NLP - Assignment 4

In this assignment you will pimp your twitter streamer, stream tweets for an extended period of time and analyse language use across the United States.

1 Pimp you streamer

To allow for extended streaming, program a revolving streamer that iterates through your 3 sets of access keys.

1.1 Create 3 Oauth objects

Follow the procedure of last's assignment to create 3 working Oauth objects and store them in a `list()`.

```
# load packages
library(ROAuth)
library(streamR)
```

```
## Loading required package: RCurl
```

```
## Loading required package: bitops
```

```
## Loading required package: rjson
```

```
# define keys
keys = c('VLN7sAK5PLg3vvR6uJ71jKcpD',
         'RTtHlXYeXz2ZE51co9jJaT8utbfhBaGJziS2BYL6a0qLsfFFpW',
         '832990021879726085-yFJabNKfxmANCxtxxscecr7326fnlsJ',
         '2UPpDVkjpecr5R9Y2WY9zIr3DHm53VJ104jma2f7Hf8Up')

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

# create OAuth
oauth = OAuthFactory$new(
  consumerKey=keys[1],
  consumerSecret=keys[2],
  requestURL=requestURL,
  accessURL=accessURL,
  authURL=authURL)
oauth$oauthKey = keys[3]
oauth$oauthSecret = keys[4]
oauth$handshakeComplete = TRUE
oauth$signMethod = "HMAC"

# OAuth list (in your case 3 or 4 OAuths)
oauths = list(oauth, oauth, oauth)
```

1.2 Loop through Oauth objects

Write a loop that streams tweets while alternating between the 3 Oauth objects. To do this, set up a loop that iterates from, e.g., 1 to 100, and each time initiates a new streaming attempt. Select the Oauth object using `i %% 3 + 1`, where `i` is the index of the loop. `%%` is the modulus operator which divides the number to the left of the operator by the number to the right of it and returns the whole-number rest.

For every iteration execute the `filterStream()`-function inside a `try()`-function. This time set the `file.name`-argument to an actual filepath and file, and adapt it to reflect the current iteration index. E.g., set it to `paste0('_streams/','stream_',i,'.txt')`, which for the first iteration becomes `paste0('_streams/stream_1.txt')`, for the second iteration `paste0('_streams/stream_2.txt')`, and so on. Check out the `paste0()` help file. Make sure that everything before the final slash `/` actually exists as folders on your hard drive (otherwise you will get an `could not open connection` error). Also, make sure you use a folder that otherwise contains no other files.

To study the regionality of language choose two words, e.g., *dinner* and *supper*, and provide them separated by comma as the track argument, e.g., `dinner,supper`. Also set the coordinates to `c(-125, 30, -65, 50)` to include only tweets originating in the US. Set the timeout to, e.g., 300 seconds.

If you want to be 100% sure that Twitter won't cut you off, include additionally an `Sys.sleep(runif(120,600))` at the end of each iteration to pause the code for some randomly determined duration between 2 and 10 minutes.

```
# define parameters
track  = 'supper,dinner'
n_runs = 3 # increase this
run_time = 5 # increase this
wait_time = c(1, 6)

# run loop over streamers
for(i in 1:n_runs){

  # keep track
  # cat('Run',i)

  # select oauth
  oauth = oauths[[i %% 3 + 1]]

  # stream
  try(filterStream(
    file.name = paste0('~/Desktop/stream/stream_',i,'.txt'),
    track = track,
    language = 'en',
    oauth = oauth,
    locations = c(-125, 30, -65, 50),
    verbose = F,
    timeout = run_time))

  # wait
  Sys.sleep(runif(1, wait_time[1], wait_time[2]))
  }
```

2 Gather data

Extract the file paths (including names) of the tweet data sets using `list.files()`. Set `full.names` to `TRUE` to receive the full file paths. Iterate over the file paths, read the data using `readLines()`, and put it into te

ith position of a list. Then `unlist()` the list to obtain a character vector containing all of your streamed tweets.

```
# create container
tweets = list()

# get filenames
files = list.files('~/Desktop/stream/', full.names = TRUE)

# iterate over files
for(i in 1:length(files)) tweets[[i]] = readLines(files[i])

# unlist list
tweets = unlist(tweets)
```

3 Process data

Process the data the same way you processed it before. This time, however, also extract whether `tweet$text` contains one or the other track term using `str_detect()`. Note that the streamer will return tweets that match the `track` argument in other elements of the tweet than merely the text. Post the result on twitter.

```
# get stringr & jsonlite
require(stringr)
```

```
## Loading required package: stringr
```

```
require(jsonlite)
```

```
## Loading required package: jsonlite
```

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

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

```
# number of tweets
n_tweets = length(tweets)

# prepare matrix
res_mat = matrix(nrow = n_tweets, ncol = 3)

# iterate through twets
for(i in 1:n_tweets){

  # parse tweet
  tweet = try(fromJSON(tweets[i]))

  # get text and coordinates
  text = tweet$text
  coordinates = tweet$place$bounding_box$coordinates

  # tests
  if(!is.list(tweet)) next
  if(is.null(text)) next
  if(!str_detect(text, 'supper|dinner')) next
```

```
  if(is.null(coordinates)) next

  # store hit
  res_mat[i, 1] = str_detect(tweet$text, 'dinner')

  # process coordinates
  coordinates <- coordinates[1,c(1,3),]
  xy = c(coordinates[1] + diff(coordinates[1:2]) / 2,
         coordinates[3] + diff(coordinates[3:4]) / 2)

  res_mat[i, 2:3] = xy

  }
```

4 Plot tweets

Plot again a map of the tweets and color the points according to which track term was included in the tweet text. One easy way of doing this is via, e.g., `ifelse(res_mat[,1], 'blue', 'red')`, which will color the point blue, if the test (stored in the first column of the result matrix) is TRUE and red otherwise. Since you streamed tweets from the US use `map('state)` rather than `map()`.

```
# get maps
require(maps)
```

```
## Loading required package: maps
```

```
# map
map('state')
points(res_mat[, -1], pch = 16, col = ifelse(res_mat[,1] == 0, 'blue', 'red'))
```