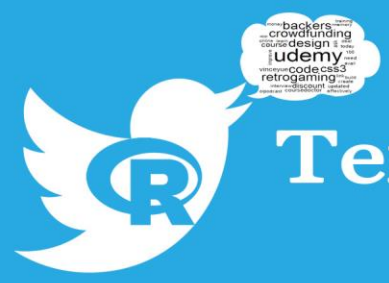


Text mining, Web scraping and Sentiment Analysis with R

BY R-TUTORIALS.COM

Twitter is a great source for sentiment data and social media mining furthermore it is quite easy to get significant amounts of data to be able to scrape data from Twitter you need a standard Twitter account and you need to update it to a developer account

- note that Twitter limits the amount of searches you can perform (15min: 15 scrapes)

- **package twitteR**

```
library("twitteR")
```

- all this info is obtained for the Twitter developer account:
  - key = "your key"
  - secret = "your secret"

- set a working directory for the whole process - you need to download a few files and R needs to know where to look for that stuff

```
setwd("C:/Users/HCBM/Desktop/Mining Course")
```

- this is a crucial step - at least for windows users
- if you are on Linux or Mac you might skip this step, Win needs the certificate collection
- Cacert.pem is a collection of certificates

```
download.file(url="http://curl.haxx.se/ca/cacert.pem",
              destfile="C:/Users/HCBM/Desktop/Mining Course/cacert.pem",
              method="auto")
```

- hint: download.file is really handy when it comes to downloading material from the web
- the dest file is the location on your computer, here it is my working directory
- and url points to the place where you want to get the file from

- we are entering the whole Twitter API info and call the whole object authenticate

```
authenticate <-  OAuthFactory$new(consumerKey=key,
                  consumerSecret=secret,
                  requestURL='https://api.twitter.com/oauth/request_token',
                  accessURL='https://api.twitter.com/oauth/access_token',
                  authURL='https://api.twitter.com/oauth/authorize')
```

- this will get you to a Twitter Site - obtain the PIN
- the whole process is meant to provide the signature for your Twitter usage

```
authenticate$handshake(cainfo="C:/Users/HCBM/Desktop/Mining Course/cacert.pem")
```

- insert the PIN from Twitter

```
4264654
```

```
save(authenticate, file="twitter authentication.Rdata")
```

```
registerTwitterOAuth(authenticate)
```

Let's start with the Twitter scraping!

```
library("twitteR")
```

- we need to specify the cainfo to avoid a SSL cert error - this is for Windows machines

Let's check the latest tweets of Udemy

```
userTimeline("Udemy", cainfo="cacert.pem")
```

- searchTwitter is the main function of the package

```
?searchTwitter
```

- arguments: since and until are for time specifications
- lang: for languge specification
- geocode: for location specification

- we are now scraping 1k tweets for Udemy, and we als specify our certificate

```
udemytweets = searchTwitter("#Udemy", n=1000, cainfo="cacert.pem")
```

- as you can see, scraping that data is quite time consuming - your machine limits the efficiency and speed of your mining
- if you are plan to scrape a lot in the future 64bit systems and high RAM is desirable

```
class(udemytweets)

length(udemytweets)

head(udemytweets)


library("tm")


udemylist <- sapply(udemytweets, function(x) x$getText()) # initiating a function
```

- in depth info about the apply family and functions in the course "R Level 1"

```
udemycorpus <- Corpus(VectorSource(udemylist)) # use the corpus function
```

- a corpus is the text body consisting of all the text including the meta info

udemycorpus <- tm_map(udemycorpus, tolower) # **putting text to lower case**

udemycorpus <- tm_map(udemycorpus, removePunctuation) # **remove punct.**

udemycorpus <- tm_map(udemycorpus,

function(x)removeWords(x,stopwords())) # **remove stopwords (meaningless words)**

- there is a link to a stop word list in the link lecture

Let's see which other transformations tm offers

?getTransformations

- to trasform to plain text which wordcloud can use

udemycorpus <- tm_map(udemycorpus, PlainTextDocument)

library("wordcloud")

? wordcloud

```
wordcloud(udemycorpus, min.freq=4, scale=c(5,1),
          random.color=F, max.word=45, random.order=F)
```

- changing to a tdm

```
udemytdm <- TermDocumentMatrix(udemycorpus)
```

A DocumentTermMatrix is a very useful tool when it comes to text mining. It structures the text in a matrix where each term is organized in a column. Each row is a document and the number represents the counts of that term.

```
udemytdm
```

- frequent terms

```
findFreqTerms(udemytdm, lowfreq=11)
```

```
?findFreqTerms
```

- associations

```
findAssocs(udemytdm, 'android', 0.60)
```

Let's get a dendrogram to see related terms

- remove sparse (infrequently used) terms from the term-document matrix

```
udemy2tdm <-removeSparseTerms(udemytdm, sparse=0.9)
```

Let's scale the data

```
udemy2tdmscale <- scale(udemy2tdm)
```

- distance matrix

```
udemydist <- dist(udemy2tdmscale, method = "euclidean")
```

- hierarchical clustering

```
udemyfit <- hclust(udemydist)
```

- Visualize the result

```
plot(udemyfit)
```

- to calculate a certain number of groups

```
cutree(udemyfit, k=6)
```

- we can even color the 6 groups and plot them

```
rect.hclust(udemyfit, k=6, border="red")
```

SENTIMENT ANALYSIS

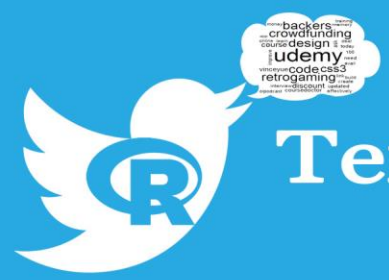
Sentiment analysis is used to see if a text is neutral, positive or negative emotion analysis is used to see which emotion a text has (happy, fear, anger) both are using similar codes but the comparison lexicon is different.

Example: What is the sentiment towards my company?

Twitter data is useful for that type of analysis because:

- high volumes (500 mill/day)
- short messages like sms - 140 words
- special strings (hashtags)
- but creative word usage makes it hard for analysis, spelling mistakes

There is TONS of sentiment in it!



Example of text sentiments:

#1. *Udemy provides great opportunity for life long learning* - both words would hint for pos

#2. *Udemy is too expensive and slow* - both words would hint towards neg sentiment

#3. *Udemy is a learning platform* - neutral sentiment

#4. Problem: *what? - Udemy is a fast platform?* - sarcasm is hard to analyse

Sentiment Lexicon: a list of words which you are using to compare your scraped txt with

Hu Liu Lexicon got the standard of sentiment analysis lately list of pos and negative words - manually created - approx. 6800

- download the txt files to your wd

- import positive and negative words

```
pos = readLines("positive_words.txt")

neg = readLines("negative_words.txt")
```

Let's run a test to see how this works!

```
mytest= c("great you re here", "awesome experience",
          "You had a bad night", "She loves ugly candy")
```

- the score.sentiment function is self written

```
testsentiment = score.sentiment(mytest, pos, neg)
```

```
class (testsentiment)
```

```
testsentiment$score
```

- output corresponds to the 4 test sentences - sentences can be manipulated

Let's do the whole process: writing the function and scraping - approach after J. Breen

```
library("stringr")
```

```
library("plyr")
```

- function score.sentiment - this is how the whole function is written

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

{
```

Parameters

- sentences: vector of text to score
- pos.words: vector of words of postive sentiment
- neg.words: vector of words of negative sentiment
- .progress: passed to laply() to control of progress bar

- create simple array of scores with laply

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

- remove punctuation - using global substitute

```
               sentence = gsub("[[:punct:]]", "", sentence)
```

- remove control characters

```
               sentence = gsub("[[:cntrl:]]", "", sentence)
```

- remove digits

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

- define error handling function when trying tolower

```
tryTolower = function(x)
{
```

- create missing value

```
  y = NA
```

- tryCatch error

```
  try_error = tryCatch(tolower(x), error=function(e) e)
```

- if not an error

```
  if (!inherits(try_error, "error"))
    y = tolower(x)
```

- result

```
  return(y)
}
```

- use tryTolower with sapply

```
sentence = sapply(sentence, tryTolower)
```

- split sentence into words with str_split (stringr package)

```
word.list = str_split(sentence, "\\s+")
words = unlist(word.list)
```

- compare words to the dictionaries of positive & negative terms

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

- get 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)
```

- final score

```
score = sum(pos.matches) - sum(neg.matches)
return(score)
}, pos.words, neg.words, .progress=.progress )
```

- data frame with scores for each sentence

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

- tweets for country

```
usatweets = searchTwitter("usa", n=900, lang="en", cainfo="cacert.pem")

indiatweets = searchTwitter("india", n=900, lang="en",
cainfo="cacert.pem")

russiatweets = searchTwitter("russia", n=900, lang="en",
cainfo="cacert.pem")

chinatweets = searchTwitter("china", n=900, lang="en",
cainfo="cacert.pem")
```

- get text

```
usa_txt = sapply(usatweets, function(x) x$getText())

india_txt = sapply(indiatweets, function(x) x$getText())

russia_txt = sapply(russiatweets, function(x) x$getText())

china_txt = sapply(chinatweets, function(x) x$getText())
```

- how many tweets of each country

```
nd = c(length(usa_txt), length(india_txt), length(russia_txt),
length(china_txt))
```

- join texts

```
country = c(usa_txt, india_txt, russia_txt, china_txt)
```

- apply function score.sentiment

```
scores = score.sentiment(country, pos, neg, .progress='text')
```

- add variables to data frame

```
scores$country = factor(rep(c("usa", "india", "russia", "china"), nd))
scores$very.pos = as.numeric(scores$score >= 2)
scores$very.neg = as.numeric(scores$score <= -2)
```

- how many very positives and very negatives

```
numpos = sum(scores$very.pos)
numneg = sum(scores$very.neg)
```

- global score

```
global_score = round( 100 * numpos / (numpos + numneg) )
```

```
head(scores)
```

```
boxplot(score~country, data=scores)
```

```
library("lattice")


histogram(data=scores, ~score|country, main="Sentiment Analysis of 4
Countries", xlab="", sub="Sentiment Score")
```