BOSTON UNIVERSITY

CS 688 Web Analytics and Mining

Tweeter Stock Market Sentiment Analysis

Archana Balachandran

Spring 2017

Table of Contents

Introduction 3

Tools and Packages 4

Text Mining Methodology 5

Sentiment Analysis Methodology 6

Understanding Stock Data 7

Twitter Text Mining 8

- Creating Twitter Application 8
- Setting up RStudio and Twitter API 8
- Extracting Tweets 9
- Creating the Data Corpora 10
- Transforming the Data Corpus 11
- Creating the Term-Document Matrix 13
- Finding Frequent Terms 15
- Generating WordCloud 17

Sentiment Analysis 18

- Loading the Lexicons into the workspace 18
- Generating Sentiment Scoring Function 19
- Computing Sentiment Score for Stock Tweets 20

Analysis and Conclusion 22

- Analysis of Individual Loser stock tweets 23

Bonus Section: visualization 27

References 29

Project Code 30

Introduction

Twitter is a social networking and microblogging site that allows users to broadcast short burst messages to a world called 'twitterverse', by answering a simple question: "What's happening?".

Text mining involves extraction of meaningful information from a large collection of unstructured text, extracted from various text mines. Text mining in the context of this project involves mining tweets from twitter on a particular stock and performing text processing on the extracted data.

Sentiment Analysis involves extracting opinions or sentiment from texts, which is then converted into a numeric index to determine the overall sentiment related to a particular subject.

This project appreciates the fact that, Twitter, apart from being an excellent medium for real-time communication, is also a massive repository for conducting text analysis to extract public sentiment on a particular stock.

This project aims to derive a correlation between public sentiment and market sentiment from twitter data. That is, to observe whether computation of public behavior or collective mood, is related to stock market behavior. The project also points out the public response in the form of tweets to management decisions by companies that have affected their stock prices.

The project scope does not include analysis of non-text information, such as emojis. It also does not involve analysis of links present within tweets. The text mined from Twitter indicates the public mood only based on the particular day of text extraction(April 21, 2017).

Tools And Packages

RStudio GUI, along with the following packages were used for the completion of this project:

twitteR - this packages allows the R console to interface with the Twitter web API. Functions such as setup_twitter_oauth( ) and searchTwitter( ) are primarily used in this project.

tm package offers a powerful framework for text mining applications in R. Functions for creating the data corpus and text processing such as createCorpus( ), content_transformer( ), inspect( ) and tm_map( ) are widely used, while other functions such as TermDocumentMatrix( ), findFreqTerms( ), stripWhitespace( ), stemDocument( ) are also used in this project.

wordcloud allows R users to plot a cloud of words shared across documents based on a word occurrence frequency set by the user.

googleVis package is installed for generating bar charts using Google Charts.

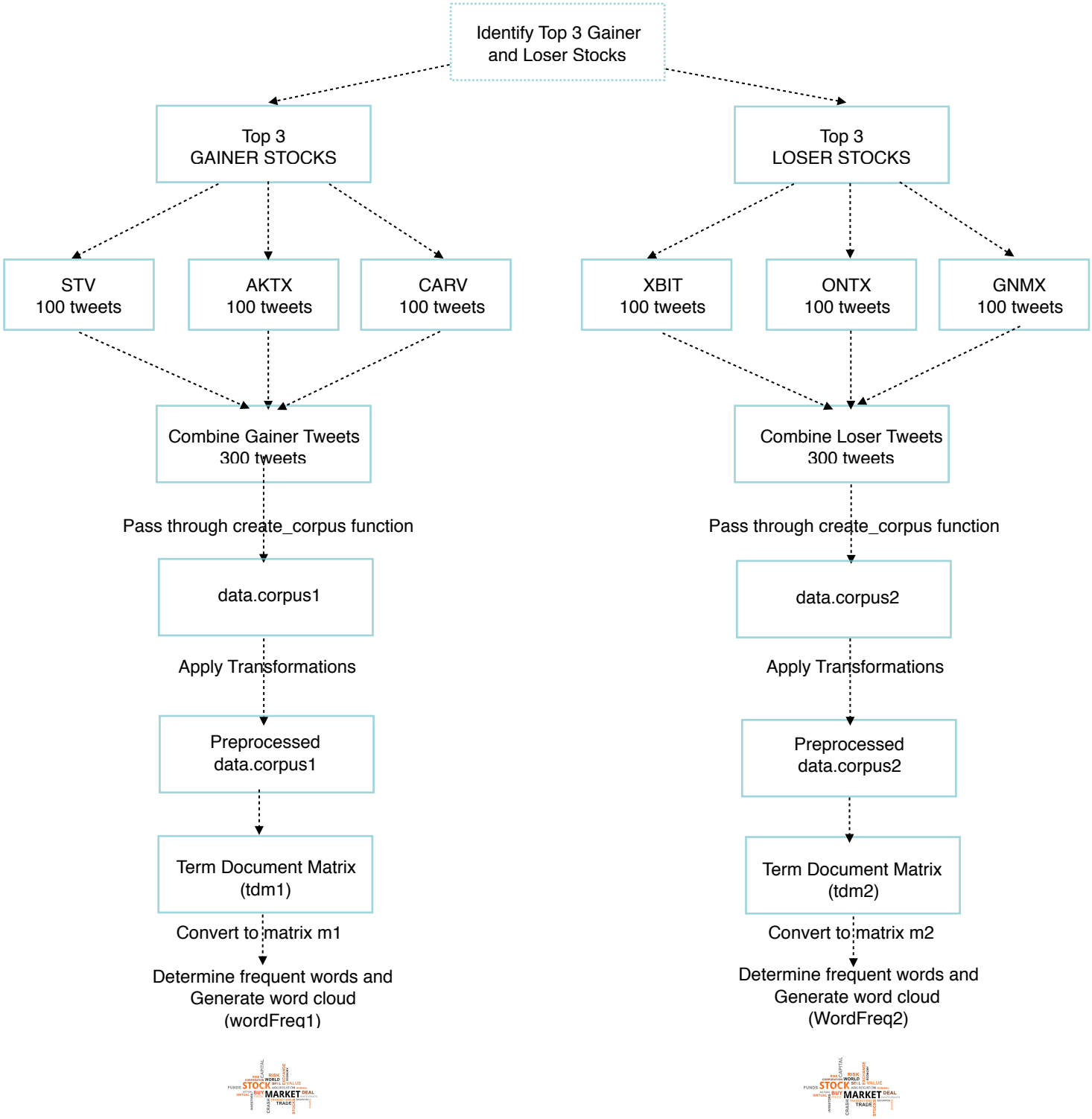
quantmod package is used for retrieving current stocks information from Yahoo Finance using the getQuote( ) function.

Word lists for Lexicons[1]:

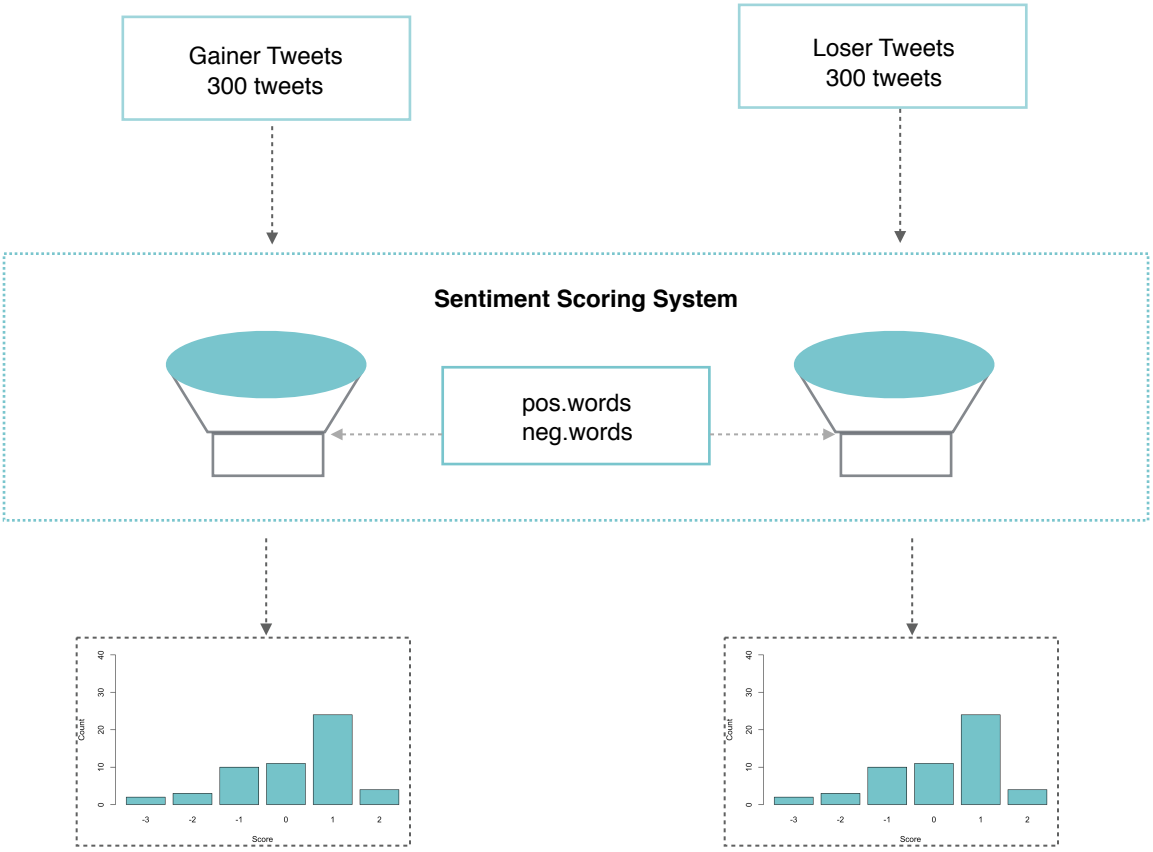
- positive-words.txt contains a list of all the positive words used in the sentiment analysis.
- negative-words.txt contains a list of all the negative words used in the sentiment analysis.

[1] Liu, B., Hu, M., & Cheng, J. (2005). Opinion observer. Proceedings of the 14th international conference on World Wide Web - WWW '05. doi:10.1145/1060745.1060797

Text Mining Methodology



Sentiment Analysis Methodology



Understanding Stock Data

The top 3 gainer and loser stocks on April 21st are selected for this project. This data was extracted from https://finance.yahoo.com.

Stocks: Gainers ›

Symbol	Last Price	Change	% Change
STV China Digital TV Holding Co., Ltd.	1.71	0.44	34.65%
AKTX Akari Therapeutics, Plc	20.50	4.13	25.23%
CARV Carver Bancorp, Inc.	4.90	0.91	22.81%
GLBS Globus Maritime Limited	2.95	0.53	21.90%
CAI CAI International, Inc.	18.57	3.07	19.81%

Stocks: Losers ›

Symbol	Last Price	Change	% Change
XBIT XBIOTECH INC.	10.23	-6.79	-39.89%
CRHM CRH Medical Corporation	7.35	-1.70	-18.78%
GNMX Aevi Genomic Medicine, Inc.	1.54	-0.32	-17.20%
ONTX Onconova Therapeutics, Inc.	2.11	-0.38	-15.26%
CLW Clearwater Paper Corporation	48.50	-8.60	-15.06%

Based on the % change values, it can be observed that STV, AKTX and CARV are the top three gainers(top positive % change), while XBIT, ONTX and GNMX are the top losers(top negative % change) on that day. (Note: $CRHM returned only 53 tweets on the day of analysis, hence ONTX was selected for creating a better dataset for analysis).

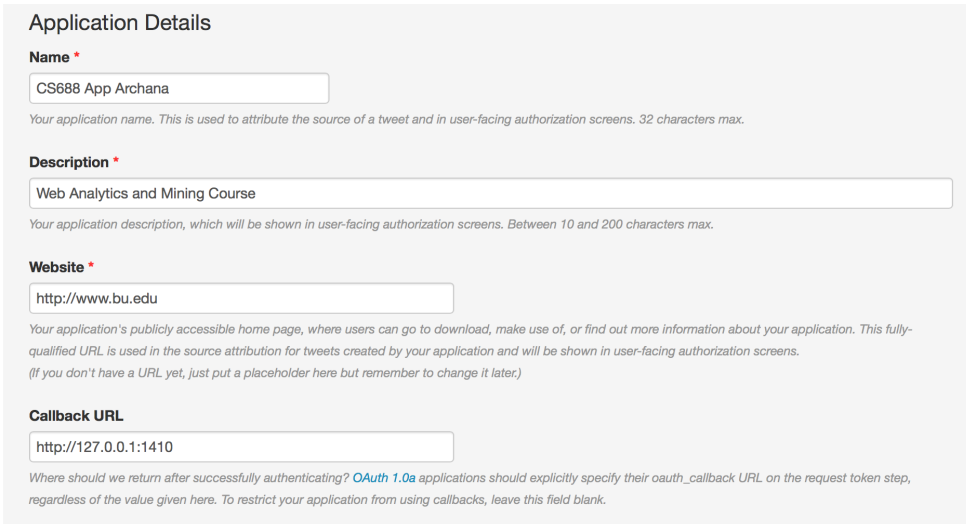
The top six gainer and loser stocks mentioned above were selected for analysis, and tweets based on their cashtags('$') were gathered. Cashtag is a feature by Twitter that allows users to track stock-related tweets. Text processing was then performed on the data in order to determine frequent terms and also to generate the word cloud for each set of gainers and losers dataset.

The next part of the project will perform sentiment analysis on twitter data and compute the sentiment score for the gainer and loser dataset. A histogram of the scores for each set will help us understand if a correlation exists between public sentiment and market sentiment.

Twitter Text Mining

Creating Twitter Application

A Twitter application is required to perform analysis on data obtained from Twitter,. This application enables users to connect the R console to Twitter using Twitter API. A new App is created and registered using apps.twitter.com, providing the details as shown below:



Setting up RStudio and Twitter API

The first step performed in the R console is installing and loading required packages into the workspace.

```
install.packages("twitteR")
install.packages("tm")
library(twitteR)
library(tm)
```

Handshaking between R console and Twitter is done using Consumer Key and Consumer Secret values obtained after creating the Twitter application, which is entered in the R console as shown below:

```
t.api.key<-"ieaUlcnfMv8vSXYNkX5wFBZTK"
t.api.secret<-"bEbt6ekRgOvWiv8LwsV8L4uzIzhVmU5eoMCl8iQ2ZLUmb29dSx"
setup_twitter_oauth(t.api.key, t.api.secret, access_token=NULL, access_secret=NULL)
save(list=(c("t.api.key","t.api.secret")), file="twitter_credentials.RData")
```

Extracting Tweets

For each of the six selected stocks (3 gainers, 3 losers), 100 most recent tweets related to their respective cashtags are extracted using searchTwitter function. The searchTwitter function returns a list containing 100 tweets, on each of the 6 stocks, which is then saved into a variable (for e.g., $STV tweets are saved into a tweets.stv variable). Once three lists for each of the three gainer stocks are extracted and saved, they are combined into one list named gainer.tweets. The same process is repeated for the loser stocks.

```
# GAINER STOCKS
tweets.stv <- searchTwitter('$STV',n=100)
tweets.aktx <- searchTwitter('$AKTX',n=100)
tweets.carv <- searchTwitter('$CARV',n=100)
gainer.tweets<-c(tweets.stv,tweets.aktx,tweets.carv)

# LOSER STOCKS
tweets.xbit <- searchTwitter('$XBIT',n=100)
tweets.ontx <- searchTwitter('$ONTX',n=100)
tweets.gnmx <- searchTwitter('$GNMX',n=100)
loser.tweets<-c(tweets.xbit,tweets.ontx,tweets.gnmx)
```

This list contains non-text data such as emojis, which are removed by passing the list through a function that converts the data into ASCII characters, and returns a new text list gainer.tweets.text and loser.tweets.text

For performing a program sanity check, once the final two lists of gainer and loser tweets are created, it is verified that the list type is maintained throughout the extraction and transformation process, along with the correct number of rows.

```
> head(gainer.tweets.text,n=2)
[[1]]
[1] "Why STV Was UP Friday https://

[[2]]
[1] "Top penny stocks $STV, $CARV,
o/jrg1rpGBTg"

> length(gainer.tweets.text)
[1] 300
> typeof(gainer.tweets.text)
[1] "list"
```

```
> head(loser.tweets.text,n=2)
[[1]]
[1] "Why XBIT Was DOWN Friday htt

[[2]]
[1] "Sonar is a 24/7 social activ
intraday https://t.co/wlXajTZaE3

> length(loser.tweets.text)
[1] 300
> typeof(loser.tweets.text)
[1] "list"
```

Values	
gainer.tweets	List of 300
gainer.tweets.text	List of 300
loser.tweets	List of 300
loser.tweets.text	List of 300
t.api.key	"ieaUlcnfMv8v
t.api.secret	"bEbt6ekRgOvW
tweets.aktx	List of 100
tweets.carv	List of 100
tweets.gnmx	List of 100
tweets.ontx	List of 100
tweets.stv	List of 100
tweets.xbit	List of 100

Creating the Data Corpora

This section explains how two separate data corpora were created for each of the two sets created in the previous step. A function called createCorpus( ) that takes the set of tweets obtained from the previous section (gainer.tweets.text and loser.tweets.text) and returns a corpus is set up. The corpus obtained from gainer.tweets.text is labelled data.corpus1, and that from loser.tweets.text is labelled data.corpus2. The corpora are then inspected to confirm the type.

```
# function that takes a tweet and returns a corpus
createCorpus<- function(tweetset){
data.source <-VectorSource(tweetset)
data.corpus<-VCorpus(data.source)
return(data.corpus)
}

# Creating corpora using the createCorpus function
data.corpus1<-createCorpus(gainer.tweets.text)
data.corpus2<-createCorpus(loser.tweets.text)
typeof(data.corpus1)
# Inspecting the corpora
inspect(data.corpus1[1:2])
inspect(data.corpus2[1:2])
```

data.corpus1	Large VCorpus (300 elements, 1.1 Mb)
data.corpus2	Large VCorpus (300 elements, 1.1 Mb)

```
[1] "list"
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 2

[[1]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 87

[[2]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 100

<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 2

[[1]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 132

[[2]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 54
```

Transforming the Data Corpus

Both corpora created in the previous step are then transformed by applying pre-processing steps. This is implemented using a function called transformCorpus( ) which takes a corpus, and returns a pre-processed corpus.

The various pre-processing performed on each corpora are:

- Conversion to Lowercase
- Removal of URLs
- Removal of English Stop words
- Removal of Punctuation
- Removal of words beginning with a number
- Document Stemming
- Stripping Whitespace

```
# Defining the transformCorpus() function
transformCorpus<- function(data.corpus){
content_transformer(function(x, from, to) gsub(from, to, x))

# convert to lowercase
data.corpus <-
  tm_map(data.corpus,
         content_transformer(tolower))

# Defining URL elements for removing URL from text
removeURL <- function(x) {
  gsub("(http[^ ]*)", "", x)
}

# remove URL
data.corpus <-
  tm_map(data.corpus,
         content_transformer(removeURL))

# Defining stop words in this context
english.stopwords <- stopwords("en")
head(english.stopwords)

# remove stopwords
data.corpus <-
  tm_map(data.corpus,
         content_transformer(removeWords),
         english.stopwords)
```

```r
# Remove punctuation

data.corpus <-
  tm_map(data.corpus,
         content_transformer(removePunctuation))


# remove words that begin with a number
removeNumberWords <- function(x) {
  gsub("([[:digit:]]+)([[:alnum:]])*", "", x)
}

data.corpus <-
  tm_map(data.corpus,
         content_transformer(removeNumberWords))

# Perform stemming

data.corpus <-
  tm_map(data.corpus,
        content_transformer(stemDocument))


# Strip white space
data.corpus <-
  tm_map(data.corpus,
         content_transformer(stripWhitespace))

return(data.corpus)
}
```

Each of the corpora is passed to this function, and a transformed corpora is returned. The same is inspected to ensure the type and number of elements are preserved.

```
> inspect(data.corpus1[1:2])
<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (indexed): 0
Content:  documents: 2

[1] top penni stock stv carv cytr glbs trch voc yeco
[2] biggest gainer friday stv aktx carv cytr glbs cai trch sngx
> inspect(data.corpus2[1:2])
<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (indexed): 0
Content:  documents: 2

[1] think subscrib checkout introductori offer promo code newsubd asp nadl xbit cbli
[2] xbit friday xbit
```

Creating the Term-Document Matrix

The term document matrix is generated for each of the pre-processed corpora obtained from the previous step, and named tdm1 and tdm2 respectively, and saved as R objects. For the purpose of analysis in the next step (finding frequent terms), the term document matrix is also converted into a matrix.

- **Creation of tdm1**

```
> tdm1 <- TermDocumentMatrix(data.corpus1)
> inspect(tdm1[1:10, 50:60])
<<TermDocumentMatrix (terms: 10, documents: 11)>>
Non-/sparse entries: 36/74
Sparsity           : 67%
Maximal term length: 5
Weighting          : term frequency (tf)
Sample             :
       Docs
Terms   50 51 52 53 54 56 57 58 59 60
  carv   1  1  1  1  0  0  0  1  0  0
  cytr   0  0  0  0  0  0  0  0  1  1
  glbs   1  1  1  1  1  1  1  1  0  0
  penni  0  0  0  0  0  0  0  0  0  0
  stock  0  0  0  0  0  0  0  0  0  0
  stv    1  1  1  1  1  1  1  1  1  1
  top    1  0  0  0  0  0  0  1  1  1
  trch   0  0  0  0  0  0  0  1  0  0
  voc    1  1  1  1  0  0  0  1  0  0
  yeco   0  0  0  0  0  0  0  0  0  0
>
> # convert TDM to a matrix and view portion of the matrix
>
> m1 <- as.matrix(tdm1)
> m1[1:10, 51:60]
       Docs
Terms   51 52 53 54 55 56 57 58 59 60
  carv   1  1  1  0  0  0  0  1  0  0
  cytr   0  0  0  0  0  0  0  0  1  1
  glbs   1  1  1  1  0  1  1  1  0  0
  penni  0  0  0  0  0  0  0  0  0  0
  stock  0  0  0  0  0  0  0  0  0  0
  stv    1  1  1  1  1  1  1  1  1  1
  top    0  0  0  0  0  0  0  1  1  1
  trch   0  0  0  0  0  0  0  1  0  0
  voc    1  1  1  0  0  0  0  1  0  0
  yeco   0  0  0  0  0  0  0  0  0  0
```

- **Creation of tdm2**

```
> tdm2 <- TermDocumentMatrix(data.corpus2)
> inspect(tdm2[1:10, 50:60])
<<TermDocumentMatrix (terms: 10, documents: 11)>>
Non-/sparse entries: 7/103
Sparsity           : 94%
Maximal term length: 12
Weighting          : term frequency (tf)
Sample             :
               Docs
Terms           50 51 52 53 54 55 56 57 58 59
  asp            0  0  0  0  0  0  0  0  0  1
  cbli           0  0  0  0  0  0  0  0  0  1
  checkout       0  0  0  0  0  0  0  0  0  0
  code           0  0  0  0  0  0  0  0  0  1
  introductori   0  0  0  0  0  0  0  0  0  0
  nadl           0  0  0  0  0  0  0  0  0  1
  newsubd        0  0  0  0  0  0  0  0  0  1
  offer          0  0  0  0  0  0  0  0  0  0
  promo          0  0  0  0  0  0  0  0  0  1
  subscrib       0  0  0  0  0  0  0  0  0  1
> # convert TDM to a matrix and view portion of the matrix
> m2 <- as.matrix(tdm2)
> m2[1:10, 51:60]
               Docs
Terms           51 52 53 54 55 56 57 58 59 60
  asp            0  0  0  0  0  0  0  0  1  0
  cbli           0  0  0  0  0  0  0  0  1  0
  checkout       0  0  0  0  0  0  0  0  0  0
  code           0  0  0  0  0  0  0  0  1  0
  introductori   0  0  0  0  0  0  0  0  0  0
  nadl           0  0  0  0  0  0  0  0  1  0
  newsubd        0  0  0  0  0  0  0  0  1  0
  offer          0  0  0  0  0  0  0  0  0  0
  promo          0  0  0  0  0  0  0  0  1  0
  subscrib       0  0  0  0  0  0  0  0  1  0
```

```
# Saving tdm1 and tdm2 as R objects

save(list=("tdm1"),
     file = "GainerTDM.RData")

load(file="GainerTDM.RData")


save(list=("tdm2"),
     file = "LoserTDM.RData")

load(file="LoserTDM.RData")
```

Finding Frequent Terms

The top ten frequent terms are determined using the matrices(named m1 and m2) of term-document matrices(tdm1 and tdm2 respectively) obtained from the previous step. This is done by first applying the rowSums( ) function to each matrix(m1 and m2), and then sorting in descending order. This sorted list is further filtered by viewing the first 10 rows, and passed through cbind function (column bind) to display the results vertically for better visual interpretation.

To find all the frequent terms with at least 10 occurrences, the term document matrix is passed to the findFreqTerms function, also by setting the lowfreq argument as 10. The same process is applied to m2 and tdm2 to obtain the frequent terms.

```
> wordFreq1 <- rowSums(m1)
> wordFreq1 <- sort(wordFreq1, decreasing=TRUE)
> cbind(wordFreq1[1:10])
      [,1]
stv    135
carv   133
aktx   124
ptx     98
glbs    87
top     64
trade   54
market  52
gainer  48
cytr    45
> findFreqTerms(tdm1, lowfreq=10)
 [1] "carv"          "cytr"          "glbs"          "stock"
 [5] "stv"           "top"           "trch"          "voc"
 [9] "aktx"          "cai"           "gainer"        "sngx"
[13] "ocn"           "spi"           "ophc"          "ptx"
[17] "trade"         "market"        "open"          "snap"
[21] "youtub"        "auph"          "monday"        "ipdn"
[25] "make"          "dsx"           "midday"        "profittraderoom"
[29] "rcki"          "xbit"          "bgg"           "cdti"
[33] "knd"           "pcti"          "pfpt"          "ppc"
[37] "wern"          "loser"         "delt"          "nadl"
[41] "rnva"          "day"           "profit"        "share"
[45] "today"         "amp"           "price"         "short"
[49] "week"          "dgtw"          "dividend"      "grnh"
[53] "jump"          "negat"         "pennystock"    "sentiment"
[57] "updat"         "watch"         "get"           "daytrad"
[61] "optt"          "daili"         "recap"         "fake"
[65] "news"
```

```
> wordFreq2 <- rowSums(m2)
> wordFreq2 <- sort(wordFreq2, decreasing=TRUE)
> cbind(wordFreq2[1:10])
      [,1]
xbit   160
gnmx   140
ontx   138
admp    74
immu    40
offer   38
top     38
loser   37
mnkd    37
stock   34
> findFreqTerms(tdm2, lowfreq=10)
 [1] "asp"        "cbli"       "nadl"       "offer"      "xbit"       "stock"
 [7] "biotech"    "today"      "admp"       "clw"        "futuresopt" "gnmx"
[13] "immu"       "loser"      "mnkd"       "ontx"       "option"     "top"
[19] "announc"    "read"       "xbiotech"   "cancer"     "share"      "sell"
[25] "amp"        "price"      "apri"       "efii"       "fmsa"       "market"
[31] "mat"        "stv"        "inc"        "midday"     "ptx"        "gap"
[37] "crhm"       "declin"     "premarket"  "hmni"       "imnp"       "inuv"
[43] "ocn"        "txmd"       "vnce"       "posit"      "common"     "onconova"
[49] "public"     "data"       "therapeut"  "fizz"       "mxim"       "aevi"
[55] "saga"       "latebloom"  "showtim"    "mutat"      "cntn"
```

Since the top three frequent words for both sets are the names of the stocks themselves, we will be removing these terms from WordFreq1 and WordFreq2 for a better word cloud representation.

```
> wordFreq1 <- sort(wordFreq1[-c(1,2,3)], decreasing=TRUE)
> cbind(wordFreq1[1:10])
       [,1]
ptx      98
glbs     87
top      64
trade    54
market   52
gainer   48
cytr     45
delt     45
cai      39
day      38
```

```
> wordFreq2 <- sort(wordFreq2[-c(1,2,3)], decreasing=TRUE)
> cbind(wordFreq2[1:10])
         [,1]
admp       74
immu       40
offer      38
top        38
loser      37
mnkd       37
stock      34
xbiotech   34
read       33
price      33
```

Generating WordCloud

A function called genWordCloud( ) is created to generate a word cloud. This function takes the wordFreq1 or wordFreq2 as arguments, and passes it further to wordcloud( ) function from the package wordcloud. The color palette to be used in the word cloud is also set within genWordCloud( ) function. Thus, a word cloud is created based on the frequent words and their frequencies passed to this function.

```
library(wordcloud)


genWordCloud <- function(wordFreqList){
palette <- brewer.pal(8,"Dark2")
set.seed(123)
wordcloud(words=names(wordFreqList),
          freq=wordFreqList,
          min.freq=10,
          random.order=F,
          colors=palette)
}

genWordCloud(wordFreq1)
genWordCloud(wordFreq2)
```

WordCloud for Gainer Tweets
The word trade could indicate a bullish market.

Analyzing the loser stocks wordcloud, words such as xbiotech, cancer relates to XBIT since the drop in their shares was due to the inability to get approved for a cancer treating drug. The words announce, onconova, therapeut relates to ONTX because their shares dropped after they announced the stock prices of their increased shares volume. The word mutat relates to GNMX stock tweets, after the company identified novel genetic mutations in response to their drug, which could potentially yield positive results in future.

WordCloud for Loser Tweets

Sentiment Analysis

A sentiment analysis of a tweet analyzes whether the selected tweet conveys a positive, negative or neutral sentiment. The total number of positive words is compared against the total number of negative words in the tweet to determine the overall sentiment.

Loading the Lexicons into the workspace

The pre-defined list of positive and negative words[2], which are the lexicons for the sentiment analysis, are loaded into the workspace. The first six words in each of the lists are displayed using head( ) function.

```
> pos.words = scan('positive-words.txt',
+                  what='character',
+                  comment.char=';')
Read 2012 items
>
> neg.words = scan('negative-words.txt',
+                  what='character',
+                  comment.char=';')
Read 4783 items
> head(pos.words)
[1] "a+"         "abound"     "abounds"    "abundance"  "abundant"   "accessable"
>
> head(neg.words)
[1] "2-faced"    "2-faces"    "abnormal"   "abolish"    "abominable" "abominably"
```

[2] Source: Liu, B., Hu, M., & Cheng, J. (2005). Opinion observer. Proceedings of the 14th international conference on World Wide Web - WWW '05. doi:10.1145/1060745.1060797

Generating Sentiment Scoring Function

This function takes a tweet, or a list of tweets along with a vector of positive and negative words as inputs, and computes a positive or negative integer value as the sentiment score. It returns zero as the value for a tweet conveying neutral sentiment. The function returns the difference between the number of positive word-matches(word in tweet matches a word in positive word list) and number of negative word-matches(word in tweet matches a word in negative word list).

The tweet is pre-processed using the steps below:

1. Punctuation characters, control characters and digits are removed.
2. Convert text to lowercase.
3. Convert lowercase text into vector of words
4. The positive and negative word-matches are computed.
5. The difference between the sum calculated in the previous step is returned.
6. The code also handles an exception that returns the value 0 if the number of positive and negative words are equal to zero, by returning 'NA' instead, in such cases.

```
sentiment <- function(text, pos.words, neg.words) {
  text <- gsub('[[:punct:]]', '', text)
  text <- gsub('[[:cntrl:]]', '', text)
  text <- gsub('\\d+', '', text)
  text <- tolower(text)
  # split the text into a vector of words
  words <- strsplit(text, '\\s+')
  words <- unlist(words)
  # find which words are positive
  pos.matches <- match(words, pos.words)
  pos.matches <- !is.na(pos.matches)
  # find which words are negative
  neg.matches <- match(words, neg.words)
  neg.matches <- !is.na(neg.matches)
  # calculate the sentiment score
  p <- sum(pos.matches)
  n <- sum(neg.matches)
  if (p == 0 & n == 0)
    return (NA)
  else
    return (p - n)
}
```

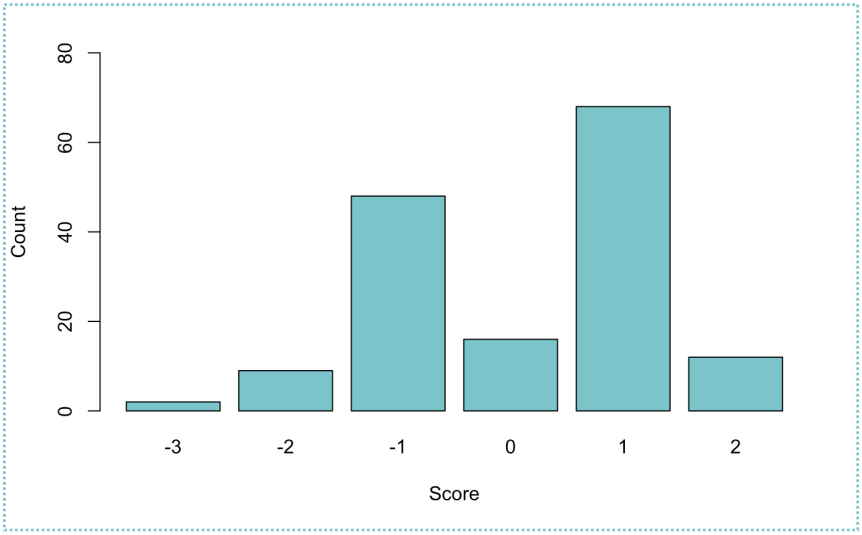
Computing Sentiment Score for Stock Tweets

The sentiment scoring function is applied on specific gainer and loser tweets to verify the accuracy of the function.

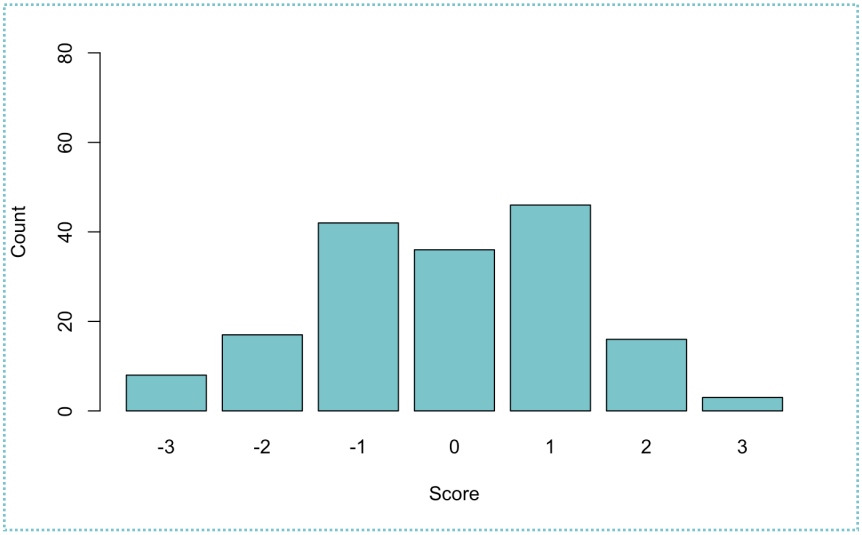
```
> gainer.tweets.text[[5]]
[1] "RT @futuresoption: Top Gainers: $STV +32%. $GLBS +24%. $CAI +22%. $PTX +18%. $CARV +18%.
$VOC +12%. $OPHC +12% #trading"
> sentiment(gainer.tweets.text[[5]], pos.words, neg.words)
[1] 1
>
> gainer.tweets.text[[50]]
[1] "Top Gainers: $STV +32%. $GLBS +24%. $CAI +22%. $PTX +18%. $CARV +18%. $VOC +12%. $OPHC +1
2% #trading"
> sentiment(gainer.tweets.text[[50]], pos.words, neg.words)
[1] 1
>
> gainer.tweets.text[[4]]
[1] "RT @futuresoption: Top Gainers: $STV +32%. $GLBS +24%. $CAI +22%. $PTX +18%. $CARV +18%.
$VOC +12%. $OPHC +12% #trading"
> sentiment(gainer.tweets.text[[4]], pos.words, neg.words)
[1] 1
>
```

```
> loser.tweets.text[[50]]
[1] "Austin immunotherapy company's shares slide ~40% on bad regulatory news $XBIT https://t.
o/iH6QXNejWm"
> sentiment(loser.tweets.text[[50]], pos.words, neg.words)
[1] -1
>
> loser.tweets.text[[4]]
[1] "3 Things In Biotech You Should Learn Today: April 22, 2017 https://t.co/IGw86MbDh0 $BGNE
$BMY $GILD $MRK $XBIT"
> sentiment(loser.tweets.text[[4]], pos.words, neg.words)
[1] NA
```

The function is then applied for all the tweets in the gainers and losers set(gainer.tweets.text and loser.tweets.text). A bar graph of the computed scores are also plotted.



Bar graph for gainer tweets



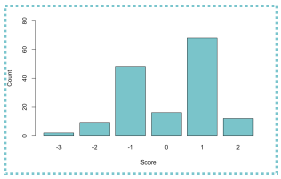
Bar graph for loser tweets

Analysis And Conclusion

A correlation was observed between the public sentiment and market sentiment for the day this study was conducted, especially for the gainers. The three largest gainer stocks for that day(STV, AKTX, CARV) was characterized by a positive sentiment from the twitter data. However, the tweets about the 3 largest loser stocks for that day showed some interesting insights on investor behavior/mood, which this section will be analyzing in detail.

The table below displays the number of tweets associated with each distinct sentiment score for **Gainer stocks**. The total number of positive tweets are higher than the number of negative tweets, and therefore, there exists an overall positive sentiment. Thus, we can deduct a positive correlation between public sentiment and market sentiment. (Note: This analysis is only based on a limited dataset for the purpose of this project)

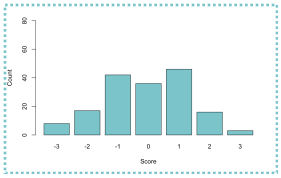
```
> table(gainer.scores)
gainer.scores
-3 -2 -1  0  1  2
 2  9 48 16 68 12
```



Sentiment Score of gainer tweets

However, for the **Loser stocks,** the total number of positive and negative scores are almost the same, with only a slight difference in values. The bar chart displays a normally distributed data. Further analysis was conducted on individual loser stocks, and for comparison purposes, each of their Day Charts from Yahoo Finance was retrieved.

```
> table(loser.scores)
loser.scores
-3 -2 -1  0  1  2  3
 8 17 42 36 46 16  3
```



Total positive = 65
Total Negative = 67
Total Neutral = 36

Sentiment Score of loser tweets

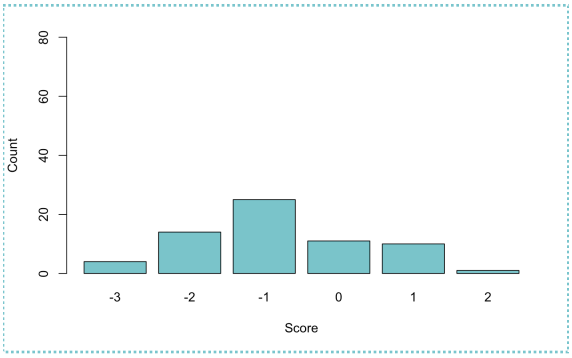
It is interesting to note that all three loser stocks are in the **Biotech industry**, which is known to be a notoriously risky market. Most biotechnology products do not consistently produce the desired results, while others fail to gain acceptance in the marketplace. The investors make a lot of money in cases when a biotech product succeeds, which may be an explanation for the mixed sentiment.

Analysis of Individual Loser stock tweets

1. XBIT (XBIOTECH)

```
> analyze.loser.tweet(tweets.xbit)
loser.scores
-3 -2 -1  0  1  2
 4 14 25 11 10  1
```

Total positive = 11
Total Negative = 43
Total Neutral = 11



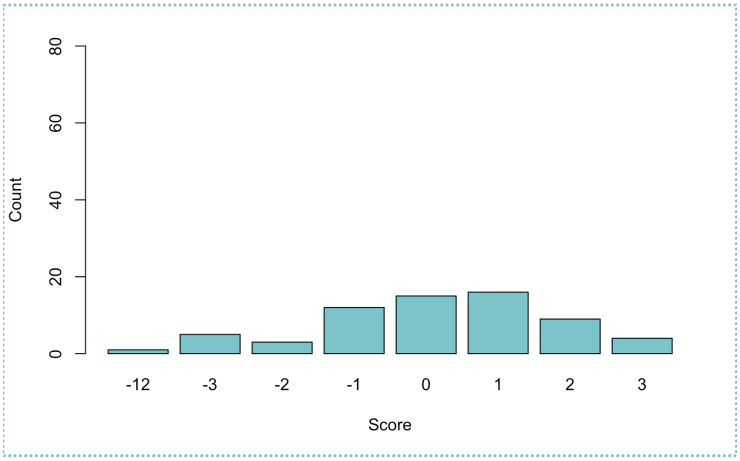
Bar graph of XBIT tweets scores

XBIOTECH(XBIT) stocks made a sharp fall on April 21, 2017 when they failed to convince the European Medicines Agency(EMA) regulators that their new drug for colorectal cancer treatment, called Xilonix is approvable. The trend vote taken by the regulators was negative which indicates that the drug is also unlikely to receive a positive response from Committee for Medicinal Products for Human Use (CHMP) when the official decision is made next month. This could be a possible explanation for the pessimistic public sentiment on Friday, making them believe there is no potential in the stocks to rise again. The analysis of the one-month chart also shows that the company's stock prices were stable until April 21, and the sharp and unhealthy fall in prices could also indicate a tough recovery.

2. GNMX (Aevi Genomic Medicine Inc.)

```
> analyze.loser.tweet(tweets.gnmx)
loser.scores
-12  -3  -2  -1   0   1   2   3
  1   5   3  12  15  16   9   4
```

Total positive = 29
Total Negative = 21
Total Neutral = 15



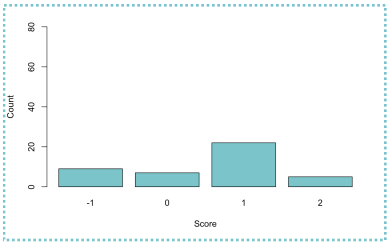
Bar graph of GNMX tweets scores

The bar graph for the GNMX sentiment scores is skewed to the left, indicating that the overall score/sentiment is predominantly positive. The company announced their trial results conducted on patients with adolescent Attention Deficit Hyperactivity Disorder (ADHD), and even though the results did not meet the required results on the ADHD rating scale, the results held significant potential for development in future. This may have caused the public sentiment to be optimistic despite the fact that the stock prices fell sharply from the previous day's close. The one-month chart of the GNMX stocks also indicate that the company stocks continuously faces ups and downs, and the public in general is optimistic about a subsequent stock price recovery.

3. ONTX (Onconova Therapeutics Inc.)

```
> analyze.loser.tweet(tweets.ontx)
loser.scores
-1  0  1  2
 9  7 22  5
```

Total positive = 27
Total Negative = 9
Total Neutral = 7



ONTX shares were down on Friday, April 21 after it priced its stock offering at $2.10 per share, when compared to the share value of $2.49 as of Thursday's close. This was done with the intention of raising capital for their drug, Rigosertib, under trial with higher-risk Myelodysplastic Syndromes, which is known to yield very promising results. The shares may have reduced because the Earnings Per Shares for the investors have now reduced, since the proposed number of shares is 2.5 m, while the current share volume is only approximately 900K. However, the positive sentiment may be related to more investors getting hold of the company's profitable stocks which has the potential to yield positive results with their current drug trial. The tweet shown below from the dataset is a good example of a positive tweet showing optimistic sentiment:

"$ONTX offering more shares has given great opportunity to buy and hold now. You won't need to trade another stock to become wealthy, imo"

Another possible reason for the positive sentiment for the loser stock could be explained by the fact that their drug trial is expected to be active in all 19 countries this quarter, and the public is holding on to their optimistic opinions on their future earnings per share as a result of the trial's success following the expansion.

Comparison of all 3 Largest Loser Stock Tweets

Positive sentiment associated with stock from twitter data

Negative sentiment associated with stock from twitter data

Therefore, as discussed in the previous sections, the overall negative sentiment for XBIT could be associated with the difficulty to recover from the fall in stocks, while the positive sentiment for GNMX and ONTX could be associated with the optimistic mindset of investors who believe in the potential for the companies to perform well in the future despite their fall in the market, by assuming it is a temporary situation.

Bonus Section: Visualization

This section explains the usage of googleVis package for visualizing the stock prices for top three gainer and loser stocks using a Bar Chart. The packages used in this section are googleVis for using Google Chart functions, and quantmod for retrieving current stocks from Yahoo Finance.

Retrieving Stock data using quantmod package:

Using the getQuote function in the quantmod package, the gainer and loser stocks are retrieved and stored as a character list called stock.quotes.

```
# installing and loading required packages
#install.packages("quantmod")
#install.packages("googleVis")
library(quantmod) # For obtaining stock prices
library(googleVis)

# Obtaining stock prices for STV,AKTX,CARV, XBIT, GNMX, ONTX
stock.quotes <- getQuote("STV;AKTX;CARV;XBIT;GNMX;ONTX")
```

	Trade Time <S3: POSIXct>	Last <dbl>	Change <dbl>	% Change <chr>	Open <dbl>	High <dbl>	Low <dbl>	Volume <int>
STV	2017-04-21 04:00:00	1.71	0.44	+34.65%	1.80	1.82	1.62	6316311
AKTX	2017-04-21 04:00:00	20.50	4.13	+25.23%	16.36	21.86	16.08	1881347
CARV	2017-04-21 04:00:00	4.90	0.91	+22.81%	4.45	6.61	4.45	1222993
XBIT	2017-04-21 04:00:00	10.23	-6.79	-39.89%	10.41	10.80	9.80	2617794
GNMX	2017-04-21 04:00:00	1.54	-0.32	-17.20%	1.78	1.78	1.50	1969398
ONTX	2017-04-21 03:59:00	2.11	-0.38	-15.26%	2.23	2.24	2.09	900415

6 rows

Preparing Data for Visualization using googleVis:

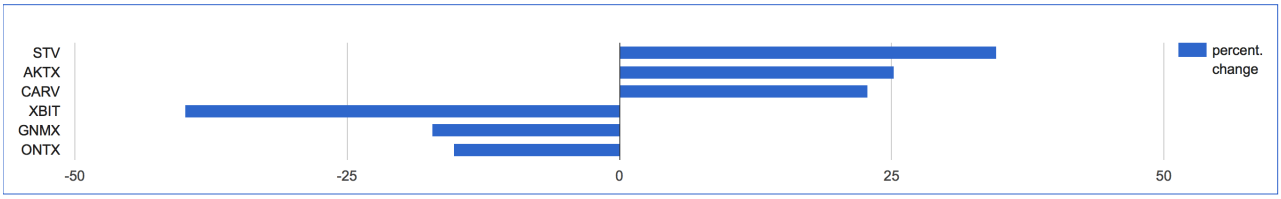
From the stock.quotes list, the % Change values are stored in stock.quotes.char list. Since googleVis requires a data frame as input, the character list needs to be converted into a numeric list, after removing the unwanted symbols(e.g., %, double quotes). This numeric list is then converted into a data frame called 'df' with the corresponding numeric stock values. The bar chart is generated by passing the data frame 'df' into the gvisBarChart function.

```
#Extracting the % Change values and storing as characted vector
stock.quotes.char <- stock.quotes$`% Change`
# cleaning data and converting to numeric format
# E.g. "+34.65%" convert to 34.65
# Remove double quotes and % sign and convert to numeric vector
db <- gsub(",","",stock.quotes.char)
db <- gsub("%","",db)
db<-as.numeric(db)

# Converting numeric vector to dataframe for googleVis
data <- data.frame(percent.change=db)


df=data.frame(stock=c("STV", "AKTX", "CARV", "XBIT","GNMX","ONTX"),
              val1=data)

# Generating Bar Chart
trend.chart<- gvisBarChart(df)
plot(trend.chart)
```



Data: data • Chart ID: BarChartID7dc52f1df4dc • googleVis-0.6.2
htmlFooter R version 3.3.3 (2017-03-06) • Google Terms of Use • Documentation and Data Policy

References

- Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science, 2*(1), 1-8. doi:10.1016/j.jocs.2010.12.007
- Bing, L., Chan, K. C., & Ou, C. (2014). Public Sentiment Analysis in Twitter Data for Prediction of a Company's Stock Price Movements. *2014 IEEE 11th International Conference on e-Business Engineering*. doi:10.1109/icebe.2014.47
- Liu, B., Hu, M., & Cheng, J. (2005). Opinion observer. Proceedings of the 14th international conference on World Wide Web - WWW '05. doi:10.1145/1060745.1060797
- Twitter unveils 'cashtags' to track stock symbols. Retrieved April 22, 2017, from http://money.cnn.com/2012/07/31/technology/twitter-cashtag/
- Hayes, C. A. (2017, April 19). Stock Basics Tutorial. Retrieved April 22, 2017, from http://www.investopedia.com/university/stocks/
- Yahoo Finance Data source. Retrieved April 21, 2017, from https://finance.yahoo.com
- Top 5 Biotech Stocks for 2017 | Investopedia http://www.investopedia.com/investing/biotech-stocks/#ixzz4f61EJrSR
- Loth, R. (2007, April 10). Why do share prices fall after a company has a secondary offering? Retrieved April 23, 2017, from http://www.investopedia.com/ask/answers/07/secondary_offering.asp

Stock news sources:

- Orelli, B. (1970, January 01). Why XBiotech Inc. Got Crushed Today. Retrieved April 23, 2017, from https://www.fool.com/investing/2017/04/21/why-xbiotech-inc-got-crushed-today.aspx#ixzz4f62MYJCF
- ATRA Gains After-hours, ONTX Slumps, XBIT Disappointed, BPMX On Watch. (2017, April 21). Retrieved April 23, 2017, from http://www.nasdaq.com/article/atra-gains-afterhours-ontx-slumps-xbit-disappointed-bpmx-on-watch-20170421-00028#ixzz4f6346Vpe
- Court, E. (2017, April 21). Onconova Therapeutics stock drops 12% after pricing stock offering. Retrieved April 23, 2017, from http://www.marketwatch.com/story/onconova-therapeutics-stock-drops-12-after-pricing-stock-offering-2017-04-21-9915535?siteid=yhoof2&yptr=yahoo

Project Code

The project code is organized into 7 sections:

1. Connecting to Twitter API
2. Extracting 100 tweets for the stock cashtags
3. Creating Data Corpus for tweets sets
4. Applying Transformations
5. Creating the Term Document Matrix
6. Determining frequent words and Generate Word Cloud
7. Sentiment Analysis
8. Analyzing Sentiment Score for Loser Tweets
9. Bonus: Visualization using googleVis package

---
title: "CS688 Term Project"
output: html_notebook
author: Archana Balachandran

 # 1. CONNECTING to Twitter API

#----------- Loading Required Libraries ------

```{r}
install.packages("twitteR")
install.packages("tm")
install.packages("wordcloud")
library(twitteR)
library(tm)
library(wordcloud)
```
#--------- Authenticating Twitter App ---------

```{r}
t.api.key<-"ieaUlcnfMv8vSXYNkX5wFBZTK"
t.api.secret<-"bEbt6ekRgOvWiv8LwsV8L4uzIzhVmU5eoMCl8iQ2ZLUmb29dSx"
setup_twitter_oauth(t.api.key, t.api.secret, access_token=NULL, access_secret=NULL)
save(list=(c("t.api.key","t.api.secret")), file="twitter_credentials.RData")

```

 # 2. EXTRACT 100 tweets for the stock cashtags

```{r}
# GAINER STOCKS
tweets.stv <- searchTwitter('$STV',n=100)
tweets.aktx <- searchTwitter('$AKTX',n=100)
tweets.carv <- searchTwitter('$CARV',n=100)
gainer.tweets<-c(tweets.stv,tweets.aktx,tweets.carv)

# LOSER STOCKS
tweets.xbit <- searchTwitter('$XBIT',n=100)
tweets.ontx <- searchTwitter('$ONTX',n=100)
tweets.gnmx <- searchTwitter('$GNMX',n=100)
loser.tweets<-c(tweets.xbit,tweets.ontx,tweets.gnmx)

# Converting both lists to ASCII

gainer.tweets.text <-
  lapply(gainer.tweets,
        function(t) {
          iconv(t$getText(),
            "latin1", "ASCII", sub="")
                })

loser.tweets.text <-
  lapply(loser.tweets,
         function(t) {
          iconv(t$getText(),
             "latin1", "ASCII", sub="")
         })
```

```
# Verifying accuracy of data

head(gainer.tweets.text,n=2)
length(gainer.tweets.text)
typeof(gainer.tweets.text)

head(loser.tweets.text,n=2)
length(loser.tweets.text)
typeof(loser.tweets.text)
```

 # 3. Creating DATA CORPUS for tweets sets

Inputs:

- gainer.tweets.text
- loser.tweets.text

```{r}
# function that takes a tweet and returns a corpus
createCorpus<- function(tweetset){
  data.source <-VectorSource(tweetset)
  data.corpus<-Corpus(data.source)
  return(data.corpus)
}
# Creating corpora using the createCorpus function
data.corpus1<-createCorpus(gainer.tweets.text)
data.corpus2<-createCorpus(loser.tweets.text)
typeof(data.corpus1)
# Inspecting the corpora
inspect(data.corpus1[1:2])
inspect(data.corpus2[1:2])

```

 # 4. Applying TRANSFORMATIONS

Implement the pre-processing as a function that takes a corpus and returns a pre-processed corpus.

inputs:

data.corpus1

data.corpus2

```{r}

# Defining the transformCorpus() function
transformCorpus<- function(data.corpus){
  content_transformer(function(x, from, to) gsub(from, to, x))

  # convert to lowercase
  data.corpus <-
    tm_map(data.corpus,
           content_transformer(tolower))

  # Defining URL elements for removing URL from text
  removeURL <- function(x) {
    gsub("(http[^ ]*)", "", x)
  }

  # remove URL
  data.corpus <-
    tm_map(data.corpus,
           content_transformer(removeURL))

  # Defining stop words in this context
```

```
english.stopwords <- stopwords("en")
head(english.stopwords)

# remove stopwords
data.corpus <-
  tm_map(data.corpus,
         content_transformer(removeWords),
         english.stopwords)


# Remove punctuation

data.corpus <-
  tm_map(data.corpus,
         content_transformer(removePunctuation))


# remove words that begin with a number
removeNumberWords <- function(x) {
  gsub("([[:digit:]]+)([[:alnum:]])*", "", x)
}

data.corpus <-
  tm_map(data.corpus,
         content_transformer(removeNumberWords))

# Perform stemming

data.corpus <-
  tm_map(data.corpus,
         content_transformer(stemDocument))
```

```
  # Strip white space
  data.corpus <-
    tm_map(data.corpus,
           content_transformer(stripWhitespace))

  return(data.corpus)
}

# Performing transformation
data.corpus1<-transformCorpus(data.corpus1)
data.corpus2<-transformCorpus(data.corpus2)

# Inspecting the two Corpora after applying transformations
inspect(data.corpus1[1:2])
inspect(data.corpus2[1:2])
```

 # 5. Creating the TERM DOCUMENT MATRIX --

--Inputs:

-- data.corpus1

-- data.corpus2

```{r}

# CONSTRUCTING tdm1
# Build the term document matrix for data.corpus1 and inspect part of matrix

tdm1 <- TermDocumentMatrix(data.corpus1)
inspect(tdm1[1:10, 50:60])

# convert TDM to a matrix and view portion of the matrix

m1 <- as.matrix(tdm1)
m1[1:10, 51:60]


# CONSTRUCTING tdm2
# Build the term document matrix for data.corpus2 and inspect part of matrix
tdm2 <- TermDocumentMatrix(data.corpus2)
inspect(tdm2[1:10, 50:60])
# convert TDM to a matrix and view portion of the matrix
m2 <- as.matrix(tdm2)
m2[1:10, 51:60]

# Saving tdm1 and tdm2 as R objects

save(list=("tdm1"),
    file = "GainerTDM.RData")
```

```
load(file="GainerTDM.RData")


save(list=("tdm2"),
    file = "LoserTDM.RData")

load(file="LoserTDM.RData")
```

 # 6. Determining FREQUENT WORDS and Generate WORD CLOUD

-- Calculate the frequency of words, and examine part of the frequencies--

-- Sort the words by descending order of frequency--

-- NOTE: the first 3 rows containing

stock names are removed to get a better wordcloud representation

```{r}
# calculate the frequency of words in m1
wordFreq1 <- rowSums(m1)
wordFreq1 <- sort(wordFreq1, decreasing=TRUE)

# Examine the top ten words
cbind(wordFreq1[1:10])

# All frequent terms having atleast 10 occurrences
findFreqTerms(tdm1, lowfreq=10)


# calculate the frequency of words in m2
wordFreq2 <- rowSums(m2)
wordFreq2 <- sort(wordFreq2, decreasing=TRUE)
# Examine the top ten words
```

```
cbind(wordFreq2[1:10])

# frequent terms
findFreqTerms(tdm2, lowfreq=10)



#Removing stock names from  WordFreq1 and WordFreq2
wordFreq1 <- sort(wordFreq1[-c(1,2,3)], decreasing=TRUE)
cbind(wordFreq1[1:10])

wordFreq2 <- sort(wordFreq2[-c(1,2,3)], decreasing=TRUE)
cbind(wordFreq2[1:10])

```
```{r}
# Word Cloud Generation
library(wordcloud)
genWordCloud <- function(wordFreqList){
  palette <- brewer.pal(8,"Dark2")
  set.seed(123)
  wordcloud(words=names(wordFreqList),
            freq=wordFreqList,
            min.freq=10,
            random.order=F,
            colors=palette)
}
genWordCloud(wordFreq1)
genWordCloud(wordFreq2)
```
```
```

 # 7. --- SENTIMENT ANALYSIS

```{r}
getwd()
setwd("/Users/archanabalachandran/Documents/")

# Lexicons
pos.words = scan('positive-words.txt',
                 what='character',
                 comment.char=';')

neg.words = scan('negative-words.txt',
                 what='character',
                 comment.char=';')

head(pos.words)
head(neg.words)

# Sentiment determining function

sentiment <- function(text, pos.words, neg.words) {
  text <- gsub('[[:punct:]]', '', text)
  text <- gsub('[[:cntrl:]]', '', text)
  text <- gsub('\\d+', '', text)
  text <- tolower(text)
  # split the text into a vector of words
  words <- strsplit(text, '\\s+')
  words <- unlist(words)
  # find which words are positive
  pos.matches <- match(words, pos.words)
```

```
  pos.matches <- !is.na(pos.matches)
  # find which words are negative
  neg.matches <- match(words, neg.words)
  neg.matches <- !is.na(neg.matches)
  # calculate the sentiment score
  p <- sum(pos.matches)
  n <- sum(neg.matches)
  if (p == 0 & n == 0)
    return (NA)
  else
    return (p - n)
}


# Verifying sentiment score for specific Gainer tweets
gainer.tweets.text[[2]]
sentiment(gainer.tweets.text[[2]], pos.words, neg.words)

gainer.tweets.text[[5]]
sentiment(gainer.tweets.text[[5]], pos.words, neg.words)

gainer.tweets.text[[50]]
sentiment(gainer.tweets.text[[50]], pos.words, neg.words)

gainer.tweets.text[[4]]
sentiment(gainer.tweets.text[[4]], pos.words, neg.words)

# Verifying sentiment score for specific Loser tweets
loser.tweets.text[[2]]
sentiment(loser.tweets.text[[2]], pos.words, neg.words)
```

```
loser.tweets.text[[5]]
sentiment(loser.tweets.text[[5]], pos.words, neg.words)

loser.tweets.text[[50]]
sentiment(loser.tweets.text[[50]], pos.words, neg.words)

loser.tweets.text[[4]]
sentiment(loser.tweets.text[[4]], pos.words, neg.words)
```

```{r}
sink(tempfile())

# SENTIMENT SCORE FOR GAINER TWEETS
gainer.scores <- sapply(gainer.tweets.text,
                        sentiment,
                        pos.words, neg.words)
table(gainer.scores)


barplot(table(gainer.scores),
        xlab="Score", ylab="Count",
        ylim=c(0,80), col="cadetblue3")
```

 # Computing overall sentiment

```
total.neg.score <-table(gainer.scores)[[1]]+table(gainer.scores)[[2]]+table(gainer.scores)[[3]]
```

```
total.neg.score

total.neutral.score <-table(gainer.scores)[[4]]
total.neutral.score

total.pos.score<-table(gainer.scores)[[5]]+table(gainer.scores)[[6]]
total.pos.score

if(total.pos.score>total.neg.score)
{
  print("Positive Sentiment")
} else { print("Not positive sentiment")}


# Data frame of scores and tweets

stv.vector <- sapply(gainer.tweets.text,
                     function (t) {(t)})
x1 <- data.frame(Score=gainer.scores, Text=stv.vector)
View(x1)

```
# SENTIMENT SCORE FOR LOSER TWEETS
loser.scores <- sapply(loser.tweets.text,
                     sentiment,
                     pos.words, neg.words)
table(loser.scores)
barplot(table(loser.scores),
        xlab="Score", ylab="Count",
        ylim=c(0,80), col="cadetblue3")
```
```

```
total.neg.score <-table(loser.scores)[[1]]+table(loser.scores)[[2]]+table(loser.scores)[[3]]
total.neg.score

total.neutral.score <-table(loser.scores)[[4]]
total.neutral.score

total.pos.score<-table(loser.scores)[[5]]+table(loser.scores)[[6]]
total.pos.score

if(total.pos.score<total.neg.score)
  {
print("Negative Sentiment")
} else { print("Not Negative sentiment")}

# Data frame of scores and tweets

stv.vector <- sapply(loser.tweets.text,
                     function (t) {(t)})
x2 <- data.frame(Score=loser.scores, Text=stv.vector)
View(x2)
```

```{r}
```

8. Analyzing Sentiment Score for Loser Tweets

```{r}
# Function for LOSER ANALYSIS

# Analyzing individual stock info for loser stock-because the positive and negative sentiments are almost the same

analyze.loser.tweet <-function(losertweet){
# Computing overall sentiment
sentiment <- function(text, pos.words, neg.words) {
  text <- gsub('[[:punct:]]', '', text)
  text <- gsub('[[:cntrl:]]', '', text)
  text <- gsub('\\d+', '', text)
  text <- tolower(text)
  # split the text into a vector of words
  words <- strsplit(text, '\\s+')
  words <- unlist(words)
  # find which words are positive
  pos.matches <- match(words, pos.words)
  pos.matches <- !is.na(pos.matches)
  # find which words are negative
  neg.matches <- match(words, neg.words)
  neg.matches <- !is.na(neg.matches)
  # calculate the sentiment score
  p <- sum(pos.matches)
  n <- sum(neg.matches)
  if (p == 0 & n == 0)
    return (NA)
  else
    return (p - n)
```

```
}


loser.tweets.text <-
  lapply(losertweet,
              function(t) {
                iconv(t$getText(),
                          "latin1", "ASCII", sub="")
         })
# applying sentiment analysis function
loser.scores <- sapply(loser.tweets.text,
                  sentiment,
                  pos.words, neg.words)
# printing sentiment scores
print(table(loser.scores))

#displaying bar chart
barplot(table(loser.scores),
       xlab="Score", ylab="Count",
       ylim=c(0,80), col="cadetblue3")

}
#passing each loser Stock tweets to the analysis function
analyze.loser.tweet(tweets.xbit)
analyze.loser.tweet(tweets.ontx)
analyze.loser.tweet(tweets.gnmx)
analyze.loser.tweet(loser.tweets)

```
```
```

9. Bonus: Visualization using googleVis package

```{r}
# installing and loading required packages
#install.packages("quantmod")
#install.packages("googleVis")
library(quantmod) # For obtaining stock prices
library(googleVis)

# Obtaining stock prices for STV,AKTX,CARV, XBIT, GNMX, ONTX
stock.quotes <- getQuote("STV;AKTX;CARV;XBIT;GNMX;ONTX")
#Extracting the % Change values and storing as characted vector
stock.quotes.char <- gainer.quotes$`% Change`

# cleaning data and converting to numeric format
# E.g. "+34.65%" convert to 34.65
# Remove double quotes and % sign and convert to numeric vector
db <- gsub(",","",stock.quotes.char)
db <- gsub("%","",db)
db<-as.numeric(db)
# Converting numeric vector to dataframe for googleVis
data <- data.frame(percent.change=db)
df=data.frame(stock=c("STV", "AKTX", "CARV", "XBIT","GNMX","ONTX"),
            val1=data)

# Generating Bar Chart
trend.chart<- gvisBarChart(df)
plot(trend.chart)
```

------ END OF PROJECT CODE -------