

Twitter APIs

T.Kim

Preperation

This course utilizes `rtweet` package for connecting to Twitter APIs and retrieve Twitter data. If you have any questions how to use the function of the package, reading its documantation is the first thing you have to do. You can find the documentation here (<https://cran.r-project.org/web/packages/rtweet/rtweet.pdf>).

Let's install and load the package first.

Install `rtweet`

```
install.packages("rtweet")  
library(rtweet)
```

First, we need to generate a token using the `create_token()` function.

```
# set name of application  
appname <- "your_appname"  
  
# set API keys  
consumer_key <- "your_consumer_key"  
consumer_secret <- "your_consumer_secret"  
access_token <- "your_access_token"  
access_secret <- "your_access_secret"  
  
# handshake authorization  
twitter_token <- create_token(  
  appname,  
  consumer_key,  
  consumer_secret,  
  access_token,  
  access_secret)
```

As long as `twitter_token` is existed in your global environment, `rtweet` functions find the token. Remember, you need to create a token everytime you start a new sessions. The code above is stored in `0_token.R`. Fill your information and execute the code.

REST API

Through REST API, we can retrieve diverse information which is not new. In this exercise introduces four main end points, which are useful for social science research. Use `1_collect_tweets.R` to follow this examples.

Search

You can retrieve recent tweets matching a search query using **Standard Search API**. You can only retrieve tweets published in the **past 7 days**.

Twitter documentation: here (<https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/overview/standard>).

Note that you might miss some tweets. According to Twitter,

...it's important to know that the standard search API is focused on relevance and not completeness. This means that some Tweets and users may be missing from search results. If you want to match for completeness you should consider the premium or enterprise search APIs.

```
t1 <- search_tweets(
  q = "trump", # search query
  n = 1000 # the total number of desired tweets to return. Defaults to 100.
)

head(t1)[,1:5]
dim(t1)
```

Collect more than 18000 tweets

It is often a case that the number of desired tweets are exceed 18,000, which we are allowed to retrieve in 15 min time window. In order to retrieve more than 18,000 tweets in a single call, we have to set `max_id` parameter, `status_id` of the oldest tweets among previous retrieval, along with the query. But thanks to `rtweet`, this process will be automatically handled if we set `retryonratelimit` to `TRUE`.

```
t2 <- search_tweets(
  q = "trump", # search query
  n = 20000 # the total number of desired tweets to return. Defaults to 100.
  retryonratelimit = TRUE # set TRUE if the number of desired tweets exceeds 18,000.
)
```

Work with max_id

There is no exact limits for the number of tweets to search, so theoretically, we can retrieve massive number of tweets, like more than a million, by using this method. But these searches can take days, and it is very likely that you have connections to time out. Thus, I recommend that you breakup data retrieval into smaller chunks and set `max_id` to resume searching where the previous efforts left off. In this example, let's continue collecting tweets older than tweets collected in `t1`.

```
load("collect_tweets.RData")
```

```
# create max_id
max_id <- min(t1$status_id) # oldest status_id
```

```
t3 <- search_tweets(
  q = "trump", # search query
  n = 1000, # the total number of desired tweets to return. Defaults to 100.
  max_id = max_id
)
```

Now join two data.frame, `t1` and `t3`.

```
t4 <- rbind(t1, t3[-1,]) # Quick question: Why do we remove one row here?
dim(t4)
```

```
## [1] 1638 90
```

Parameters

query

We need to set parameter `q` : query to be searched. You can find the detailed explanation for the parameter at Twitter documentation (<https://developer.twitter.com/en/docs/twitter-api/v1/rules-and-filtering/overview/standard-operators>). Although it accepts very long and complicated query (at max 500 characters), it is recommended that you limit your searches to 10 keywords and operators. If query is too complicated you have more chance to get error. So, try to keep it simple! Another important parameter in `search_tweets` function in `rtweet` is `n`, the total number of desired tweets to return. Defaults to 100, i.e., if you do not set `n`, it returns 100 tweets.

```
t5 <- search_tweets(q = "trump", n = 10) # We search 10 tweets containing "trump".
t6 <- search_tweets(q = "biden trump" , n = 10) # containing both "biden" and "trump". It is the same as "donald AND trump".
t7 <- search_tweets(q = "biden OR trump" , n = 10) # containing either "biden" or "trump"
t8 <- search_tweets(q = "biden -trump" , n = 10) # containing "biden" but not "trump"
t9 <- search_tweets(q = "#trump" , n = 10)
```

Geolocalization

You can restrict your query by a location using the geocode parameter specified with the template "latitude,longitude,radius". For example, if you want to search tweets within 100km of Oldenburg University, "37.781157,-122.398720,100km". The parameter radius must be specified as either `mi` (miles) or `km` (kilometers). According to the documentation,

when conducting geo searches, the search API will first attempt to find Tweets which have lat/long within the queried geocode, and in case of not having success, it will attempt to find Tweets created by users whose profile location can be reverse geocoded into a lat/long within the queried geocode, meaning that is possible to receive Tweets which do not include lat/long information.

Note that there are not many users who provides their location, approx 1 to 2% of tweets?

```
# Search tweets within 100km of Oldenburg University
t10 <- search_tweets("trump", geocode = "53.147505,8.181818,100km")
```

Result type

Three result types are available: recent, popular or mix of both tweets. The default of `rtweet` is `type = "recent"`.

```
tweets <- search_tweets(q = "trump", type = "recent") # default
tweets <- search_tweets(q = "trump", type = "popular")
tweets <- search_tweets(q = "trump", type = "mixed")
```

Language

You can restrict your query to the given language, given by an ISO 639-1 (https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes) code.

Time frame

You can restrict your query by time (in 7-day time window). `until` parameter returns tweets created before the given date and `since` parameter return tweets created after the given date. Date should be formatted as YYYY-MM-DD.

```
tweets <- search_tweets(q = "trump", lang = "en", since = "2020-12-03", until = "2020-12-04")
```

Exercise

Write Search codes to collect following tweets:

1. 100 recent tweets mentioning Twitter account of President Donald Trump. *Hint: Check Twitter documentation: here (<https://developer.twitter.com/en/docs/twitter-api/v1/rules-and-filtering/overview/standard-operators>)*
2. 10 recent tweets written in English, containing both words “rigged” and “election”.
3. 10 tweets written in English, containing both words “rigged” and “election”, published yesterday.
4. 10 recent tweets written in English, containing exact phrase “rigged election”.
5. 30 recent tweets containing word “trump”, written in German.
6. 30 tweets replying to Donald Trump’s account.

User timeline

You can retrieve a user's recent tweets, at most 3,200. You need to specify a user using the `screen_name` or `user_id` parameter. Note that you can retrieve more than one user's posts in a single call. 900 requests are allowed in 15-min window and it returns 200 tweets per request. Thus, you can collect 180,000 tweets at maximum in 15 min. But be careful that this endpoint there is a rate limit for 24-hour window, 100,000 requests.

```
trump_tweets <- get_timeline("realDonaldTrump", n = 3200)
biden_trump_tweets <- get_timeline(c("realDonaldTrump", "JoeBiden"), n = 3200)
```

Exercise

Collect and save following tweets:

1. Collect 3000 each tweets from following three party accounts: "AfD", "CDU", "spdde".
2. Save (1)'s tweets using `RData` format. Use file name "party_timeline.RData".
3. Collect 3000 each tweets from Donald Trump's account and Joe Biden's account.
4. Store (3)'s tweets using 'RData' format. Use file name "trump_biden_timeline.RData".

Followers

You can retrieve a list of user IDs for the accounts following specified user. To return more than 75,000 user IDs in a single call, `retryonratelimit` must be set to `TRUE`. Note that this endpoints allows to retrieve 75,000 IDs in 15 min. In case desired number of IDs exceeds 75,000 and you want to break up retrieving data, you can use `next_cursor()` function.

```
trump_flw_ids <- get_followers("realDonaldTrump", n = 20000, retryonratelimit = TRUE) # specify user screen_name or user_id
try1 <- get_followers("realDonaldTrump", n = 75000)
try2 <- get_followers("realDonaldTrump", n = 75000, page = next_cursor(try1))
```

User object

You can retrieve 90,000 twitter users objects, Twitter User account metadata, in 15 min. To collect more than 90,000 users, you need to request after 15 min to avoid rate limits. For more information about User object, see here (<https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/overview/user-object>).

```
users <- as.matrix(trump_flw_ids)[,1] # create a vector containig a list of user ids
trump_flw_user <- lookup_users(users) # user screen_name or user_id
```

Exercise

1. Collect 100 user id of Donald Trump's followers.
2. Collect user objects of (1)'s users.

Streaming API

By using the function `stream_tweets()`, you can collect realtime tweets. If you don't set any parameter, you receive a sample of all publicly available tweets. Mainly there are three ways to restrict stream:

Track tweets containing specified keywords

You can filter stream by specifying keywords. Commas are logical ORs, while spaces are equivalent to logical ANDs. Non-space separated languages, i.e., Chinese, Japanese, and Korean are not supported.

```
stream_sample <- stream_tweets() # Collect available sample tweets for 30 seconds
stream_trump <- stream_tweets("trump", timeout = 10) # collect tweets mentioning t
rump
stream_trump_biden <- stream_tweets("trump biden", timeout = 10)
stream_mention_trump <- stream_tweets("realdonaldtrump", timeout = 30) ## stream t
weets mentioning Trump's account for 30 seconds
```

Follow specified user's activity

You can follow up to 5,000 users activity at once, such as posting tweets and retweeting, replying etc, by specifying their `user_id` (not `screen_name` !)

```
user <- lookup_users("realDonaldTrump")
user_id <- user$user_id
stream_trump_follow <- stream_tweets(q = user_id, timeout = 60)
```

Location

By specifying a set of bounding boxes, pairs of longitude and latitude, you can filter Tweets by location. Only geolocated Tweets falling within the bounding boxes will be retrieved. Note that the user's location field is not used to filter Twitter. To specify bounding boxes, you can use websites such as bboxfinder.com (bboxfinder.com). `rtweet` function `lookup_coords` provides bounding boxes of major cities around the world.

```
head(rtweet::citycoords, 30)
```

```
##           city      lat      lng
## 1   aberdeen scotland 57.15000 -2.150000
## 2   aberdeen          57.15000 -2.150000
## 3   aberdeen scotland 57.15000 -2.150000
## 4   adelaide australia -34.91667 138.600000
## 5   adelaide          -34.91667 138.600000
## 6   adelaide australia -34.91667 138.600000
## 7   algiers  algeria  36.83333   3.000000
## 8   algiers          36.83333   3.000000
## 9   algiers  algeria  36.83333   3.000000
## 10  amsterdam netherlands 52.36667   4.883333
## 11  amsterdam          52.36667   4.883333
## 12  amsterdam netherlands 52.36667   4.883333
## 13  ankara  turkey   39.91667  32.916667
## 14  ankara          39.91667  32.916667
## 15  ankara  turkey   39.91667  32.916667
## 16  asunción paraguay -25.25000 -57.666667
## 17  asunción          -25.25000 -57.666667
## 18  asunción paraguay -25.25000 -57.666667
## 19  athens  greece   37.96667  23.716667
## 20  athens          37.96667  23.716667
## 21  athens  greece   37.96667  23.716667
## 22  auckland new zealand -36.86667 174.750000
## 23  auckland          -36.86667 174.750000
## 24  auckland new zealand -36.86667 174.750000
## 25  bangkok thailand  13.75000 100.500000
## 26  bangkok          13.75000 100.500000
## 27  bangkok thailand  13.75000 100.500000
## 28  barcelona spain   41.38333   2.150000
## 29  barcelona          41.38333   2.150000
## 30  barcelona spain   41.38333   2.150000
```

```
osaka <- rtweet::lookup_coords("osaka japan")
print(osaka)
```

```
## $place
## [1] "osaka japan"
##
## $box
## sw.lng.lng sw.lat.lat ne.lng.lng ne.lat.lat
## 135.45000 34.48333 135.55000 34.58333
##
## $point
## lat lng
## 34.53333 135.50000
##
## attr(,"class")
## [1] "coords" "list"
```

```
stream_osaka <- stream_tweets(osaka, timeout = 10)
```

In addition to this, there are more ways to filter tweets. For more detail, see Twitter documentation, here (<https://developer.twitter.com/en/docs/twitter-api/v1/tweets/filter-realtime/guides/basic-stream-parameters>).

Work with json file

When you collect tweets using streaming API for hours, the size of data tend to get large quickly. It is not recommended that you connect streaming API for long hours. I recommend you connect streaming API less than an hour and save the data, and reconnect again. Additionally, `rtweet` let you store tweets collected from streaming API a json file and parse to `data.frame` format afterwards. Since parsing tweets takes time, and you won't lose any data when you store them into json file. You can do it in following way (the exemplary code are taken from `rtweet` documentation):

```
stream_tweets(timeout = (60 * 10), parse = FALSE, file_name = "tweets1")  
## parse tweets at a later time using parse_stream function  
tw1 <- parse_stream("tweets1.json")
```

Exercise

1. Collect tweets containing following keywords “rigged” for 30 seconds.
2. Track two users activity, Donald Trump and Joe Biden, for 10 seconds.
3. When you track users, does it return tweets mentioning the users? Read Twitter documentation and find out: here (<https://developer.twitter.com/en/docs/twitter-api/v1/tweets/filter-realtime/guides/basic-stream-parameters>).