springboard capstone

milti leonard

7/9/2017

Detecting Bot Communities on Twitter and Determining Their Influence

Social media imparts tremendous influence on the ongoing zeitgeist. This influence can have IRL (or real world) consequences and repercussions. The intent of the platform is to allow people of various backgrounds, but similar interests, to connect and form communities of the like-minded that is not tied to geospatial constraints. However, the reality is that various agents create and unleash *bot* accounts that disrupt those communities and alters the digital landscape.

This project will focus on a selected set of *loaded* hashtags (meaning those whose LIWC is representative of a defined agenda) that command a respectable trending status to see whether or no it attracts these *bot* accounts and to what degree. Also to be investigated, is whether it is the human accounts or their *bot* counterparts that take the lead in establishing that trending status. The individual tweet atrributes to be looked at will include

1. timestamp

2. retweet status

3.

On the various accounts themselves, this project will look to the following

1. profile text
2. profile pic (is it an egg)
3. followers
4. following

The R packages that will be utilised in this study are as follows (with one or two being dropped, dependent on subsequent decisions made in the experimental design)

```r
library(caret)
library(igraph)
library(twitteR)
library(streamR)
library(RNeo4j)
library(tidyverse)
library(stringr)

requestURL <- "https://api.twitter.com/oauth/request_token"
accessURL <- "https://api.twitter.com/oauth/access_token"
authURL <- "https://api.twitter.com/oauth/authorize"
consumer_key <- 'QpSBwPC1PwqUQ4eSMCVeXenDQ'
consumer_secret <- '39YRwPaXF9u3TCvjqFaElSxmEEe7xcUJTzg0M2oa3egutPa6B4'
access_token <- '1240280636-GFbqEWiGKiP17KmN4oyrFwdZwQBJEJA3BRy2TmW'
access_secret <- 'vsnTNryz83p0XS4rSz6dw0mGRf0SZC2pPtxGnAkvUikSR'
setup_twitter_oauth(consumer_key, consumer_secret, access_token, access_secret)


library(reticulate)
library(XML)
path_to_python <- "/usr/local/bin/python3.6"
```

```
use_python(path_to_python)
dbot=import("debot")
db = dbot$DeBot('YocR3mKAc7U6hjKZrNnOCk2jjnWrqgSfwWYo8yOb')#This is my API key
#bots=htmlParse(db$get_related_bots('election2016'))#Change 'election2016' to any other topic of your i
cbots=htmlParse(db$get_related_bots('Charlottesville'))

#Now parse the XML data into a dataframe
botlist <- xmlToList(cbots)

library(listviewer)
flatbot <- flatten(botlist)
flatterbot <- flatten(flatbot)
remove(flatbot)
flatbot <- flatten(flatterbot)
newBotlist <- unname(sapply(flatbot, `[`, 2))


#my_oauth <- OAuthFactory$new(consumerKey=consumer_key,consumerSecret=consumer_secret, requestURL=reque
#my_oauth$handshake(cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl"))
#save(my_oauth, file = "my_oauth.Rdata")
load("~/R/github/springboard capstone/my_oauth.Rdata")

botAccts <- lookupUsers(newBotlist)
botAcctlist <- twListToDF(botAccts)
acctNames <- row.names.data.frame(botAcctlist)
#electionTweets <- lapply(acctNames, userStream(file.name = "", with = "user", oauth = my_oauth))

#head(botAccts)
#
# #fromAccts <- rownames(botAcctlist)
# from <- "from:"
# fromAccts <- paste(from, rownames(botAcctlist))
# fromAccts <- gsub(" ", "", fromAccts[])

library(stringr)
#head(fromAccts)

#unlistedBots <- unlist(botAcctlist[]$screenName)
#botdetails <- map(unlistedBots[], ~searchTwitteR(.x, n=100))
#botDF <- twListToDF(flatten(botdetails))
#botRT <- filter(botDF, isRetweet == "TRUE")
#botOPEN <- filter(botDF, isRetweet != "TRUE")
#acctRT <- word(botRT$text, 2)
#unique(as.character(sort(acctRT)))

recentTweetsDS <- map(unlistedBots[], ~searchTwitteR(.x, resultType = "recent", n=200))
recentTweetsDF <- twListToDF(flatten(recentTweetsDS))
botDS <- data.frame(botAcctlist$screenName, botAcctlist$id, botAcctlist$created, botAcctlist$statusesCou
columnnames <- names(botDS)
columnnames[1] <- "screenName"
columnnames[2] <- "ID"
columnnames[3] <- "acct_created"
columnnames[4] <- "statusesCount"
```

```
colnames(botDS) <- columnnames

tweetsRT <- filter(recentTweetsDF, isRetweet == "TRUE")
tweetsOPEN <- filter(recentTweetsDF, isRetweet != "TRUE")
tweeters <- unique(tweetsOPEN$screenName)
length(tweeters)

##> unlist(paste(filter(tweetsOPEN, screenName == "15_margiecastro")$text, collapse = " | "))
##[1] "@mor1019chacha Kamo... | @mor1019chacha Paki tanong naman po @mor1019chacha  kung Pati ba ugali I

#save.image("brooklyn.Rdata")

library(rvest)
#code to log into webpage to submit tweets for LIWC evaluation
txtInspect <- "http://textinspector.com/account/login?redirect=/"
pgsession <-html_session(txtInspect)
pgmform <- html_form(pgsession)[[1]]
filled_form <- set_values(pgmform, 'email' = "milti@mac.com", 'password' = "rf13nn3$")
submit_form(pgsession, filled_form)

#actual form that submits the tweets
workFlow <- "http://textinspector.com/workflow"
wkSession <- html_session(workFlow)
wkForm <- html_form(wkSession)[[1]]
filled_wkForm <- set_values(wkForm, 'text' = unlist(paste(filter(tweetsOPEN, screenName == "15_margieca
result <- submit_form(wkSession, filled_wkForm)

htmlParse(read_html(result$url))
```

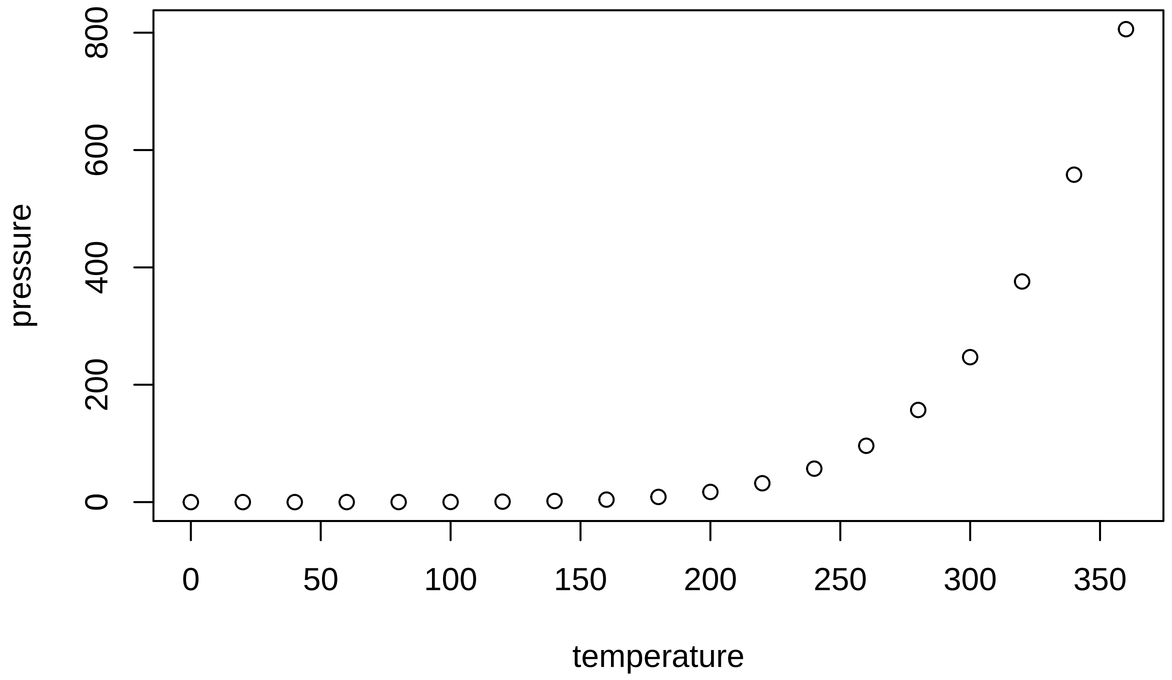
The results will be summarised in a network graph of the accounts in an attempt to detect how the community of resultant accounts are interacting and how the clustering is structured: can this be represented as a single cluster or is there any sequestering and can any interaction between the sequestered groups be found?

steps to continue forward

- [x] Get your API import debot db = debot.DeBot('your_api_key') db.get_related_bots('Charlottesville')
- [x] twitteR
- [x] Search for feeds
- [x] By the username of the bot
- [] Then look for hashtags in those feeds
- [] Now search twitteR again for those hashtags
- [x] This will give you a dataset which contains tweets from both bots and humans
- [] So now we have a labeled dataset which contains both bots and humans this is the startpoint for training a machine learning algo Thereafter we identify a suitable classification algo

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.