentiments$polarity)
```

```
##
## negative  neutral positive
##      417     1543     1240
```

- We can also plot them by time to see how the sentiments have changed over time. This type of analysis is very commonly done with political candidates to estimate/predict their popularity and see what contributes to it.

```
# sentiment plot

sentiments$score <- 0
sentiments$score[sentiments$polarity == "positive"] <- 1
sentiments$score[sentiments$polarity == "negative"] <- -1
sentiments$date <- as.IDate(tweets.df$created)
result <- aggregate(score ~ date, data = sentiments, mean)
plot(result, type = "l", ylab= "Mean sentiment", xlab = "Date")
```



1 Project Goals:

1. Extract Twitter data using the code here with your choice of keyword or person. Use your imagination. Think what trends you want to analyze? Social movement / education / sports - anything is fine !
2. I expect at least two different searches, keywords or personalities or trends and both types of trend tests + plots for each, but if you want to do more (one each), that would be even better.
3. Perform trend test for the tweets you have extracted. Plot the trends. The 'trend test' tests trend over time, if you look at the total counts, can you use a nonparametric test to test if there were significantly more positive tweets than negative tweets (or vice versa)?
4. Report your Statistical conclusion in terms of the story.

5. What else would be interesting to find out? (Open question).