TwitterMining

December 23, 2017

1 Twitter Text Mining - Required Libraries

In [2]:
```
library(twitteR)
library(ROAuth)
library(RCurl)
library(httr)

library(stringr)
library(plyr)
library(dplyr)
library(tm)

#library(ggmap)
#library(wordcloud)
```

2 Establishing A Connection - Direct Method

Enter your key and token from your twitter developer page

In [ ]:
```
key=" "
secret=" "

atoken =  " "
asecret = " "

setup_twitter_oauth(key, secret, atoken, asecret)
```

3 Sentiment Score Function - approach after J. Breen

In [ ]:
```
library("stringr")

library("plyr")

# Function is called sentimentfun
sentimentfun = function(tweettext, pos, neg, .progress='non')
{
```

```
# Parameters
# tweettext: vector of text to score
# pos: vector of words of postive sentiment
# neg: vector of words of negative sentiment
# .progress: passed to laply() 4 control of progress bar

# create simple array of scores with laply
scores = laply(tweettext,
      function(singletweet, pos, neg){
          # remove punctuation - using global substitute
          singletweet = gsub("[[:punct:]]", "", singletweet)
          # remove control characters
         singletweet = gsub("[[:cntrl:]]", "", singletweet)
         # remove digits
         singletweet = gsub("\\d+", "", singletweet)

         # 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
         singletweet = sapply(singletweet, tryTolower)

         # split sentence into words with str_split (stringr package)
         word.list = str_split(singletweet, "\\s+")
         words = unlist(word.list)

         # compare words to the dictionaries of positive & negative terms
         pos.matches = match(words, pos)
         neg.matches = match(words, neg)

         # 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, neg, .progress=.progress)
```

```
    # data frame with scores for each sentence
    sentiment.df = data.frame(text=tweettext, score=scores)
    return(sentiment.df)
  }
```

4 Using searchTwitter for our project

- Los Angeles, geocode="34.052,-118.244,200mi"
- New York, geocode="40.713,-74.006,200mi"
- Austin, geocode="30.267,-97.743,500mi"
- Seattle, geocode="47.606,-122.332,500mi"

4.0.1 Searching for 'apple+iphone' or 'samsung+galaxy'

4.0.2 Since is always 14 days prior to run-date, due to API restrictions

```
In [ ]: tweets =searchTwitter("samsung+galaxy", n=2000,
                              lang="en",
                              geocode="47.606,-122.332,500mi",
                              since = "2017-12-04")
```

5 Extracting the text

```
In [ ]: tweettext = sapply(tweets, function(x) x$getText())
```

5.1 First cleaning stage

```
In [ ]: tweettext=lapply(tweettext, function(x) iconv(x, "latin1",
                                                      "ASCII", sub=""))
        tweettext=lapply(tweettext, function(x) gsub("htt.*",' ',x))
        tweettext=lapply(tweettext, function(x) gsub("#",'',x))
        tweettext=unlist(tweettext)
```

6 Getting the opinion lexicons from working directory

```
In [ ]: pos = readLines("positive_words.txt")
        neg = readLines("negative_words.txt")

        neg2 = c(neg, "bearish", "fraud"); tail(neg2)
```

6.1 Apply function score.sentiment

```
In [ ]: scores = sentimentfun(tweettext, pos, neg, .progress='text')
```

7 Extracting further elements (besides text) for the export csv

```
In [ ]: tweetdate=lapply(tweets, function(x) x$getCreated())
        tweetdate=sapply(
            tweetdate,function(x) strftime(
                x, format="%Y-%m-%d %H:%M:%S",tz = "UTC"))

        isretweet=sapply(tweets, function(x) x$getIsRetweet())

        retweetcount=sapply(tweets, function(x) x$getRetweetCount())

        favoritecount=sapply(tweets, function(x) x$getFavoriteCount())
```

8 Creating the Data Frame

```
In [ ]: data=as.data.frame(cbind(ttext=tweettext,
                                date=tweetdate,
                                isretweet=isretweet,
                                retweetcount=retweetcount,
                                favoritecount=favoritecount,
                                score = scores$score,
                                product = "Samsung Galaxy",
                                city = "Seattle", country = "USA"))
```

8.1 Remove duplicates

```
In [ ]: data2 = duplicated(data[,1])
        data$duplicate = data2
```

8.2 Create file to wd

```
In [ ]: write.csv(data, file= "samsung_seattle.csv")
```