DATA INTENSIVE COMPUTING

Data Acquisition

Ramanpreet Singh Khinda | DIC 587 | March 5, 2016
rkhinda@buffalo.edu
50169622

Problem 1A: Fetching data using R Twitter Streaming API

Step 1:

To use the R Twitter Streaming API, we will need to install the following packages:

```
# Installing the required packages
install.packages("streamR")
install.packages("ROAuth")

# Loading the installed packages
library(ROAuth)
library(streamR)
```

streamR provides a series of functions that allow R users to access Twitter's filter, sample, and user streams, and to parse the output into data frames

ROAuth provides an interface to the OAuth 1.0 specification allowing users to authenticate via OAuth to the server of their choice.

Step 2:

In the next step we need to set up an OAuth handshake. This is accomplished using the ROAuth package. This will redirect to Twitter OAuth screen. Once you click on authorize, copy the pin provided by Twitter into the R console and hit enter.

```
# CreatingOAuth Credentials
credential <- OAuthFactory$new(consumerKey='ZeyLvp3eud3K9a1JBPnvqmRQh',
                               consumerSecret='O59dRWMXrP6o1CLxSNN3etNCJlshJ5tTZyK8WazmNogIPQmao5'
                               requestURL='https://api.twitter.com/oauth/request_token',
                               accessURL='https://api.twitter.com/oauth/access_token',
                               authURL='https://api.twitter.com/oauth/authorize')

# Authentication Process
options(RCurlOptions = list(cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl")))
download.file(url="http://curl.haxx.se/ca/cacert.pem", destfile="cacert.pem")
credential$handshake(cainfo="cacert.pem")
```

Step 3:

Now we can start collecting data. This is done by using the filterStream function within the streamR package.

```
# Function to scrape Twitter

# Scrapping data for Manhattan
filterStream( file.name="Tweets_Manhattanj.json", track="manhattan", tweets=100, oauth=credential, timeout=10, lang='en' )

# Scrapping data for Manhattan
filterStream( file.name="Tweets_Brooklyn.json", track="brooklyn", tweets=100, oauth=credential, timeout=10, lang='en' )

# Scrapping data for Manhattan
filterStream( file.name="Tweets_Bronx.json", track="bronx", tweets=100, oauth=credential, timeout=10, lang='en' )

# Scrapping data for Manhattan
filterStream( file.name="Tweets_StatenIsland.json", track="staten island", tweets=100, oauth=credential, timeout=10, lang='en' )
```

The filterStream function takes the following parameters:

- file.name = the name of the file where tweets will be written to.
- track = a string containing keywords to track.
- follow = string or vector containing twitter user IDs if we want to only track tweets of specific users.
- locations = a vector of latitude and longitude pairs (southwest corner coming first) specifying a set of bounding boxes to filter incoming tweets.
- language = a list of BCP 47 language identifiers.
- timeout = in seconds, the max length of time to connect to the stream. Setting a timeout = 10 will end the connection after 10 seconds. Default is 0, which means that the connection is always on.
- tweets = number of tweets to collect. For example, if you only want to collect 100 tweets, you could leave timeout = 0, and specify tweets = 100. The connection would end after the 100th tweet is collected.
- oauth = object where we specify our oauth setup (my_oauth).
- verbose = can be TRUE or FALSE, generates output to the R console with information about the capturing process.

Time Period for Data Collection

From Feb 27 to March 4

The data was collected for a period of 1 week. Since there were not much talk on twitter regarding apartments in Manhattan, Brookly, Bronx and StatenIsland, I could not collect much tweets. But still till the end of week I was able to get more than 1000 unique tweets.

Problem 1B: Importing JSON into R from file

Step 1:

In the first step we will load the required libraries

```
1  # loading the installed packages
2  library(streamR)
3  library(rjson)
```

Step 2:

In the next step we will set our working directory to the path where the json files are saved

```
5  # Setting the current working directory
6  setwd("/Users/raman/Documents/SUNY_BUFFALO/DIC_CSE_587/Project_1/Problem_1")
```

Step 3:

Next we will create a function which will do the parsing and cleaning of tweets

```
8  # Function to parse the tweets
9  get_tweets_df<-function(filename, data_location_name){
10   tweets <- parseTweets(tweets=filename)
11
12   # Capturing only information of interest and creating data frame out of it
13   df_tweets <- data.frame(tweets$id_str, tweets$created_at, tweets$text, tweets$favourites_count,
14                           tweets$retweet_count, tweets$user_id_str, tweets$verified,
15                           tweets$followers_count, tweets$friends_count, tweets$listed_count,
16                           tweets$statuses_count, tweets$location)
17   df_tweets$data_location <- data_location_name
18
19   # Adding intuitive column names
20   colnames(df_tweets) <- c("tweet_id","tweet_created_at","tweet_text","tweet_favorite_count",
21                            "tweet_retweet_count","user_id","user_verified","user_followers_count",
22                            "user_friends_count","user_listed_count","user_statuses_count",
23                            "user_location","data_location")
24
25   # Capturing only unique tweets
26   df_tweets <- unique(df_tweets)
27
28   return(df_tweets)
29 }
```

We are also creating data frame out of the tweets using data.frame() function. This function creates data frames, tightly coupled collections of variables which share many of the properties of matrices and of lists, used as the fundamental data structure by most of R's modeling software.

Also when dealing with twitter data we can run into situation where many of the collected tweets are similar. To avoid this scenario we used the unique() function. This function returns a matrix with duplicated rows (or columns) removed

Step 4:

In the next step we will start fetching tweets from the stored JSON files

```
# Fetching data for Manhattan
df_manhattan <- get_tweets_df('Tweets_Manhattan.json', "Manhattan")
head(df_manhattan)

# Fetching data for Brooklyn
df_brooklyn <- get_tweets_df('Tweets_Brooklyn.json', "Brooklyn")
head(df_brooklyn)

# Fetching data for Bronx
df_bronx <- get_tweets_df('Tweets_Bronx.json', "Bronx")
head(df_bronx)

# Fetching data for Staten island
df_staten_island <- get_tweets_df('Tweets_StatenIsland.json', "Staten Island")
head(df_staten_island)
```

Here we are using a function called head(<object>). This function returns the top 6 elements in the object (which is data frame in our case).

References:

Tutorials followed to understand Twitter Streaming API's

http://bogdanrau.com/blog/collecting-tweets-using-r-and-the-twitter-streaming-api/

http://pablobarbera.com/blog/archives/1.html