Twitter Mining

Searching for tweets
&
Analysing the 140 character

INTRODUCTION

What is twitter?

- An online social networking service that enables users to send and read short 140-character messages called "tweets".
- Over 300 million monthly active users (as of 2015).
- Creating over 500 million tweets per day.
- Tweets also contain links to pictures, videos, or other websites as well as hashtags—words that begin with # and are turned into links to make it easier to find certain terms.

Searching for tweets

- Problem:

  You want to search Twitter for tweets using specific keywords and query constraints.

- Solution:

  Use the Search API to perform a custom query.

Searching for tweets

- Twitter's Search API returns results in batches, and we can configure the number of results per batch to a maximum value using the **count** keyword parameter.
- Generally the count is taken as 200. It is possible that more than 200 results (or the maximum value that you specify for count) may be available for any given query.
- In Twitter's API, we also have **cursor** to navigate to the next batch of results.

Searching for tweets

- Cursors are a new enhancement to Twitter's API and provide a more robust scheme than the pagination paradigm.
- The essence of the cursor paradigm is that it is able to better accommodate the dynamic and real-time nature of the Twitter platform.
- It could be the case that while you are navigating a batch of query results, relevant information becomes available that you would want to have included in your current results while you are navigating them, rather than needing to dispatch a new query.

```
def twitter_search(twitter_api, q, max_results=200, **kw):

    # See https://dev.twitter.com/docs/api/1.1/get/search/tweets and
    # https://dev.twitter.com/docs/using-search for details on advanced
    # search criteria that may be useful for keyword arguments

    # See https://dev.twitter.com/docs/api/1.1/get/search/tweets
    search_results = twitter_api.search.tweets(q=q, count=100, **kw)

    statuses = search_results['statuses']

    # Iterate through batches of results by following the cursor until we
    # reach the desired number of results, keeping in mind that OAuth users
    # can "only" make 180 search queries per 15-minute interval. See
    # https://dev.twitter.com/docs/rate-limiting/1.1/limits
    # for details. A reasonable number of results is ~1000, although
    # that number of results may not exist for all queries.

    # Enforce a reasonable limit
    max_results = min(1000, max_results)

    for _ in range(10): # 10*100 = 1000
        try:
            next_results = search_results['search_metadata']['next_results']
        except KeyError, e: # No more results when next_results doesn't exist
            break

        # Create a dictionary from next_results, which has the following form:
        # ?max_id=313519052523986943&q=NCAA&include_entities=1
        kwargs = dict([ kv.split('=')
                        for kv in next_results[1:].split("&") ])

        search_results = twitter_api.search.tweets(**kwargs)
        statuses += search_results['statuses']

        if len(statuses) > max_results:
            break

    return statuses

# Sample usage

twitter_api = oauth_login()

q = "CrossFit"
results = twitter_search(twitter_api, q, max_results=10)

# Show one sample search result by slicing the list...
print json.dumps(results[0], indent=1)
```

Analysing the 140 character

- Tweets Analysis
  - Extracting Tweets
  - Text Cleaning
- Frequent Words and Word Cloud
  - Word Associations
- Topic Modelling
- Sentiment Analysis

Techniques and Tools

- Techniques
  - Text mining
  - Topic modelling
  - Sentiment analysis
  - Social network analysis
- Tools
  - Twitter API
  - R and its packages:
    - *twitter*
    - *tm*
    - *topicmodels*
    - *sentiment140*
    - *igraph*

Process

- Extract tweets and followers from the Twitter website with R and the *twitteR* package
- With the *tm* package, clean text by removing punctuations, numbers, hyperlinks and stop words, followed by stemming and stem completion
- Build a term-document matrix
- Analyse topics with the *topicmodels* package
- Analyse sentiment with the *sentiment140* package
- Analyse following/followed and retweeting relationships with the *igraph* package

Retrieve Tweets

```
## Option 1: retrieve tweets from Twitter
library(twitteR)  library(ROAuth)
## Twitter authentication
setup_twitter_oauth(consumer_key, consumer_secret, access_token,
    access_secret)
## 3200 is the maximum to retrieve
tweets<-userTimeline("RDataMining",n=3200)
```

```
## Option 2: download @RDataMining tweets from RDataMining.com
url<-"http://www.rdatamining.com/data/RDataMining-Tweets-20160212.rds"
download.file(url,destfile="./data/RDataMining-Tweets-20160212.rds")
## load tweets into R
tweets<-readRDS("./data/RDataMining-Tweets-20160212.rds")
```

```
(n.tweet<-length(tweets))

## [1] 448

#convert tweets to a data frame
tweets.df<-twListToDF(tweets)
#tweet #190
tweets.df[190,c("id","created","screenName","replyToSN",
  "favoriteCount","retweetCount","longitude","latitude","text")]

##       id       created screenName re...   ## 190
362866933894352898 2013-08-01 09:26:33 RDataMining    ...

##      favoriteCount retweetCount longitude latitude
## 190             9  9        NA       NA
##                                                   ...
## 190 The RReference Card for Data Mining nowprovides lin...

#print tweet #190 and make text fit for slide width
writeLines(strwrap(tweets.df$text[190],60))

## The R Reference Card for Data Mining now provides links to  ##
packages on CRAN. Packages for MapReduce and Hadoop added.  ##
http://t.co/RrFypol8kw
```

Text Cleaning

```
library(tm)
# build a corpus, and specify the source to be character vectors
myCorpus<-Corpus(VectorSource(tweets.df$text))
# convert to lower case
myCorpus<-tm_map(myCorpus,content_transformer(tolower))
# remove URLs
removeURL<-function(x)gsub("http[^[:space:]]*","", x)  myCorpus<-
tm_map(myCorpus,content_transformer(removeURL)) # remove anything
other than English letters or space
removeNumPunct<-function(x)gsub("[^[:alpha:][:space:]]*","", x)
myCorpus<-tm_map(myCorpus,content_transformer(removeNumPunct)) #
remove stopwords
myStopwords<-c(setdiff(stopwords('english'),c("r","big")),
                    "use","see","used","via","amp")
myCorpus<-tm_map(myCorpus, removeWords, myStopwords)
# remove extra whitespace
myCorpus<-tm_map(myCorpus, stripWhitespace)

# keep a copy for stem completion later
myCorpusCopy<-myCorpus
```

Stemming and Stem Completion 1

```
myCorpus<-tm_map(myCorpus, stemDocument)          # stem words
writeLines(strwrap(myCorpus[[190]]$content,60))

## r refer card data mine nowprovid link packag cran packag ##
mapreduc hadoop ad

stemCompletion2<-function(x, dictionary)          { x<-
  unlist(strsplit(as.character(x)," "))  x<-x[x!=""]
x<-stemCompletion(x,dictionary=dictionary)  x<-
  paste(x,sep="",collapse=" ")
  PlainTextDocument(stripWhitespace(x))
}
myCorpus<-lapply(myCorpus, stemCompletion2,dictionary=myCorpusCopy)
myCorpus<-Corpus(VectorSource(myCorpus))
writeLines(strwrap(myCorpus[[190]]$content,60))

## r reference card data miner nowprovided link package cran ##
package mapreduce hadoop add
```

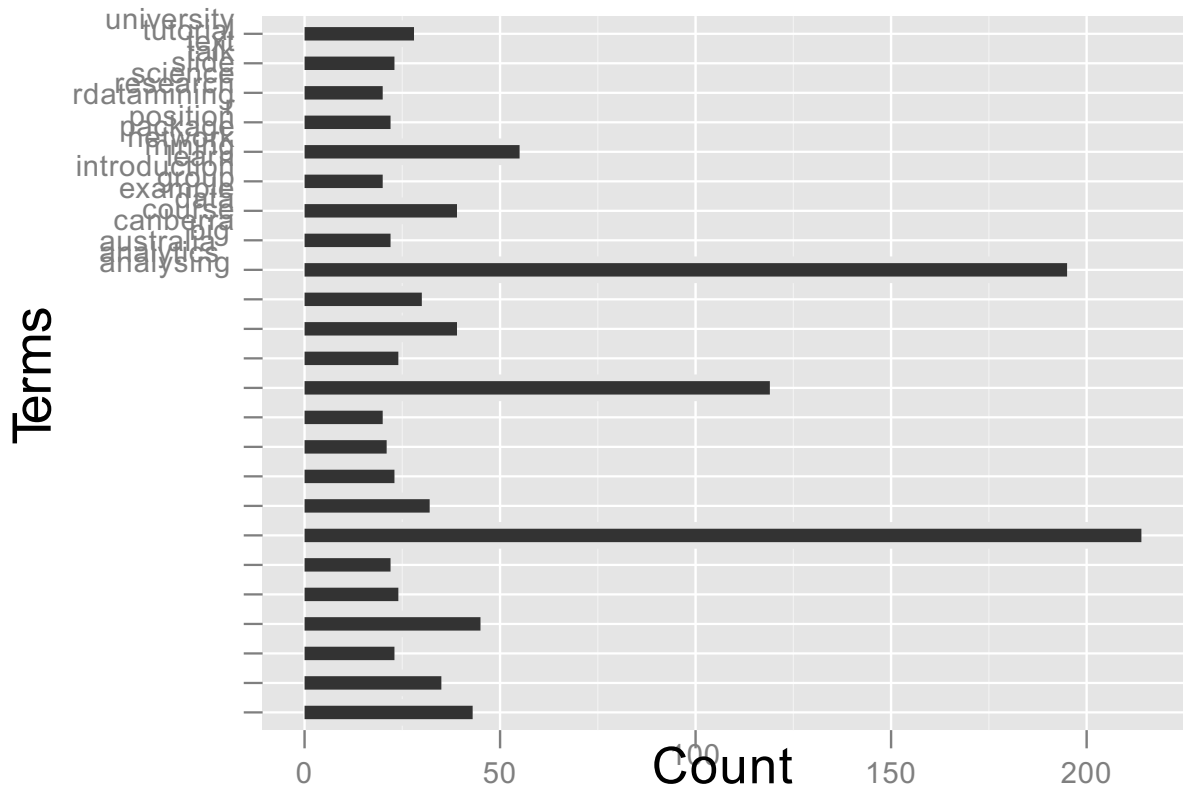
Build Term Document Matrix

```
tdm<-TermDocumentMatrix(myCorpus,
                        control=list(wordLengths=c(1,Inf)))
tdm

## <<TermDocumentMatrix (terms: 1073, documents: 448)>>  ## Non-/sparse
entries: 3594/477110
## Sparsity           : 9## Maximal term length: 23
## Weighting          : term frequency (tf)

idx<-which(dimnames(tdm)$Terms inc("r","data","mining"))
as.matrix(tdm[idx,21:30])

##           Docs
## Terms      21 22 23 24 25 26 27 28 29 30
##    data     0  1  0  0  1  0  0  0  0  1
##    mining   0  0  0  0  1  0  0  0  0  1
##    r        1  1  1  1  0  1  0  1  1  1
```

Top Frequent Terms

```
# inspect frequent words
(freq.terms<-findFreqTerms(tdm,lowfreq=20))

##  [1] "analysing"     "analytics"     "australia"     "big"
##  [5] "canberra"      "course"        "data"          "example"
##  [9] "group"         "introduction"  "learn"         "mining"
## [13] "network"       "package"       "position"      "r"  "slide"
## [17] "rdatamining"   "research"      "science"       "university"
## [21] "talk"          "text"          "tutorial"

term.freq<-rowSums(as.matrix(tdm))  term.freq<-
subset(term.freq, term.freq>=20)
df<-data.frame(term=names(term.freq),freq= term.freq)
```

```
library(ggplot2)
ggplot(df,aes(x=term,y=freq))+geom_bar(stat="identity")+
  xlab("Terms")+ylab("Count")+coord_flip()+
  theme(axis.text=element_text(size=7))
```



Wordcloud

```
m<-as.matrix(tdm)
#calculate the frequency of words and sort it by frequency
word.freq<-sort(rowSums(m),decreasing= T)
# colors
pal<-brewer.pal(9,"BuGn")[-(1:4)]


#plot word cloud
library(wordcloud)
wordcloud(words=names(word.freq),freq= word.freq,min.freq=3,
      random.order= F,colors= pal)
```

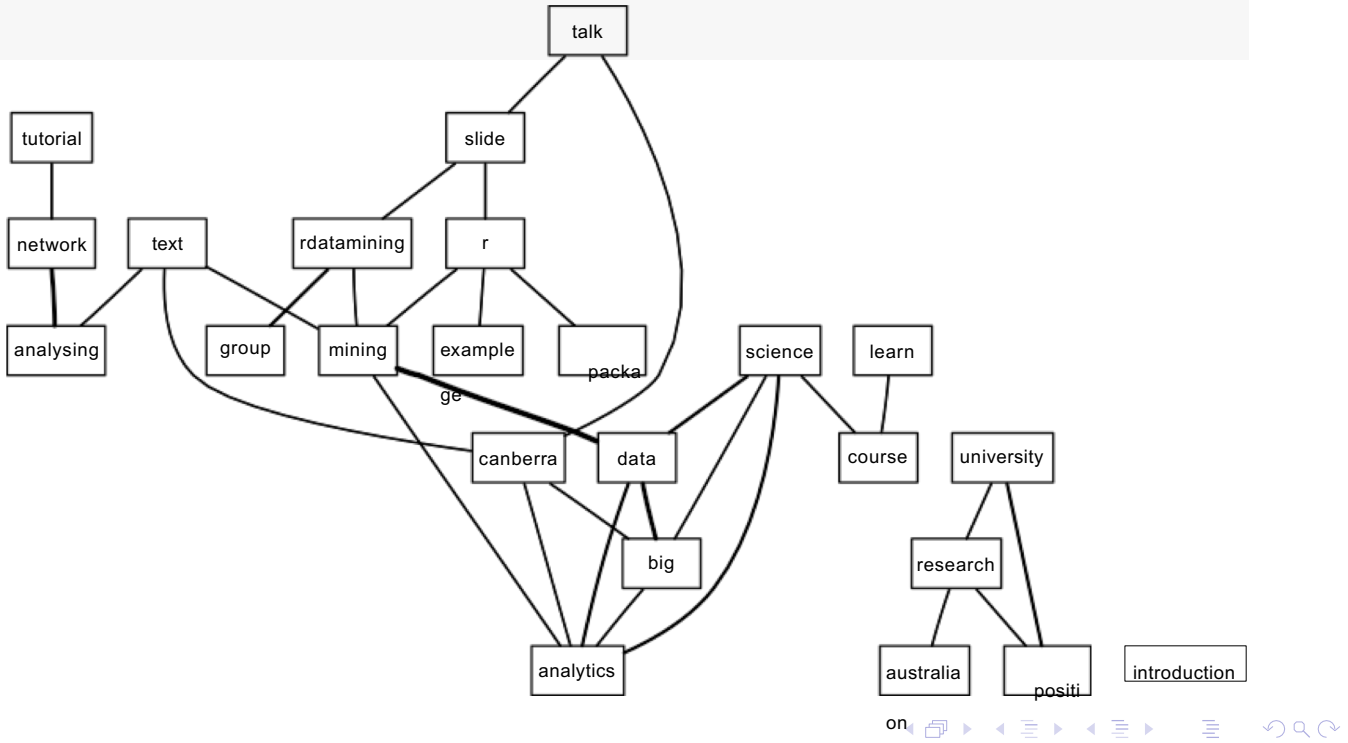
data
mining
r
slide
analysing
research
package
big
analytics
example
australia
introduction
computational
canberra
tutorial
position
network
university
social
rdatamining
scientist
ausdm
useful
online
statistical
course
available
present
series
free
modeling
text
lecture
book
workshop
conference
submission
associate
time
code
cluster
program
mapreduce
software
knowledge
database
technological
experience
access
hadoop
language
visualisation
machine
group
talk
pdf
open
rstudio
linkedin
dataset
sentiment
dynamic
tuesday
contain
looking
guidance
distributed
francisco
process
detailed
document
kdnuggets
melbourne
august
coursera
graphical
project
decision
webinar
format
advanced
business
list
function
cran
area
intern
tool
retrieval
version
nov
natural
performance
website
reference
thursday
detection
postdoctoral
source
handling
regression
week
add
skills
apache
seminar
postdoc
improve
paper
fast
make
march
system
sydney
prof
graph
together
china
industrial
case
run
san
web
algorithm
analyst
classification
member
call
find
dmapps
deadline
may
sas
healthcare
step
high
official
short
simple
give
random
extended
support
management
initial
wwwrdataminingcom
build
today
summit
southern
sunday
informal
cloud
forecasting
australian
australasian
topic
aug
datacamp
world
media
tweet
check
job
vacancies
share
follow
download
task
track
massive
will
stanford
participation
comment
chapter
various
technique
now
easier
provided
lab
youtube
plot
california
engine
senior
fellow
sept
visit
load
littlesnowfall
public
center
excel
create
twitter
video
large
risk
close
predicting
spatial

Associations

```
# which words are associated with 'r'?
findAssocs(tdm,"r",0.2)

##        r
## code 0.27
## example   0.21
## series         0.21
## markdown 0.20
## user 0.20

# which words are associated with 'data'?
findAssocs(tdm,"data",0.2)


##              data
## mining       0.48
## big          0.44
## analytics 0.31
## science      0.29
## poll         0.24
```

Network of Terms

```
library(graph)  library(Rgraphviz)
plot(tdm,term= freq.terms,corThreshold=0.1,weighting= T)
```

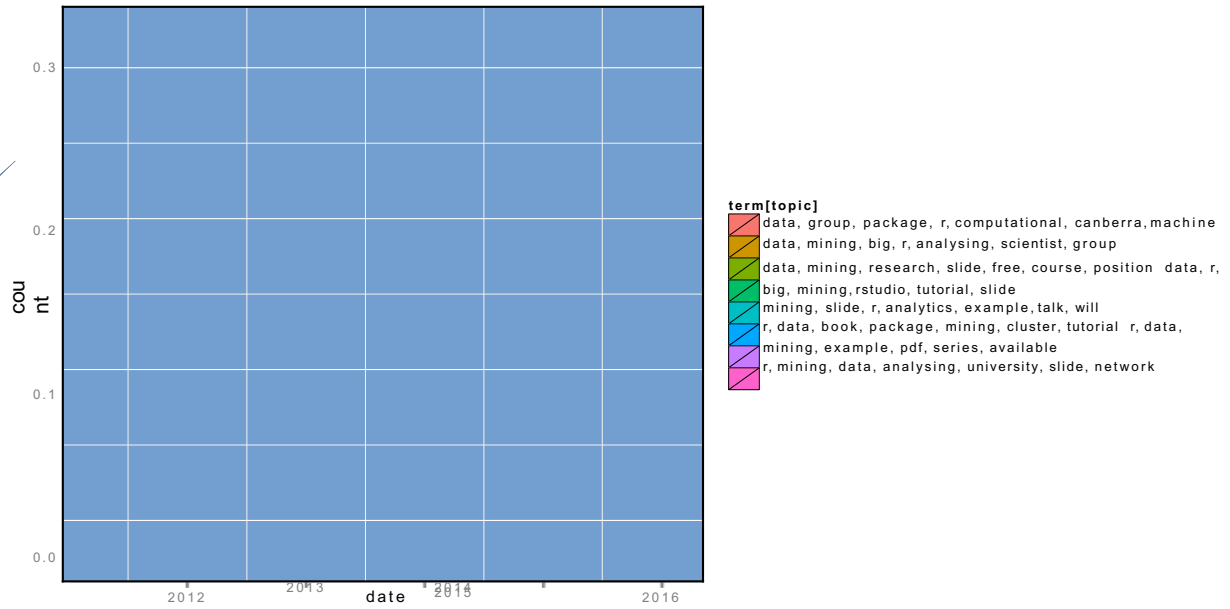


Topic Modelling

```
dtm<-as.DocumentTermMatrix(tdm)
library(topicmodels)
lda<-LDA(dtm,k=8)          #find 8 topics
term<-terms(lda,7)         #first 7 terms of every topic
(term<-apply(term,MARGIN=2, paste,collapse=", "))


##
##
##
##                Topic 1 "data, mining, big, r, analysing, scientist, group"
##           Topic 2 "r, mining, data, analysing, university, slide, network"
##                Topic 3 "r, data, book, package, mining, cluster, tutorial"
##                Topic 4 "data, r, big, mining, rstudio, tutorial, slide"
##
##            "data, mining, research, slide, free, course, position"
##
##
##
##
## "data, group, package, r, computational, canberra, machine"
##                                                          Topic 7
##             "mining, slide, r, analytics, example, talk, will"
##                                                          Topic 8
##             "r, data, mining, example, pdf, series, available"
```

Topic Modelling

```
topics<-topics(lda)      # 1st topic identified for every document (tweet)
topics<-data.frame(date=as.IDate(tweets.df$created),topic=topics)
qplot(date, ..count..,data=topics,geom="density",
      fill=term[topic],position="stack")
```



Sentiment Analysis

```
#install package sentiment140
require(devtools) install_github("sentiment140","okugami79")
```

```
#sentiment analysis
library(sentiment)
sentiments<-sentiment(tweets.df$text)
table(sentiments$polarity)

##
## neutral positive  ##  428      20

#sentiment plot
sentiments$score<-0
sentiments$score[sentiments$polarity=="positive"]<-1
sentiments$score[sentiments$polarity=="negative"]<--1
sentiments$date<-as.IDate(tweets.df$created)
result<-aggregate(score~date,data= sentiments, sum)
plot(result,type="l")
```


THANK YOU