NLP - Assignment 3

In this assignment you will create a twitter app, use it to stream tweets, parse the tweets and store the result. Note: Hyperlinks are highlighted using bold font.

1 Acquire authorization keys

In order to be able to access twitter from within R, you need to create twitter apps that provide you with the necessary authorization keys. To this end, create 3 new Email accounts that you use to create 3 new twitter acconts that you use to create 3 new twitter apps. Details below.

1.1 Email account

Open 3 new Email accounts (e.g., with **Gmail**) and store login and pass in a text file (e.g., using WordPad or TextEdit).

1.2 Twitter account

Open 3 new **twitter account** using your newly created Email accounts. Store login and pass in the text file with the mail account credentials. It does not matter how you name your account. Important: Verify the accounts using your the SIM cards that you ordered.

1.3 Twitter App

Create 3 **twitter app**. Come up with an app name and description and use http://www.dirkwulff.org in the website field.

1.4 Extract auhorization keys

Go to *Keys and Access Tokens* tab and copy the *Consumer Keys*, *Consumer Secrets*, *Access Tokens*, and *Access Token Secrets* into your text file.

2 Stream Twitter

In RStudio open a new R script.

2.1 Install `ROAuth` and `streamR`

Install and load packages `ROAuth` and `streamR` using `install.packages()` and `library()`.

```r
#install.packages('ROAuth')
#install.packages('streamR')
library(ROAuth)
library(streamR)
```

```
## Loading required package: RCurl

## Loading required package: bitops

## Loading required package: rjson
```

2.2 Setup OAuth

Setup OAuth using the `OAuthFactory$new()` function. Provide the consumer key and secret of your first app, as well as the following URLs to the respective arguments of `OAuthFactory$new()` (see `?OAuthFactory`). Assign the output to `my_oauth`:

- https://api.twitter.com/oauth/request_token
- https://api.twitter.com/oauth/access_token
- https://api.twitter.com/oauth/authorize

Next, make the following changes to the `my_oauth` object. Instead of `'your_access_token'` and `'your_access_token_secret'` use the respective keys provided by your twitter app:

```
  my_oauth$oauthKey = 'your_access_token'
  my_oauth$oauthSecret = 'your_access_token_secret'
  my_oauth$handshakeComplete = TRUE
  my_oauth$signMethod = "HMAC"
```

```
# define keys
keys = c('VLN7sAK5PLg3vvR6uJ71jKcpD',
         'RTtH1XYeXz2ZE51co9jJaT8utbfhBaGJziS2BYL6a0qLsfFFpW',
         '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"
```

2.3 Test streamer

Use `filterStream()` to stream tweets (see `?filterStream`) using the following arguments:

Argument	Description
`file.name`	Set this to `""` so that the tweets are returned to R, rather than saved on the hard drive.
`locations`	Expects a numeric vector specifying south-west and north-east corners of a target reqion in longitude and latitude. Use `c(-180, -90, 180, 90)` to include the entire world.
`track`	The function will return tweets that contain the `character` string assigned to this argument. Choose whatever you like.
`language`	Set this to `'en'` to stream tweets of English language. Alternatively, choose `'de'` for German, `'es'` for Spanish, etc.

Argument	Description
`oauth`	Assign to this argument your oauth objects
`timeout`	This argument determines for how long the streamer will attempt to stream tweets. To begin with set this to some low number, e.g., 10, to stream tweets for 10 seconds.

Assign the output of `filterStream()` to an object. When the streaming is done, inspect the result.

More info on parameters **here**.

```r
streamed_tweets = filterStream(
  file.name = "",
  locations = c(-180, -90, 180, 90),
  track = 'trump',
  language = 'en',
  oauth = oauth,
  timeout = 300)
```

```
## Capturing tweets...
```

```
## Connection to Twitter stream was closed after 300 seconds with up to 7931 tweets downloaded.
```

2.4 Stream

Stream for a longer period, minutes or even hours. Twitter will at some point cut you off, when it thinks that you have downloaded enough tweets. When this will happen, depends largely on your `track` argument. If it matches many tweets then streaming will be stopped earlier.

4 Processing Tweets

4.1 Install `jsonlite`

Install and load `jsonlite` and `maps`. You know how.

```r
#install.packages('jsonlite')
#install.packages('map')
library(jsonlite)
```

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

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

```r
library(maps)
```

4.2 Parse JSON

Using `matrix()`, create an empty matrix with two columns and as many rows as there tweets. By iterating over the tweets (using `for()`), parse each tweet using `fromJSON()` and extract the tweets coordinates. The output from `fromJSON` is a list containing lists, which means that you need to search through the object in order to find the coordinates. This should provide some **guidance**. For completing this task, there are, unfortunately, two problems to be dealt with:

Problem 1: Some tweets are improperly formatted resulting in errors when applying `fromJSON()`. To deal with this problem place the function inside a `try()`-function, i.e., `try(fromJSON())`. This will allow the function to execute despite potential errors. Then test whether the output from `try(fromJSON())` was a list or not test using `is.list()` using an `if()`-statement and `next`, when the output is not a list. I.e., you want to include something along the lines of `if(!is.list(parsed_tweet)) next`, which will jump to the next iteration, whenever the output is *not* a list (note the negation using `!`).

Problem 2: Some tweets do not contain coordinate information. This means that have you test whether the coordinate object is `NULL` using `is.null()` and only proceed (for the current tweet) when the object is not `NULL`. I.e., here you want to include something along the lines of `if(is.null(coordinates)) next`.

When you have extracted existing coordinates use the following code to condense the coordinates (originally an array describing a box around the location) to a single pair of xy-coordinates and store the pair in the matrix.

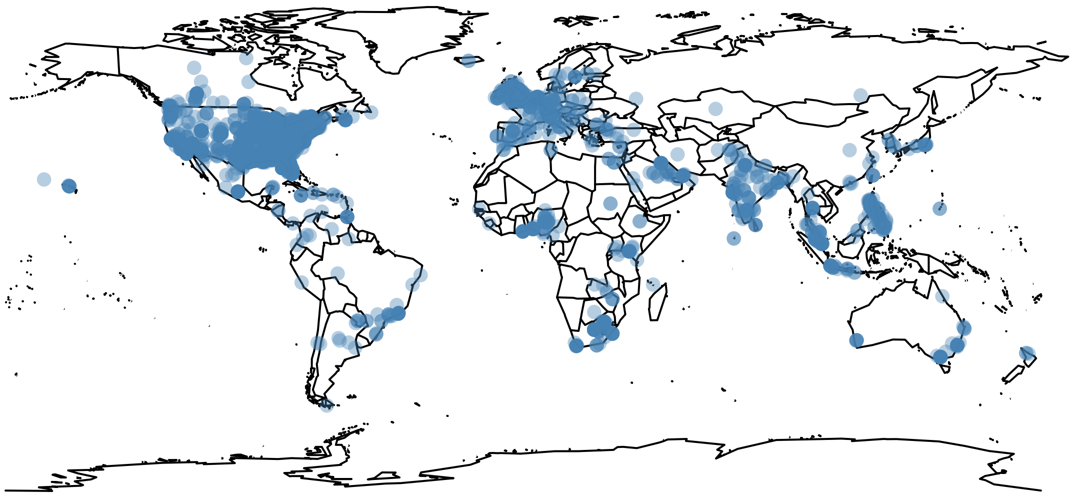
```
  coordinates <- coordinates[1,c(1,3),]
  xy = c(coordinates[1] + diff(coordinates[1:2]) / 2,
         coordinates[3] + diff(coordinates[3:4]) / 2)
```

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

# prepare matrix
xy_mat = matrix(nrow = n_tweets, ncol = 2)

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

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

  if(!is.list(tweet)) next

  # extract coordinates
  coordinates = tweet$place$bounding_box$coordinates

  # test if coordinates are present
  if(!is.null(coordinates)){

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

    xy_mat[i, ] = xy
  }

  }
```

4.3 Create user map

Plot a world map using `map()` (no arguments needed). Add points by passing on the matrix of xy-coordinates to `points()`. Play around with the arguments `pch`, `col`, etc. and when you are happy post the result on twitter using **#nlpbasel**.

```r
map()
points(xy_mat, pch = 16, col = rgb(70, 130, 180, maxColorValue = 255, alpha = 100))
```



End