Association Rule Mining and Twitter

Twitter API Setup

To access a Twitter API you will need to set up an account and receive a consumerKey, the comsumerSecret, the access_Token, and the access_Secret. A popular library and API: “twitteR”.

```
knitr::opts_chunk$set(echo = TRUE, warning=FALSE, results ='show',include=TRUE,messages=FALSE)

####### Twitter in R
#  Consumer API keys
#  Access token & access token secret

## I have created a text file that contains the
## consumerKey, the comsumerSecret, the access_Token, and the access_Secret
## They are comma seperated.
# Insert your consumerKey and consumerSecret below



consumerKey='SiMslBfTdWEimvLweRDTTrZVH'
consumerSecret='FoPYqK3uwpzutwE6G1RmQvPbRJ8RChFSLfIlgRAcFHjymKDzHh'
access_Token='1084502204038479872-v2czQaDlMt9ikoLnxhiQYk8Yb3f0RT'
access_Secret='U9ktzvd5rEwcK13mttsgwAujS0VxNPtJstxXcEE5znnid'
```

Once you have your keys, you can set up the API.

```
requestURL='https://api.twitter.com/oauth/request_token'
accessURL='https://api.twitter.com/oauth/access_token'
authURL='https://api.twitter.com/oauth/authorize'

### NOTES: rtweet is another excellent option
## https://mkearney.github.io/blog/2017/06/01/intro-to-rtweet/
### https://rtweet.info/

### Install the needed packages...
#install.packages("twitteR")
#install.packages("ROAuth")
# install.packages("rtweet")
library(arules)
```

```
## Loading required package: Matrix

##
## Attaching package: 'arules'

## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
library(rtweet)
library(twitteR)
```

```
##
## Attaching package: 'twitteR'
## The following object is masked from 'package:rtweet':
##
##     lookup_statuses
```

```r
library(ROAuth)
library(jsonlite)
```

```
##
## Attaching package: 'jsonlite'

## The following object is masked from 'package:rtweet':
##
##     flatten
```

```r
#install.packages("streamR")
#library(streamR)
#install.packages("rjson")
library(rjson)
```

```
##
## Attaching package: 'rjson'

## The following objects are masked from 'package:jsonlite':
##
##     fromJSON, toJSON
```

```r
#install.packages("tokenizers")
library(tokenizers)
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------------------ tidyv

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts --------------------------------------------------------------------------- tidyverse_c
## x tidyr::expand()   masks Matrix::expand()
## x dplyr::filter()   masks stats::filter()
## x purrr::flatten()  masks jsonlite::flatten(), rtweet::flatten()
## x rjson::fromJSON() masks jsonlite::fromJSON()
## x dplyr::id()       masks twitteR::id()
## x dplyr::lag()      masks stats::lag()
## x dplyr::location() masks twitteR::location()
## x dplyr::recode()   masks arules::recode()
## x rjson::toJSON()   masks jsonlite::toJSON()
```

```r
library(plyr)
```

```
## -------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## -------------------------------------------------------------------------

##
## Attaching package: 'plyr'
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following object is masked from 'package:purrr':
##
##     compact
```

```
## The following object is masked from 'package:twitteR':
##
##     id
```

```
library(dplyr)
library(ggplot2)
#install.packages("syuzhet")  ## sentiment analysis
library(syuzhet)
```

```
##
## Attaching package: 'syuzhet'
```

```
## The following object is masked from 'package:rtweet':
##
##     get_tokens
```

```
library(stringr)
#install.packages("arulesViz")
library(arulesViz)
```

```
## Loading required package: grid
```

Collecting Tweets

Next we will set up the API and search for a particular hash tag. We will store the tweets with the designated hash in a csv file for safe keeping. Here, we choose "#Trump" in hopes to get a 100 tweets easily.

```
##############  Using twittR ###########################################################
setup_twitter_oauth(consumerKey,consumerSecret,access_Token,access_Secret)
```

```
## [1] "Using direct authentication"
```

```
Search<-twitteR::searchTwitter("avocado",n=90,since="2019-08-01")
Search_DF <- twListToDF(Search)
TransactionTweetsFile = "Choc.csv"
#Search_DF$text[1]


## Start the file
Trans <- file(TransactionTweetsFile)
## Tokenize to words
Tokens<-tokenizers::tokenize_words(Search_DF$text[1],stopwords = stopwords::stopwords("en"),
          lowercase = TRUE,  strip_punct = TRUE, strip_numeric = TRUE,simplify = TRUE)
## Write squished tokens
cat(unlist(str_squish(Tokens)), "\n", file=Trans, sep=",")
close(Trans)

## Append remaining lists of tokens into file
```

```r
## Recall - a list of tokens is the set of words from a Tweet
Trans <- file(TransactionTweetsFile, open = "a")
for(i in 2:nrow(Search_DF)){
  Tokens<-tokenize_words(Search_DF$text[i],stopwords = stopwords::stopwords("en"),
            lowercase = TRUE,  strip_punct = TRUE, simplify = TRUE)
  cat(unlist(str_squish(Tokens)), "\n", file=Trans, sep=",")
}
close(Trans)
```

Tweets as Transactions

In this section we will read in the tweets stored in the CSV file using the (Association Rule Mining) ARM library. Each tweet will be considered a basket of words. We can use ARM to determine associations of words in tweets.

```r
######### Read in the tweet transactions
TweetTrans <- read.transactions(TransactionTweetsFile,
                                rm.duplicates = FALSE,
                                format = "basket",
                                sep=","
                                ## cols =
                                )
#inspect(TweetTrans)
## See the words that occur the most
Sample_Trans <- sample(TweetTrans, 20)
#summary(Sample_Trans)

## Read the transactions data into a dataframe
TweetDF <- read.csv(TransactionTweetsFile, header = FALSE, sep = ",")
head(TweetDF)
```

```
##   V1              V2      V3   V4     V5      V6      V7      V8      V9
## 1 rt        sxrgihoe  things give energy    eggs bananas   brown    rice
## 2 rt        sxrgihoe  things give energy    eggs bananas   brown    rice
## 3 rt       ncitybase nctbase mput pengen    baca   debut wattpad      ku
## 4 rt        sxrgihoe  things give energy    eggs bananas   brown    rice
## 5 rt        sxrgihoe  things give energy    eggs bananas   brown    rice
## 6 rt leariellesimone   foods give energy almonds  apples avocado bananas
##           V10      V11    V12     V13    V14      V15    V16 V17
## 1       sweet potatoes coffee   water yogurt  oatmeal cocain
## 2       sweet potatoes coffee   water yogurt  oatmeal cocain
## 3         gak       rt    aja   nanti   kita mutualan   dulu
## 4       sweet potatoes coffee   water yogurt  oatmeal cocain
## 5       sweet potatoes coffee   water yogurt  oatmeal cocain
## 6 blueberries    brown   rice coconut   dark    choco
```

```r
#(str(TweetDF))
```

Cleaning the text data

Note that cleaning the text data is very important in text mining applications. Tweets are especially “messy”. We will remove “rt”, “http”, etc and any other strings of no importance.

```r
## Convert all columns to char
TweetDF<-TweetDF %>%
  mutate_all(as.character)
(str(TweetDF))
```

```
## 'data.frame':    92 obs. of  17 variables:
##  $ V1 : chr  "rt" "rt" "rt" "rt" ...
##  $ V2 : chr  "sxrgihoe" "sxrgihoe" "ncitybase" "sxrgihoe" ...
##  $ V3 : chr  "things" "things" "nctbase" "things" ...
##  $ V4 : chr  "give" "give" "mput" "give" ...
##  $ V5 : chr  "energy" "energy" "pengen" "energy" ...
##  $ V6 : chr  "eggs" "eggs" "baca" "eggs" ...
##  $ V7 : chr  "bananas" "bananas" "debut" "bananas" ...
##  $ V8 : chr  "brown" "brown" "wattpad" "brown" ...
##  $ V9 : chr  "rice" "rice" "ku" "rice" ...
##  $ V10: chr  "sweet" "sweet" "gak" "sweet" ...
##  $ V11: chr  "potatoes" "potatoes" "rt" "potatoes" ...
##  $ V12: chr  "coffee" "coffee" "aja" "coffee" ...
##  $ V13: chr  "water" "water" "nanti" "water" ...
##  $ V14: chr  "yogurt" "yogurt" "kita" "yogurt" ...
##  $ V15: chr  "oatmeal" "oatmeal" "mutualan" "oatmeal" ...
##  $ V16: chr  "cocain" "cocain" "dulu" "cocain" ...
##  $ V17: chr  "" "" "" "" ...
```

```
## NULL
```

```r
# We can now remove certain words
TweetDF[TweetDF == "t.co"] <- ""
TweetDF[TweetDF == "rt"] <- ""
TweetDF[TweetDF == "http"] <- ""
TweetDF[TweetDF == "https"] <- ""

## Clean with grepl - every row in each column
MyDF<-NULL
for (i in 1:ncol(TweetDF)){
  MyList=c() # each list is a column of logicals ...
  MyList=c(MyList,grepl("[[:digit:]]", TweetDF[[i]]))
  MyDF<-cbind(MyDF,MyList)  ## create a logical DF
  ## TRUE is when a cell has a word that contains digits
}
## For all TRUE, replace with blank
TweetDF[MyDF] <- ""
(head(TweetDF,10))
```

```
##          V1              V2             V3        V4     V5      V6
## 1                  sxrgihoe         things      give energy    eggs
## 2                  sxrgihoe         things      give energy    eggs
## 3                 ncitybase        nctbase      mput pengen    baca
## 4                  sxrgihoe         things      give energy    eggs
## 5                  sxrgihoe         things      give energy    eggs
## 6           leariellesimone          foods      give energy almonds
## 7                  sxrgihoe         things      give energy    eggs
## 8                  sxrgihoe         things      give energy    eggs
## 9                  sxrgihoe         things      give energy    eggs
## 10 trilogy          rosehip transformation cleansing    oil      dr
##           V7      V8      V9         V10      V11      V12       V13
## 1    bananas   brown    rice       sweet potatoes   coffee     water
## 2    bananas   brown    rice       sweet potatoes   coffee     water
## 3      debut wattpad      ku         gak               aja     nanti
## 4    bananas   brown    rice       sweet potatoes   coffee     water
## 5    bananas   brown    rice       sweet potatoes   coffee     water
## 6     apples avocado bananas blueberries    brown      rice   coconut
## 7    bananas   brown    rice       sweet potatoes   coffee     water
## 8    bananas   brown    rice       sweet potatoes   coffee     water
## 9    bananas   brown    rice       sweet potatoes   coffee     water
## 10 roebuck’s    surf  chaser       serum   tatcha luminous overnight
##       V14      V15    V16 V17
## 1  yogurt  oatmeal cocain
## 2  yogurt  oatmeal cocain
## 3    kita mutualan   dulu
## 4  yogurt  oatmeal cocain
## 5  yogurt  oatmeal cocain
## 6    dark    choco
## 7  yogurt  oatmeal cocain
## 8  yogurt  oatmeal cocain
## 9  yogurt  oatmeal cocain
## 10 memory      ser
```

```
# Now we save the dataframe using the write table command
write.table(TweetDF, file = "UpdatedChocolate.csv", col.names = FALSE,
            row.names = FALSE, sep = ",")
TweetTrans <- read.transactions("UpdatedChocolate.csv", sep =",",
            format("basket"),  rm.duplicates = TRUE)
```

```
## distribution of transactions with duplicates:
## items
## 1 2
## 2 1
```

```
#inspect(TweetTrans)
```

ARM

Next we will apply the apriori algorithm to find the associations including computing the support, confidence and lift. Read more on the arules library to tweak / tune the following code to achieve desired results.

```
# So that you do not have an enormous amount of rules, you can thresholds for
# support, confidence and lift ... also minlength for the rules.
TweetTrans_rules = arules::apriori(TweetTrans,
            parameter = list(support=.15, confidence=.85, minlen=3))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.85    0.1    1 none FALSE            TRUE       5    0.15      3
##  maxlen target   ext
##      10  rules FALSE
##
```

```
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 13
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[156 item(s), 92 transaction(s)] done [0.00s].
## sorting and recoding items ... [16 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 7 8 9 10 done [0.01s].
## writing ... [223418 rule(s)] done [0.08s].
## creating S4 object  ... done [0.18s].
```

```r
inspect(TweetTrans_rules[1:10])
```

```
##      lhs                  rhs        support   confidence lift     count
## [1]  {cocain,sxrgihoe} => {coffee}   0.6956522 1          1.373134 64
## [2]  {cocain,coffee}   => {sxrgihoe} 0.6956522 1          1.437500 64
## [3]  {coffee,sxrgihoe} => {cocain}   0.6956522 1          1.437500 64
## [4]  {cocain,sxrgihoe} => {eggs}     0.6956522 1          1.373134 64
## [5]  {cocain,eggs}     => {sxrgihoe} 0.6956522 1          1.437500 64
## [6]  {eggs,sxrgihoe}   => {cocain}   0.6956522 1          1.437500 64
## [7]  {cocain,sxrgihoe} => {water}    0.6956522 1          1.373134 64
## [8]  {cocain,water}    => {sxrgihoe} 0.6956522 1          1.437500 64
## [9]  {sxrgihoe,water}  => {cocain}   0.6956522 1          1.437500 64
## [10] {cocain,sxrgihoe} => {things}   0.6956522 1          1.373134 64
```

```r
## sorted
SortedRules_conf <- sort(TweetTrans_rules, by="confidence", decreasing=TRUE)
inspect(SortedRules_conf[1:40])
```

```
##      lhs                    rhs        support   confidence lift     count
## [1]  {cocain,sxrgihoe}   => {coffee}   0.6956522 1          1.373134 64
## [2]  {cocain,coffee}     => {sxrgihoe} 0.6956522 1          1.437500 64
## [3]  {coffee,sxrgihoe}   => {cocain}   0.6956522 1          1.437500 64
## [4]  {cocain,sxrgihoe}   => {eggs}     0.6956522 1          1.373134 64
## [5]  {cocain,eggs}       => {sxrgihoe} 0.6956522 1          1.437500 64
## [6]  {eggs,sxrgihoe}     => {cocain}   0.6956522 1          1.437500 64
## [7]  {cocain,sxrgihoe}   => {water}    0.6956522 1          1.373134 64
## [8]  {cocain,water}      => {sxrgihoe} 0.6956522 1          1.437500 64
## [9]  {sxrgihoe,water}    => {cocain}   0.6956522 1          1.437500 64
## [10] {cocain,sxrgihoe}   => {things}   0.6956522 1          1.373134 64
## [11] {cocain,things}     => {sxrgihoe} 0.6956522 1          1.437500 64
## [12] {sxrgihoe,things}   => {cocain}   0.6956522 1          1.437500 64
## [13] {cocain,sxrgihoe}   => {oatmeal}  0.6956522 1          1.373134 64
## [14] {cocain,oatmeal}    => {sxrgihoe} 0.6956522 1          1.437500 64
## [15] {oatmeal,sxrgihoe}  => {cocain}   0.6956522 1          1.437500 64
## [16] {cocain,sxrgihoe}   => {potatoes} 0.6956522 1          1.373134 64
## [17] {cocain,potatoes}   => {sxrgihoe} 0.6956522 1          1.437500 64
## [18] {potatoes,sxrgihoe} => {cocain}   0.6956522 1          1.437500 64
## [19] {cocain,sxrgihoe}   => {sweet}    0.6956522 1          1.373134 64
## [20] {cocain,sweet}      => {sxrgihoe} 0.6956522 1          1.437500 64
## [21] {sweet,sxrgihoe}    => {cocain}   0.6956522 1          1.437500 64
## [22] {cocain,sxrgihoe}   => {yogurt}   0.6956522 1          1.352941 64
## [23] {cocain,yogurt}     => {sxrgihoe} 0.6956522 1          1.437500 64
## [24] {sxrgihoe,yogurt}   => {cocain}   0.6956522 1          1.437500 64
## [25] {cocain,sxrgihoe}   => {rice}     0.6956522 1          1.194805 64
## [26] {cocain,rice}       => {sxrgihoe} 0.6956522 1          1.437500 64
## [27] {rice,sxrgihoe}     => {cocain}   0.6956522 1          1.437500 64
## [28] {cocain,sxrgihoe}   => {brown}    0.6956522 1          1.194805 64
## [29] {brown,cocain}      => {sxrgihoe} 0.6956522 1          1.437500 64
## [30] {brown,sxrgihoe}    => {cocain}   0.6956522 1          1.437500 64
## [31] {cocain,sxrgihoe}   => {energy}   0.6956522 1          1.194805 64
## [32] {cocain,energy}     => {sxrgihoe} 0.6956522 1          1.437500 64
## [33] {energy,sxrgihoe}   => {cocain}   0.6956522 1          1.437500 64
## [34] {cocain,sxrgihoe}   => {give}     0.6956522 1          1.194805 64
## [35] {cocain,give}       => {sxrgihoe} 0.6956522 1          1.437500 64
## [36] {give,sxrgihoe}     => {cocain}   0.6956522 1          1.437500 64
## [37] {cocain,sxrgihoe}   => {bananas}  0.6956522 1          1.194805 64
## [38] {bananas,cocain}    => {sxrgihoe} 0.6956522 1          1.437500 64
## [39] {bananas,sxrgihoe}  => {cocain}   0.6956522 1          1.437500 64
## [40] {cocain,coffee}     => {eggs}     0.6956522 1          1.373134 64
```

```
SortedRules_sup <- sort(TweetTrans_rules, by="support", decreasing=TRUE)
inspect(SortedRules_sup[1:40])
```

```
##      lhs                       rhs       support   confidence lift
## [1]  {brown,rice}           => {energy}  0.8369565 1          1.194805
## [2]  {energy,rice}          => {brown}   0.8369565 1          1.194805
## [3]  {brown,energy}         => {rice}    0.8369565 1          1.194805
## [4]  {brown,rice}           => {give}    0.8369565 1          1.194805
## [5]  {give,rice}            => {brown}   0.8369565 1          1.194805
## [6]  {brown,give}           => {rice}    0.8369565 1          1.194805
## [7]  {brown,rice}           => {bananas} 0.8369565 1          1.194805
## [8]  {bananas,rice}         => {brown}   0.8369565 1          1.194805
## [9]  {bananas,brown}        => {rice}    0.8369565 1          1.194805
## [10] {energy,rice}          => {give}    0.8369565 1          1.194805
## [11] {give,rice}            => {energy}  0.8369565 1          1.194805
## [12] {energy,give}          => {rice}    0.8369565 1          1.194805
## [13] {energy,rice}          => {bananas} 0.8369565 1          1.194805
## [14] {bananas,rice}         => {energy}  0.8369565 1          1.194805
## [15] {bananas,energy}       => {rice}    0.8369565 1          1.194805
## [16] {give,rice}            => {bananas} 0.8369565 1          1.194805
## [17] {bananas,rice}         => {give}    0.8369565 1          1.194805
## [18] {bananas,give}         => {rice}    0.8369565 1          1.194805
## [19] {brown,energy}         => {give}    0.8369565 1          1.194805
## [20] {brown,give}           => {energy}  0.8369565 1          1.194805
## [21] {energy,give}          => {brown}   0.8369565 1          1.194805
## [22] {brown,energy}         => {bananas} 0.8369565 1          1.194805
## [23] {bananas,brown}        => {energy}  0.8369565 1          1.194805
## [24] {bananas,energy}       => {brown}   0.8369565 1          1.194805
## [25] {brown,give}           => {bananas} 0.8369565 1          1.194805
## [26] {bananas,brown}        => {give}    0.8369565 1          1.194805
## [27] {bananas,give}         => {brown}   0.8369565 1          1.194805
## [28] {energy,give}          => {bananas} 0.8369565 1          1.194805
## [29] {bananas,energy}       => {give}    0.8369565 1          1.194805
## [30] {bananas,give}         => {energy}  0.8369565 1          1.194805
## [31] {brown,energy,rice}    => {give}    0.8369565 1          1.194805
## [32] {brown,give,rice}      => {energy}  0.8369565 1          1.194805
## [33] {energy,give,rice}     => {brown}   0.8369565 1          1.194805
## [34] {brown,energy,give}    => {rice}    0.8369565 1          1.194805
## [35] {brown,energy,rice}    => {bananas} 0.8369565 1          1.194805
## [36] {bananas,brown,rice}   => {energy}  0.8369565 1          1.194805
## [37] {bananas,energy,rice}  => {brown}   0.8369565 1          1.194805
## [38] {bananas,brown,energy} => {rice}    0.8369565 1          1.194805
## [39] {brown,give,rice}      => {bananas} 0.8369565 1          1.194805
## [40] {bananas,brown,rice}   => {give}    0.8369565 1          1.194805
##      count
## [1]  77
## [2]  77
## [3]  77
## [4]  77
## [5]  77
## [6]  77
## [7]  77
## [8]  77
## [9]  77
## [10] 77
## [11] 77
## [12] 77
## [13] 77
## [14] 77
## [15] 77
## [16] 77
## [17] 77
## [18] 77
## [19] 77
## [20] 77
## [21] 77
## [22] 77
## [23] 77
## [24] 77
## [25] 77
## [26] 77
## [27] 77
## [28] 77
## [29] 77
## [30] 77
## [31] 77
## [32] 77
## [33] 77
## [34] 77
## [35] 77
## [36] 77
## [37] 77
## [38] 77
## [39] 77
## [40] 77
```

Displaying Results

The results will be displayed as an interactive graph.

```
plot (SortedRules_sup[1:50],method="graph",interactive=TRUE,shading="confidence")
plot (SortedRules_conf[1:50],method="graph",interactive=TRUE,shading="confidence")
```