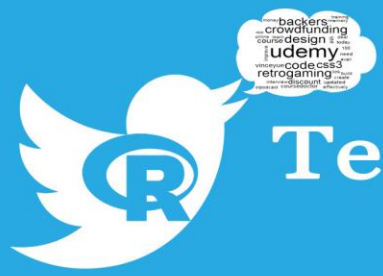


Exercise - Text Mining

Part 1

1. Get 1000 Tweets to a search term of your choice - I am using #bigpharma
2. Clean the Tweets - lower cases, remove numbers, punctuations, stopwords
3. Plot a wordcloud of the 45 most frequent terms, min frequency is 3
4. Rotate 50% of the words
5. Color your cloud (hint: Color Brewer)

Part 2

1. Use the same dataset - change it to term document matrix
2. Get a list of the most frequent terms
3. Get a dendrogram, and group it according to best group fitting

SOLUTION

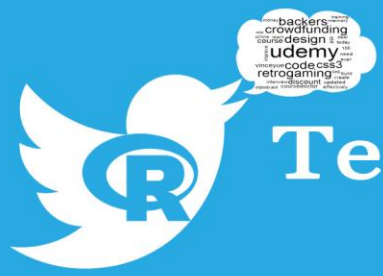
Part 1

```
tweets = searchTwitter("#bigpharma", n=1000, cainfo="cacert.pem")
head(tweets)
library("tm")
mylist <- sapply(tweets, function(x) x$getText())
mycorpus <- Corpus(VectorSource(mylist))
mycorpus <- tm_map(mycorpus, tolower)
mycorpus <- tm_map(mycorpus, removeNumbers)
mycorpus <- tm_map(mycorpus, removePunctuation)
mycorpus <- tm_map(mycorpus,
                   function(x)removeWords(x,stopwords()))
mycorpus <- tm_map(mycorpus, PlainTextDocument)
library("wordcloud")
library("RColorBrewer")
?RColorBrewer
col <- brewer.pal(5,"Dark2") # 6 is the number of colors, rest is pal name
wordcloud(mycorpus, min.freq=3, rot.per=0.5, scale=c(4,1),
          random.color=T, max.word=45, random.order=F, colors=col)
```

- scale to adjust the size
- rot.per to adjust the number of rotated words
- random.color to connect frequency and color
- get colors from the ColorBrewer

Part 2

```
mytdm <- TermDocumentMatrix(mycorpus)

findFreqTerms(mytdm, lowfreq=55) # experiment with the lowfreq

tdm <-removeSparseTerms(mytdm, sparse=0.93) # experimet with sparse

tdmscale <- scale(tdm)

dist <- dist(tdmscale, method = "canberra")

fit <- hclust(dist)

plot(fit)

# we need to change the margins and delete some titles

par(mai=c(1,1.2,1,0.5))

plot(fit, xlab="", sub="", col.main="salmon")

cutree(fit, k=7)

rect.hclust(fit, k=7, border="salmon")
```



Exercise - Sentiment Analysis

Sentiment Analysis for Investment Decisions

1. Perform a comparative sentiment analysis on 4 pharma companies: Bayer, Pfizer, Roche, Novartis
2. Get as many Tweets as possible from those companies
3. Use an opinion lexicon and a suitable function to score the sentiment
4. Visualize your results and compare the results with the stock prices (e.g. yahoo finance)
5. Do you see any correlations between the sentiment and the stock price?

SOLUTION

- import positive and negative words

```
pos = readLines("positive_words.txt")
neg = readLines("negative_words.txt")
library("stringr")
library("plyr")
score.sentiment = function(sentences, pos.words, neg.words,
.progress='none')
{
  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 companies - may not get the full 900

```
bayertweets = searchTwitter("#bayer", n=900, lang="en",
cainfo="cacert.pem")

pfizertweets = searchTwitter("#pfizer", n=900, lang="en",
cainfo="cacert.pem")

rochetweets = searchTwitter("#roche", n=900, lang="en",
cainfo="cacert.pem")

novartistweets = searchTwitter("#novartis", n=900, lang="en",
cainfo="cacert.pem")
```

- get text

```
bayer_txt = sapply(bayertweets, function(x) x$getText())

pfizer_txt = sapply(pfizertweets, function(x) x$getText())

roche_txt = sapply(rochetweets, function(x) x$getText())

novartis_txt = sapply(novartistweets, function(x) x$getText())
```

- how many tweets

```
nd = c(length(bayer_txt), length(pfizer_txt), length(roche_txt),
length(novartis_txt))
```

- join texts

```
company = c(bayer_txt, pfizer_txt, roche_txt, novartis_txt)
```

- apply function score.sentiment

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

- add variables to data frame

```
scores$company = factor(rep(c("bayer", "pfizer", "roche", "novartis"), nd))
scores$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)
par(bty="l")
boxplot(score~company, data=scores, col=c("red", "grey"))
```

```
library("lattice")


histogram(data=scores, ~score|company, main="Sentiment Analysis of 4
Companies", col=c("red", "grey"),
        xlab="", sub="Sentiment Score")
```