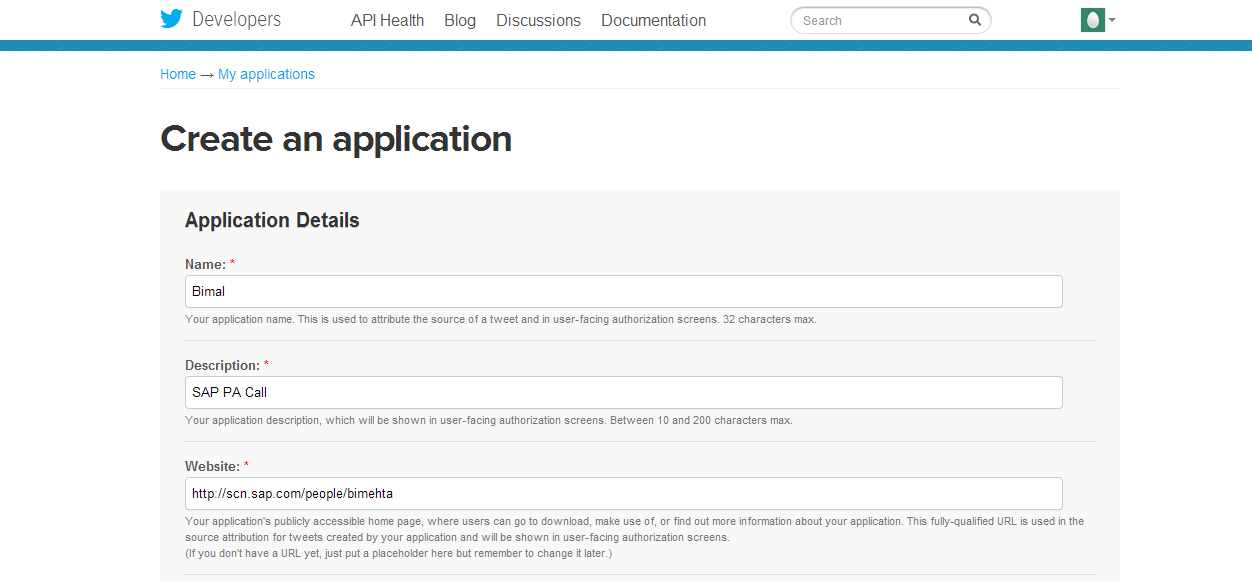
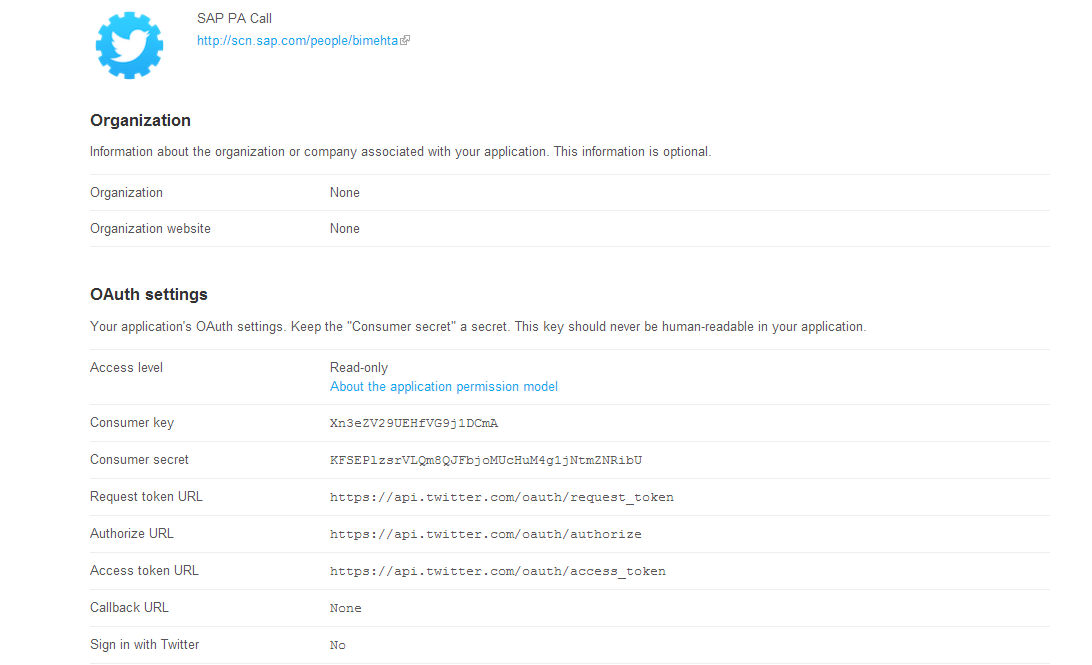
Step1: Setting up the Twitter account and API for handshake with R

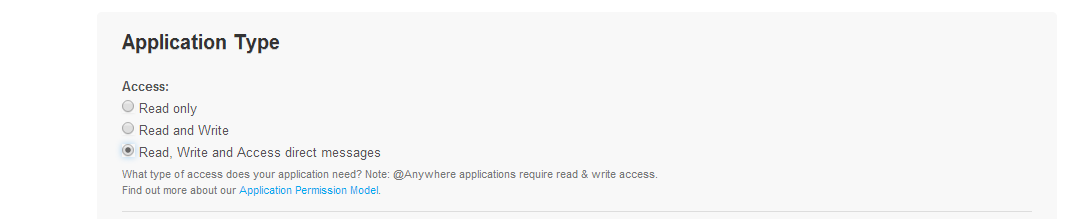
First get a twitter account if you don’t have one by registering on twitter.com website. Next go to (<https://apps.twitter.com>) with the same credentials. Here we will be setting up the API. This API is required for authentication to search tweets from a third party application. It uses an industry standard process called OAuth. OAuth creates the handshake between twitter and R using something called as “Consumer Key” and “Consumer Secret”.

First step is to create an Application. Go to ->My Applications and click “Create New Application”. Fill up your relevant details as per sample shown below. Click the check box that says “Allow this application to be used to Sign In with Twitter”.



Once the application is saved you get the below screen with your Consumer Key and Consumer Secret. Finally go to the Settings tab and make sure that "Read, Write and Access direct messages" is set. Using "OAuth tool" tab you can come back to get your Consumer key and Consumer secret information when required.





Step2 : Install the necessary packages and set up the handshake.

We will need the following packages for extracting the tweets. twitteR, RJSONIO, bitops and RCurl.

Install them and load them in the workspace you are working on.

*library(twitteR)*

*library(RJSONIO)*

*library(bitops)*

*library(RCurl)*

For Windows user we need to provide new SSL certificates required for secure communication over the internet. Using below command downloads new certificates and saves them within the current working directory for R. The other statement will set the option globally and will be used in all RCurl calls.

*download.file(url="http://curl.haxx.se/ca/cacert.pem",destfile="cacert.pem")*

*options(RCurlOptions = list(cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl")))*

Next step is to set the credentials for login to twitter. For this we require the “Consumer Secret” and “Consumer Key” that we set up in step 1. Use the below URL and eneter the Key and Secret where mentioned. This will store all the details in the credential object which will be saved in your local workspace.

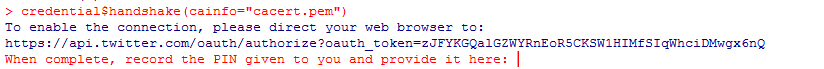
*logindetails<-*

*OAuthFactory$new(consumerKey="Xn3eZV29UEHfVG9j1DCmA",consumerSecret="KFSEPlzsrVLQm8QJFbjoMUcHuM4g1jNtmZNRibU",requestURL="https://api.twitter.com/oauth/request\_token",accessURL="https://api.twitter.com/oauth/access\_token",authURL="https://api.twitter.com/oauth/authorize")*

Next we setup the handshake by entering the below command. On doing that we get the screen shown below:

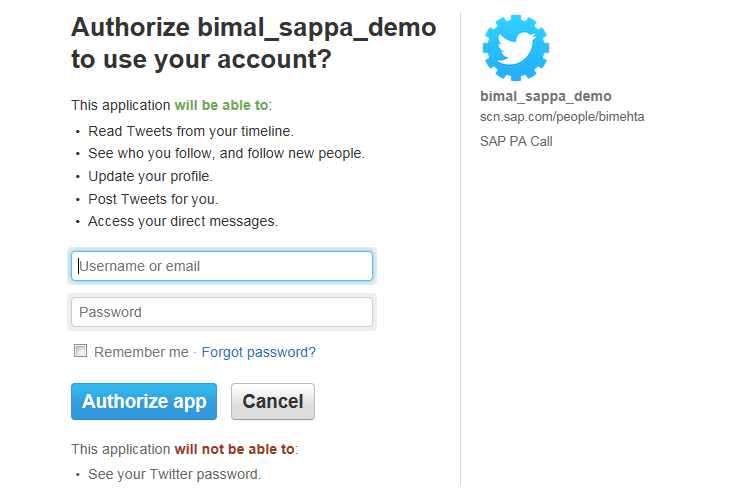
*logindetails$handshake(cainfo="cacert.pem")*

On doing that we get the screen shown below:

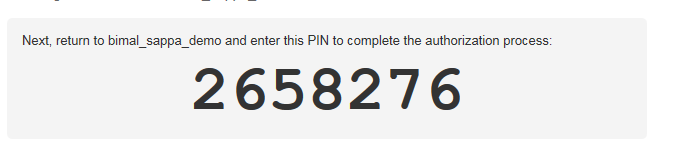


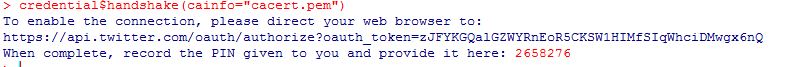
It shows a URL with an authorization token. We need to copy that URL as it is and go to that URL from a web browser.

If done correctly it will take you to your twitter account (you may need to login)



Once logged in you will receive a new screen with a PIN on it (my PIN had seven digits). Take that PIN type them into the R-console window





That was the last step. The credentials are in place now. We can use “logindetails” in our SAP PA to do some text analysis.

For checking if the connection is working just put below command and see if you get true

*registerTwitterOAuth(logindetails)*

*options(RCurlOptions = list(cainfo = system.file("CurlSSL", "cacert.pem", package = "RCurl")))*

The return value of TRUE shows that the login is working and ready to help you get data from Twitter.

Try and fetch some tweets.

*tweetList <- searchTwitter("#Audi", n=100)*

*Audi.df = twListToDF(tweetList)*

*write.csv(climate.df, file="FIFA.csv", row.names = F)*

*save(credential, file="twitter authentication.Rdata")*

*r\_stats\_text <- sapply(tweetList, function(x) x$getText())*

*install.packages("wordcloud")*

*install.packages("tm")*

*library(wordcloud)*

*library(tm)*

*#create corpus*

*r\_stats\_text\_corpus <- Corpus(VectorSource(r\_stats\_text))*

*#clean up*

*r\_stats\_text\_corpus <- tm\_map(r\_stats\_text\_corpus, tolower)*

*r\_stats\_text\_corpus <- tm\_map(r\_stats\_text\_corpus, removePunctuation)*

*r\_stats\_text\_corpus <- tm\_map(r\_stats\_text\_corpus, function(x)removeWords(x,stopwords()))*

*wordcloud(r\_stats\_text\_corpus)*

hu.liu.pos = scan('C:/Users/bimehta/Desktop/Predictive/Text Mining & SA/positive-words.txt', what='character', comment.char=';')

hu.liu.neg = scan('C:/Users/bimehta/Desktop/Predictive/Text Mining & SA/negative-words.txt', what='character', comment.char=';')

#Score all tweets

Audi.scores = score.sentiment(Audi.df$text, pos.words,neg.words, .progress='text')

sum(Audi.scores$score)

R by example: mining twitter for consumer attitudes towards airlines.

<http://txcdk.unt.edu/iralab/how>

<http://txcdk.unt.edu/iralab/sites/default/files/Baseballtweets2.R>

<http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html>

A list of positive and negative opinion words or sentiment words for English (around 6800 words). This list was compiled over many years starting from our first paper (Hu and Liu, KDD-2004)

Lexicon of Hu and Liu

packages- plyr –splitting, applying and combining data/

packages – stringr

packages – ggplot2

pos.words = c(hu.liu.pos, 'upgrade')

neg.words = c(hu.liu.neg, 'wtf', 'wait','waiting', 'epicfail', 'mechanical')

score.sentiment = function(sentences, pos.words, neg.words, .progress='none')

{

require(plyr)

require(stringr)

# we got a vector of sentences. plyr will handle a list

# or a vector as an "l" for us

# we want a simple array ("a") of scores back, so we use

# "l" + "a" + "ply" = "laply":

scores = laply(sentences, function(sentence, pos.words, neg.words) {

# clean up sentences with R's regex-driven global substitute, gsub():

sentence = gsub('[[:punct:]]', '', sentence)

sentence = gsub('[[:cntrl:]]', '', sentence)

sentence = gsub('\\d+', '', sentence)

# and convert to lower case:

sentence = tolower(sentence)

# split into words. str\_split is in the stringr package

word.list = str\_split(sentence, '\\s+')

# sometimes a list() is one level of hierarchy too much

words = unlist(word.list)

# compare our words to the dictionaries of positive & negative terms

pos.matches = match(words, pos.words)

neg.matches = match(words, neg.words)

# match() returns the position of the matched term or NA

# we just want a TRUE/FALSE:

pos.matches = !is.na(pos.matches)

neg.matches = !is.na(neg.matches)

# and conveniently enough, TRUE/FALSE will be treated as 1/0 by sum():

score = sum(pos.matches) - sum(neg.matches)

return(score)

}, pos.words, neg.words, .progress=.progress )

scores.df = data.frame(score=scores, text=sentences)

return(scores.df)

}