Capstone Project

Financial News Analytics

Trend Analysis of Market Securities based on Sentiment Analysis of News and Social Media

Submitted by:

|  |  |
| --- | --- |
| **Name** | **Student Id** |
| Abhishek Khanna | 71510003 |
| Sonu Krishnan | 71510076 |
| Sumit Janmejai | 71510081 |
| Venkata Pilla | 71510088 |

Acknowledgements

We would like to take this opportunity to express our profound gratitude and deep regards to our project sponsor, Deloitte Consulting, for providing the opportunity to learn and work on challenging Financial News Analytics – Sentiment Analysis and also to our project mentor Reema, for her efforts in facilitating this Capstone Opportunity. We are deeply indebted to all of them and welcome this opportunity to benefit further from their contribution. We wish to express our special thanks to Hemanth from ISB, Suresh Kumar, Chandra Narra and Sandeep Kumar Sharma from Deloitte and our other batch mates who have provided their valuable suggestions.

* Abhishek, Sonu, Sumit, Venkat

Contents

[Introduction 4](#_Toc442909370)

[Areas of Impact 4](#_Toc442909371)

[Project Approach 4](#_Toc442909372)

[Project Workflow 5](#_Toc442909373)

[Project Phases 6](#_Toc442909374)

[Phase I: Exploratory Analysis and Data Collection 6](#_Toc442909375)

[Phase II: Data Preparation 10](#_Toc442909376)

[Phase III: Model Building and Validation 12](#_Toc442909377)

[Phase IV: Visualization and Dashboards 13](#_Toc442909378)

[Conclusion and Observations 16](#_Toc442909379)

[References 17](#_Toc442909380)

[Annexure 18](#_Toc442909381)

[Twitter Sniffer Program 18](#_Toc442909382)

[Transformation 23](#_Toc442909383)

[Add Sentiment columns 25](#_Toc442909384)

[Trend Visualization 29](#_Toc442909385)

[Corpus 31](#_Toc442909386)

[Scrapping Relevant data 31](#_Toc442909387)

[Meta XTS 32](#_Toc442909388)

[Pre Process the Corpus 33](#_Toc442909389)

[Score Function 34](#_Toc442909390)

# Introduction

News from various parts of the world has a strong relationship with the market (investor) sentiments as markets react sharply and promptly to the news and the sentiment they carry. Effective models that incorporate news data are required by all investors (be it Fund Houses, Brokers or Investment Bank Managers). The idea behind News Analytics is to automate human thinking and reasoning to come up with a quantifiable model and enable users with the decisive capability to predict gains or control risk.

The Project Objectives are:

1. To Perform Trend Analysis of securities based on Market Sentiments through Sentiment Analysis on Financial News and Social Media updates
2. To come up with a Sentiment score or Sentiment polarity for each security, sector or benchmark by analyzing Online Financial and Generic news, Market data and Social Media updates.

This project aims to use Statistical and Analytical tools in Text Analytics on the sourced data to perform the following:

1. Sentiment Score and Mapping for each security/sector/benchmark
2. Overall market trends related to news
3. Visualizations and dashboards complementing the above

# Areas of Impact

* The portfolio specific investment advice given by Investment Management firms based on an overall sentiment from a model that captures the sentiment from financial news and social media content provides much more qualitative insights to potential investors in the markets.
* A study of the variation of security prices corresponding to the variation of sentiment polarity for securities in each sector and markets would provide good insights to Investment Management firms and Fund Broker houses.

# Project Approach

The approach taken is defined below:

* Identification of a Specific Market/Region and Securities to focus on.
* Identification of various Financial and General News websites, Social Media sites to source the information from.
* Assigning weightage to each source of data based on the Reliability factor.
* Design and develop scripts in “R” to collect the data from the identified sources (data scraping), pre-process (data cleaning) and making it ready for analysis.
* Perform various Text Analytics on the data using NLP techniques. Define a data dictionary and generation of word cloud, bi-grams and tri-grams for analysis.
* Arriving at a final score ranging from -1 to +1 for each Security that depicts the negative or the positive online sentiment based on the information.
* Perform the above analysis for Securities as well as various sectors identified.
* Support various visualizations using R and Tableau.

# Project Workflow

**Phase 1:** Exploratory Analysis and Data Collection

* Identification of a specific market/region and securities to focus on.
* Identification of various Financial and General News websites, Social Media sites to source the information from.

**Phase 2:** Data preparation

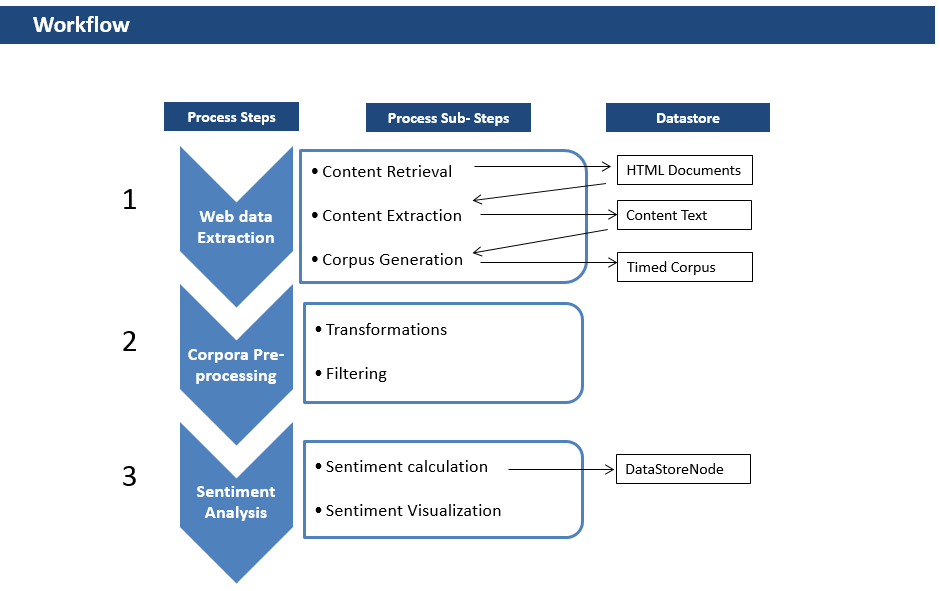
* Designing of various scripts in R to collect the data from the identified sources (data scraping), pre-process (data cleaning) and making it ready for analysis

**Phase 3:** Model building and validation

* Perform various text analytics on that using NLP techniques. Defining a data dictionary and generation of word-cloud, bi-grams and tri-grams for analysis.
* Arriving at a final score ranging from -1 to +1 for each security that depicts the negative or the positive online sentiment about a particular security

**Phase 4:** Writing the report and summarizing the findings

* Development of dashboards and visualizations in Tableau to see the variation in market securities against the sentiment polarities
* Observations, visual analysis and project report documentation



# Project Phases

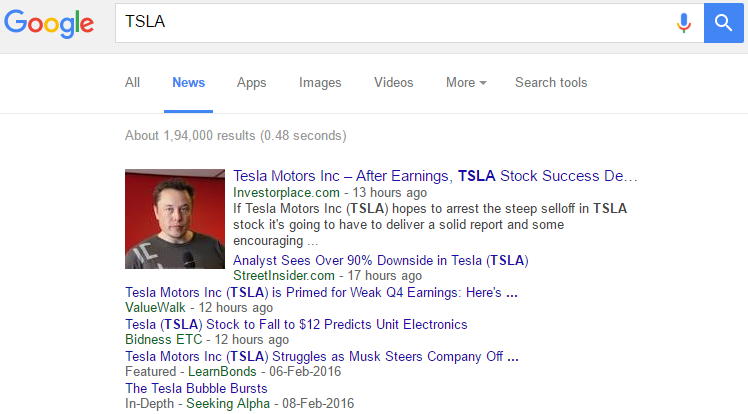
## Phase I: Exploratory Analysis and Data Collection

**Description of Data Collected:**

* The Market and Sector identified for the Project Scope is the US Markets and Automobile sector respectively.
* The Securities identified are :
  + Ford Motor Company (Stock code: F)
  + Tesla Motors Inc. (Stock code: TSLA)
  + General Motors Company (Stock code: GM)
* Data sources
  + **Online news:**

**Google News** is identified as News source as it is a News aggregator and has a collection of News content from various websites. Also this is being widely used by Investor community for Market and Industry updates.

**Sample Google News for Tesla (TSLA)**

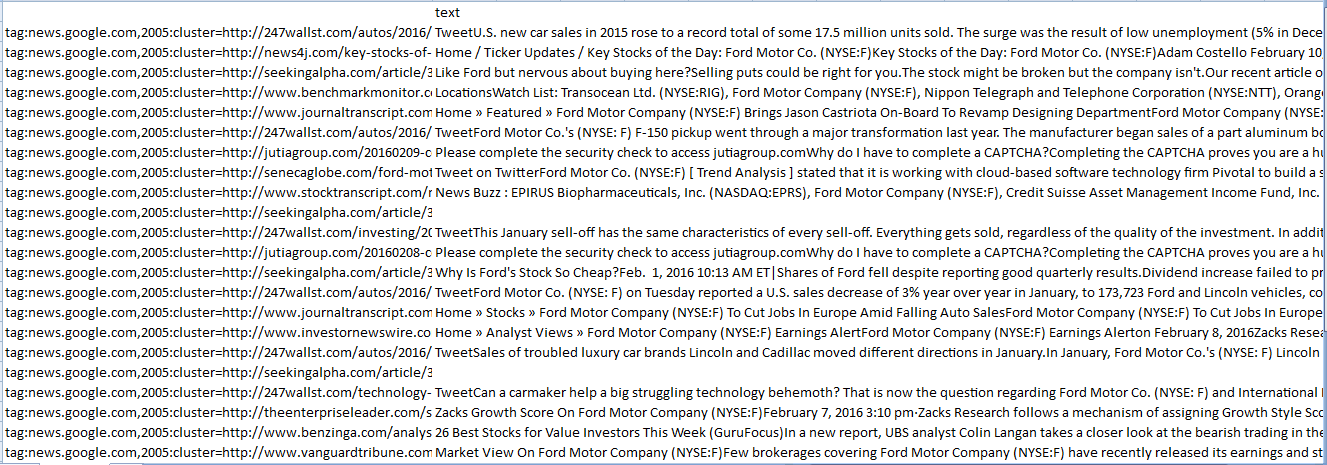


From the above news, we can clearly see that there are both positive and negative news for a particular day where Business forums are speculating the Q4 performance and thereby the Stock Price.

URL ->

[**https://www.google.co.in/search?hl=en&gl=in&tbm=nws&authuser=0&q=TSLA&oq=TSLA&gs\_l=news-cc.3..43j0l2j43i53.2556.4495.0.5169.4.4.0.0.0.0.420.824.2j0j1j0j1.4.0...0.0...1ac.1.gSbIk98vZUA**](https://www.google.co.in/search?hl=en&gl=in&tbm=nws&authuser=0&q=TSLA&oq=TSLA&gs_l=news-cc.3..43j0l2j43i53.2556.4495.0.5169.4.4.0.0.0.0.420.824.2j0j1j0j1.4.0...0.0...1ac.1.gSbIk98vZUA)

**Sample Google News Extract from Internet for Ford Motors**



**Sample Google Finance Extract from Internet for Ford Motors**



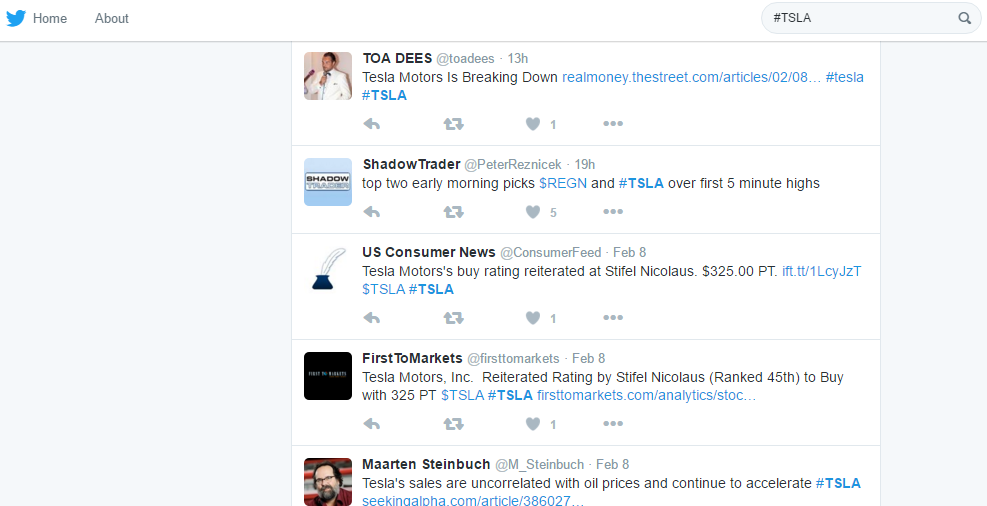
* + **Social Media:**

We sourced data from **Twitter** based on the securities’ company names and stock codes both.   
**Challenges:** The Twitter API used allowed us to fetch tweets only up to one week in the past. Hence to overcome this limitation we set up a cluster and engine to run the API once every week. We were able to have tweets collected for the past several weeks.

**Sample Twitter Output**

Similar to the Google News, the below snapshot of the tweets from Twitter website for #TSLA also clearly shows Positive as well as Negative sentiment tweets. There are tweets reinstating the “Buy” status on the stock while there is a tweet stating “Tesla Motors is breaking down”.

URL -> <https://twitter.com/search?q=%23TSLA&src=typd>



* **Web Data extraction:**

This is done through RSS Feeds and APIs.

* + **RSS feeds** Rss icon**:**

News Sources such as Google News, Google Finance, Yahoo News and Yahoo Finance provide RSS (**R**ich **S**ite **S**ummary; originally RDF Site Summary; often called **R**eally **S**imple **S**yndication) feed options. The feeds provide Metadata of News content about any topic i.e. the URLs of external sites where content resides.

This is a 2-step procedure involving:

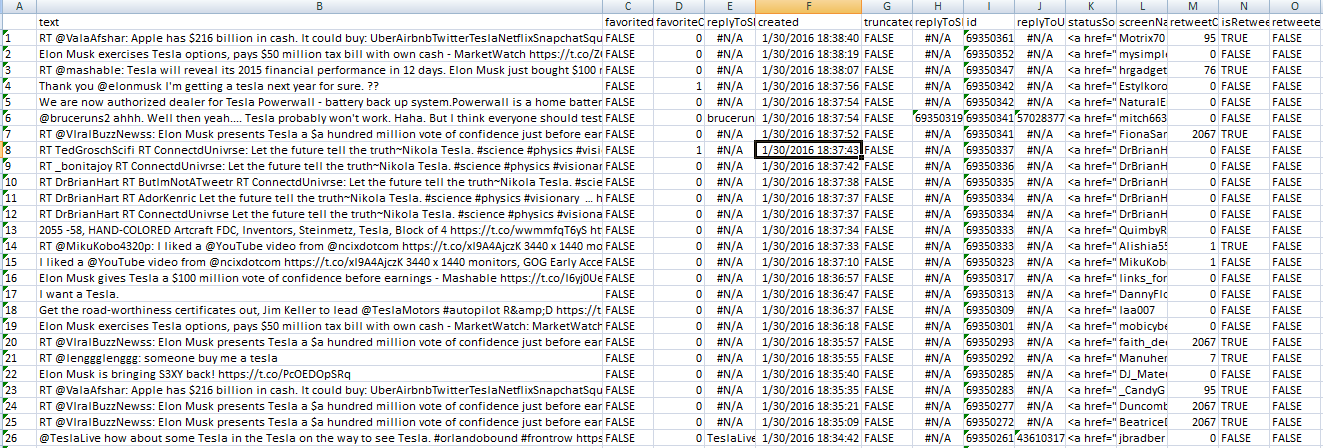
1. Downloading metadata feeds
2. Downloading content sites
   * **APIs:**News Sources: Reuters Spotlight, NY Times, Yahoo BOSS, Bing

Twitter: Publically available REST API. Twitter API poses the following limitations:

1. Number of tweets that can be fetched – rate limit
2. How far back can we go while collecting data ~ 7 days

Hence we need to set up a periodic run for collecting Twitter data on a weekly basis. More information on REST API can be found <https://dev.twitter.com/rest/public>

Sample of the Twitter data scraped from the internet using the API is shown below,



Code attached in Annexure 1.

## Phase II: Data Preparation

**Methods employed:**

1. **Content extraction from RSS Feeds**

* Blogs and News Sites often contain side bars, suggested reads, Ads and other unwanted contents in addition to the required News content
* To extract the key content from the Web Page and ignore the noise, the following technique has been used.

***Reference:*** *A detailed survey of HTML content extraction is given by [Gottron, 2008]:   
“Content Extraction: Identifying the Main Content in HTML Documents”*

* The Technique:

<html>\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_</html>

Segment and Content Classification

Heuristic Rules

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

HTML Document

Extracted Content

As shown above, the combination of Largest Block of String (LBS) and classification tasks namely CS+LBS yield the best performance for Blogs and News datasets. The following steps are required:

* + Examine each HTML sub node from top to bottom
  + If text length/total length < threshold then drill down
  + Select text from sub node with longest text length

1. **Twitter REST API**

* Twitter is one of the popular Social Media platforms where user can communicate with wider audience instantly. Additionally this platform is popular in terms of the Media attention it receives and hence makes it easier to find and follow conversations.
* Twitter has hashtag (#) norms which make it easier gathering, sorting and expanding searches. When collecting information related to major incidents, news stories, all such events tend to be centered on hashtag.
* However there is a limitation on volume and timeline of data that can be captured using the Twitter API.

**Limitations:**

* Twitter allows capture of only 300-400 tweets every 15 minutes per handle.
* Allows to capture data up to 7 days in past.

**Our Approach:**

1. Shortlisted the appropriate hashtags:
   1. FORD: #fordmotors #ford
   2. TESLA: #TeslaMotors #Tesla #TSLA
   3. General Motors : #GeneralMotors #GM
2. Executed the code every weekend to capture the conversations on the above hashtags and dumped them in an excel file.

## Phase III: Model Building and Validation

**Pre-Processing:**

Some cleaning and preparation of corpus required before sentiment classification operations.

Standard transformations like:

* Conversion to lower case
* Removal of stop-words
* Removal of punctuations
* Stemming
* Stripping of white space
* And more depending on quality of relevant text extraction

“tm” package in R has easy functions to achieve all the above.

**Sentiment Analysis:**

**Sentiment Calculation**

Bag of Words model is used for sentiment calculation. Advantages of such an approach are:

1. No need of labelled training data
2. Quick turn-around

*Pre requisites for the sentiment calculation are:*

1. Document Term Matrix (tm package in R)
2. Lexicons (dictionary of positive/negative terms)

SentiWordnet, General Inquirer from Harvard University, NTU Sentiment Dictionary, OpinionFinder’s Subjectivity Lexicon are some of the freely available and widely regarded lexicons.

Following indicators of sentiment are taken directly from the Lydia Sentiment Analysis framework as explained by Wenbin Zhang and Steven Skiena in *“Trading Strategies To Exploit News Sentiment” :*

“*The Lydia sentiment data consists of time series of favorable (positive) and unfavorable (negative)*

*words co-referenced with occurrences of each named entity (here denoting companies). Let* ***p*** *and* ***n*** *denote the number of raw positive and negative references to a given entity, which occurs a total of N times in the corpus (including neutral references).*

*Then we derive the following natural sentiment/subjectivity measures from these raw counts:*

* polarity = (p − n)/(p + n)
* subjectivity = (n + p)/N
* pos refs per ref = p/N
* neg refs per ref = n/N
* senti diffs per ref = (p − n)/N

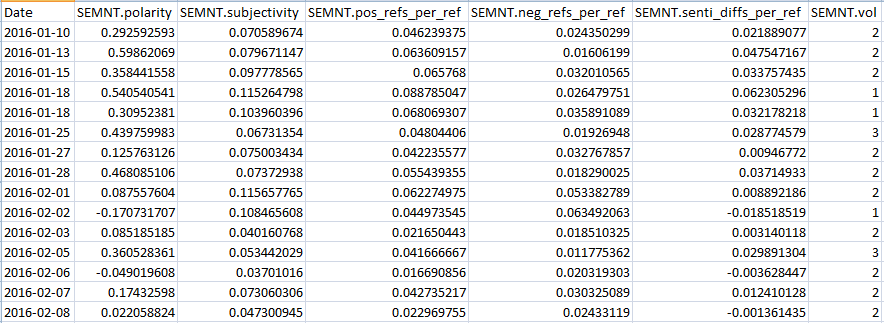
*“These measures are not highly correlated with raw sentiment counts and they can provide additional information that raw data cannot. Therefore, with them we will be able to avoid multicollinearity during linear analysis”*

Reference: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.182.8687&rep=rep1&type=pdf>

**Model Output**

The output of the model would be a file that lists the daily values of net polarity and subjectivity values for each of the securities identified.

A snapshot of the output file is shown below:

**

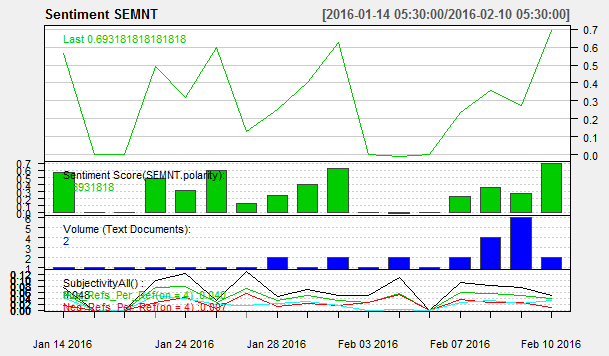
## Phase IV: Visualization and Dashboards

The following visualizations have been derived from the model output and the values of the EOD security prices on respective days:

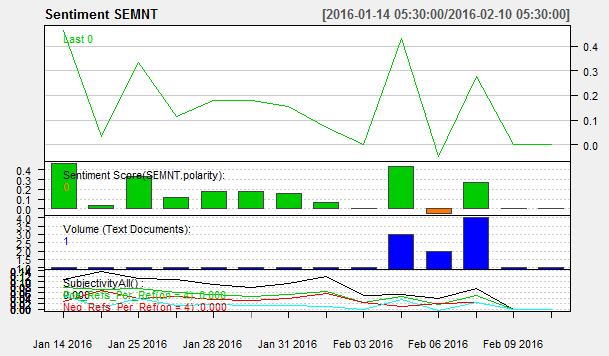
1. Daily volume trend of number of Tweets and number of web extraction for each of the securities
2. Time series trends visual on aggregated Sentiment Polarity for each of the securities
3. Time series trends visual on Sentiment Subjectivity for each security
4. A time series analysis of the variation of actual security prices on the markets with respect to the positive and negative sentiments for each of the securities
5. A time series analysis of the variation of actual security prices on the markets with respect to the aggregated sentiment polarity for each of the securities
6. A time series analysis of the variation of actual security prices against the aggregated sentiment polarity of the (T-1) day for each of the securities

The Model output graphs show the Sentiment, Polarity and Volume in the transaction. Below are the sample for the Google News and Google Finance data.

**Google News Data Analysis**

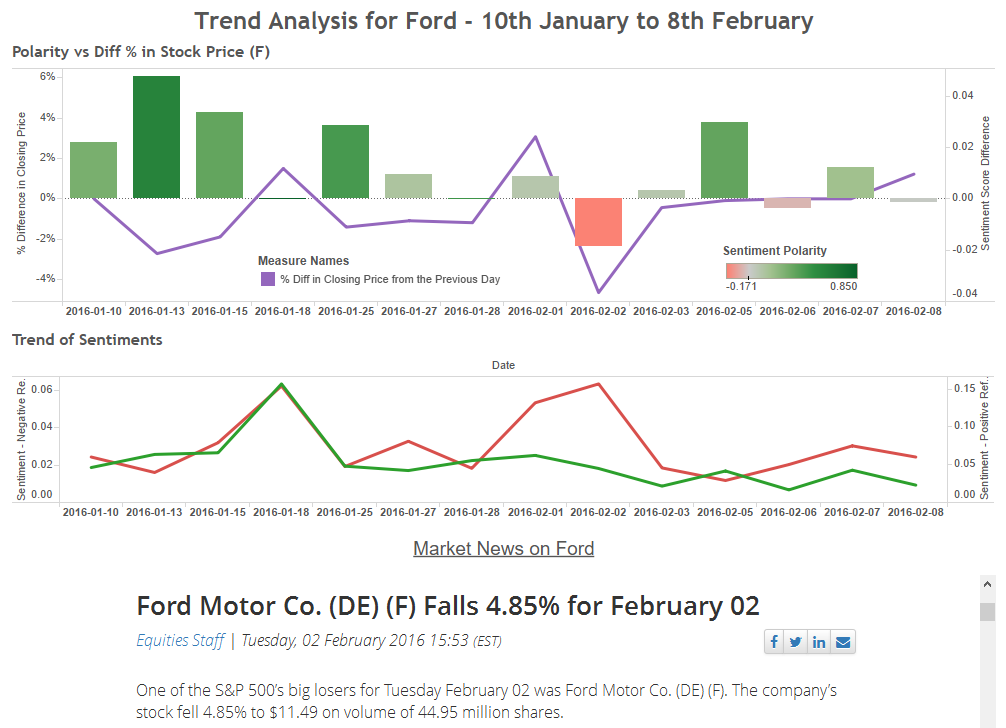


**Google Finance Data Analysis**



**Tableau Dashboard showing Trend Analysis – Polarity vs % change in Stock Price for Ford**

Below is dashboard created using Tableau showing the Polarity of News gathered and Percentage Difference in Stock Price for Ford (F) along with Trend in Sentiments. We can clearly see that the sentiments for Ford 2nd Feb was negative leading to drop in Stock Price by 4.85% (news from website also attached)



# Conclusion and Observations

One of the limitations observed while preparing the model is the reliability of Twitter Data. As Organizations are aware of the twitter trend and how the sentiments impact the trend of their stock market, they run various positive campaigns such as “Fiesta Movement” causing buzz of the product on the social media trying to divert focus on low Sales of cars during the season or Quarterly Earnings Results etc. (<http://www.bu.edu/today/2013/how-ford-became-a-leader-in-social-media/>). Also the amount of cleaning efforts is high for Twitter owing to various styles and lingo of tweets. This also brings focus to the relevancy of the data. Any news such as Product Recall is a negative sentiment.

Through Text Analytics, we are able to understand the correlation of the news sentiment to the movement of stock prices in the Market. The current approach given the limitation of time and resources is limited to retrospective processing of data i.e. scraps through the internet for relevant data from various sources, processes the same and derives the trend in the sentiments of the data. This is then correlated to the movement of the stock prices. As we can see with the dashboards, the correlation is strong.

The way-forward for this project is to be able to scrap the data from various data sources in real-time there by processing their sentiments and providing the trend as a tool to investors (typically the day traders) in taking their stand (either long or short) accordingly. One suggestion is to move the model to cloud services such as AWS and have it run through the day scrapping real time news and tweets and process the sentiment score and through the trend enable the user to take the buy/sell call accordingly.

# References

* + - 1. Deloitte -> <http://www2.deloitte.com/us/en.html>
      2. Content extraction - identifying the main content in HTML document by Thomas Gottron - <https://www.researchgate.net/publication/220690546_Content_extraction_-_identifying_the_main_content_in_HTML_documents>
      3. Trading Strategies to Exploit Blog and News Sentiment by Wenbin Zhang, Steven Skiena - <http://www.aaai.org/ocs/index.php/ICWSM/ICWSM10/paper/view/1529>
      4. Google News Link => <https://news.google.co.in/nwshp?hl=en&tab=wn>
      5. Twitter Website => <https://twitter.com/>
      6. REST API => <https://dev.twitter.com/rest/public>
      7. Plugins => <https://github.com/mannau/tm.plugin.sentiment>

# Annexure

## Twitter Sniffer Program

|  |
| --- |
| ############# Twitter Data Capture ########### |
| #TO capture tweets related to Tesla , Ford and General Motors |
| ####### Handles being sniffed ######### |
| ### Tesla: #TeslaMotors #Tesla #TSLA |
| ### FORD: #ford #fordmotors |
| ### General Motors: #generalmotors #gm |
| # Fetching required packages |
| install.packages("twitteR") |
| install.packages("RCurl") |
| install.packages("RJSONIO") |
| install.packages("stringr") |
| install.packages('base64enc') |
| install.packages('jsonlite') |
| install.packages("xlsx") |
|  |
| library(jsonlite) |
| library(twitteR) |
| library(RCurl) |
| library(RJSONIO) |
| library(stringr) |
| library(base64enc) |
| library(xlsx) |
|  |
| # Declare Twitter API Credentials |
| api\_key = "9besLwERF5I6UARDLVd1OjRBn" |
| api\_secret = "6O9NlAp8owLS12IjZDZudOfDnmFAMgAfqpIaf9AqGUJM26Mkbm" |
| access\_token = "141123071-s20Z93Nizv84W0h8WVMNeYJrMl8sEyzWfJpFqZfF" |
| token\_secret = "7AB0Nw4fhEwmHhpHgeABj4ZBigfdnvXIyHpRxhBNEljCN" |
|  |
| twitter\_log = file("twitter\_sniffer.log",open = "a") |
|  |
| putMeTOSleepCounter\_Ford = 0 |
| putMeTOSleepCounter\_GM = 0 |
| putMeTOSleepCounter\_Tesla = 0 |
|  |
| sleepCounter = 0 |
| keepCapturing = TRUE |
|  |
|  |
| # Create Twitter Connection |
| setup\_twitter\_oauth(api\_key, api\_secret, access\_token, token\_secret) |
|  |
| cat(paste("\*\* Creating Securties Data Frame ",Sys.Date()," \*\*\n"),file=twitter\_log) |
|  |
| # Defining the securities to be captured |
| securities = c("Ford","Tesla","General Motors") |
| hastags = c("'fordmotors' OR #ford OR #fordmotors", |
| "'TeslaMotors' OR 'Tesla' OR 'TSLA' OR #TeslaMotors OR #Tesla OR #TSLA", |
| "'GeneralMotors' OR #generalmotors OR #gm") |
|  |
| ford\_dataFrame = data.frame() |
| tesla\_dataFrame = data.frame() |
| generalmotors\_dataFrame = data.frame() |
|  |
| securities\_toCapture = data.frame(securities,hastags) |
|  |
| names(securities\_toCapture) <- c("Security","HashTags") # variable names |
|  |
| ##Tweet sniffing |
|  |
| while(keepCapturing){ |
| if(putMeTOSleepCounter\_Ford == 3 & putMeTOSleepCounter\_GM == 3 & putMeTOSleepCounter\_Tesla == 3 ){ |
| print("## Didnt get any data for last 3 tries.. Sleeping for 15 mins before retry.##") |
| cat("## Didnt get any data for last 3 tries.. Sleeping for 15 mins before retry. ##\n",file = twitter\_log) |
| Sys.sleep(900) |
| putMeTOSleepCounter\_Ford = 0 |
| putMeTOSleepCounter\_GM = 0 |
| putMeTOSleepCounter\_Tesla = 0 |
| sleepCounter = sleepCounter + 1 |
| } |
| if(sleepCounter == 3){ |
| print("ERROR: Didnt fetch data after 15 mins of sleep 3 times , time to quit !!") |
| cat("ERROR: Didnt fetch data after 15 mins of sleep 3 times , time to quit !!\n",file = twitter\_log) |
| keepCapturing = FALSE |
| break |
| } |
|  |
| #Capture previous data |
| for(i in 1:nrow(securities\_toCapture)) { |
| data <- securities\_toCapture[i,] |
|  |
| temp\_dataFrame = data.frame() |
|  |
| switch(as.character(data$Security), |
| 'Ford'={ |
| if(putMeTOSleepCounter\_Ford == 3){ |
| cat(paste("WARNING: No data fetched for last 3 tries for ",data$Security,", skipping it!!\n")) |
| cat(paste("WARNING: No data fetched for last 3 tries for ",data$Security,", skipping it !!\n"),file = twitter\_log) |
| next |
| } |
|  |
| cat(paste("\*\*\*\*\*\*\*\*\* Capturing Past Data for: ",data$Security," for tags: ",data$HashTags)," \*\*\*\*\*\*\*\*\*\n") |
| cat(paste("\*\*\*\*\*\*\*\*\* Capturing Past Data for: ",data$Security," for tags: ",data$HashTags)," \*\*\*\*\*\*\*\*\*\n",file = twitter\_log) |
|  |
| tweets = searchTwitter(toString(data$HashTags) , |
| n=100, lang="en", maxID = data$TailId) |
| temp\_dataFrame = twListToDF(tweets) |
|  |
|  |
| cat(paste("\*\*\* Fetched : ",nrow(temp\_dataFrame)," tweets for Security: ",data$Security)," \*\*\* \n") |
| cat(paste("\*\*\* Fetched : ",nrow(temp\_dataFrame)," tweets for Security: ",data$Security)," \*\*\* \n",file = twitter\_log) |
|  |
| if(nrow(temp\_dataFrame) == 1 ){ |
| print(" \*\* Insufficient tweets \*\* Skipping this iteration") |
| cat(" \*\* Insufficient tweets \*\* Skipping this iteration\n",file = twitter\_log) |
| putMeTOSleepCounter\_Ford = putMeTOSleepCounter\_Ford + 1 |
| next |
| } |
|  |
| print("Updating Ford Data Frame") |
| cat("Updating Ford Data Frame\n",file = twitter\_log) |
|  |
| ford\_dataFrame = rbind(ford\_dataFrame,temp\_dataFrame) |
| securities\_toCapture[i,"HeadId"] = head(ford\_dataFrame,n=1)["id"] |
| securities\_toCapture[i,"TailId"] = tail(ford\_dataFrame,n=1)["id"] |
|  |
| if(nrow(ford\_dataFrame) %% 500 == 0){ |
| print("##### Dumping Ford Data Frame Checkpoint#####") |
| cat(paste("##### Dumping Ford Data Frame Checkpoint#",nrow(ford\_dataFrame) %/% 500, "#####\n"),file = twitter\_log) |
| write.xlsx(ford\_dataFrame,paste("Ford",nrow(ford\_dataFrame) %/% 500,".xlsx",sep = "")) |
| } |
| }, |
| 'Tesla'={ |
| if(putMeTOSleepCounter\_Tesla == 3){ |
| cat(paste("WARNING: No data fetched for last 3 tries for ",data$Security,", skipping it!!\n")) |
| cat(paste("WARNING: No data fetched for last 3 tries for ",data$Security,", skipping it !!\n"),file = twitter\_log) |
| next |
| } |
|  |
| cat(paste("\*\*\*\*\*\*\*\*\* Capturing Past Data for: ",data$Security," for tags: ",data$HashTags)," \*\*\*\*\*\*\*\*\*\n") |
| cat(paste("\*\*\*\*\*\*\*\*\* Capturing Past Data for: ",data$Security," for tags: ",data$HashTags)," \*\*\*\*\*\*\*\*\*\n",file = twitter\_log) |
|  |
| tweets = searchTwitter(toString(data$HashTags) , |
| n=100, lang="en", maxID = data$TailId) |
| temp\_dataFrame = twListToDF(tweets) |
|  |
| cat(paste("\*\*\* Fetched : ",nrow(temp\_dataFrame)," tweets for Security: ",data$Security)," \*\*\* \n") |
| cat(paste("\*\*\* Fetched : ",nrow(temp\_dataFrame)," tweets for Security: ",data$Security)," \*\*\* \n",file = twitter\_log) |
|  |
| if(nrow(temp\_dataFrame) == 1 ){ |
| print(" \*\* Insufficient tweets \*\* Skipping this iteration") |
| cat(" \*\* Insufficient tweets \*\* Skipping this iteration\n",file = twitter\_log) |
| putMeTOSleepCounter\_Tesla = putMeTOSleepCounter\_Tesla + 1 |
| next |
| } |
|  |
| print("Updating Tesla Data Frame") |
| cat("Updating Tesla Data Frame\n",file = twitter\_log) |
|  |
| tesla\_dataFrame = rbind(tesla\_dataFrame,temp\_dataFrame) |
| securities\_toCapture[i,"HeadId"] = head(tesla\_dataFrame,n=1)["id"] |
| securities\_toCapture[i,"TailId"] = tail(tesla\_dataFrame,n=1)["id"] |
|  |
|  |
| if(nrow(tesla\_dataFrame) %% 500 == 0){ |
| print("##### Dumping Tesla Data Frame Checkpoint#####") |
| cat(paste("##### Dumping Tesla Data Frame Checkpoint#",nrow(tesla\_dataFrame) %/% 500, "#####\n"),file = twitter\_log) |
| write.xlsx(tesla\_dataFrame,paste("Tesla",nrow(tesla\_dataFrame) %/% 500,".xlsx",sep = "")) |
| } |
| }, |
| 'General Motors'={ |
| if(putMeTOSleepCounter\_GM == 3){ |
| cat(paste("WARNING: No data fetched for last 3 tries for ",data$Security,", skipping it!!\n")) |
| cat(paste("WARNING: No data fetched for last 3 tries for ",data$Security,", skipping it !!\n"),file = twitter\_log) |
| next |
| } |
|  |
| cat(paste("\*\*\*\*\*\*\*\*\* Capturing Past Data for: ",data$Security," for tags: ",data$HashTags)," \*\*\*\*\*\*\*\*\*\n") |
| cat(paste("\*\*\*\*\*\*\*\*\* Capturing Past Data for: ",data$Security," for tags: ",data$HashTags)," \*\*\*\*\*\*\*\*\*\n",file = twitter\_log) |
|  |
| tweets = searchTwitter(toString(data$HashTags) , |
| n=100, lang="en", maxID = data$TailId) |
| temp\_dataFrame = twListToDF(tweets) |
|  |
| cat(paste("\*\*\* Fetched : ",nrow(temp\_dataFrame)," tweets for Security: ",data$Security)," \*\*\* \n") |
| cat(paste("\*\*\* Fetched : ",nrow(temp\_dataFrame)," tweets for Security: ",data$Security)," \*\*\* \n",file = twitter\_log) |
|  |
| if(nrow(temp\_dataFrame) == 1 ){ |
| print(" \*\* Insufficient tweets \*\* Skipping this iteration") |
| cat(" \*\* Insufficient tweets \*\* Skipping this iteration\n",file = twitter\_log) |
| putMeTOSleepCounter\_GM = putMeTOSleepCounter\_GM + 1 |
| next |
| } |
|  |
| print("Updating General Motors Data Frame") |
| cat("Updating General Motors Data Frame\n",file = twitter\_log) |
|  |
| generalmotors\_dataFrame = rbind(generalmotors\_dataFrame,temp\_dataFrame) |
| securities\_toCapture[i,"HeadId"] = head(generalmotors\_dataFrame,n=1)["id"] |
| securities\_toCapture[i,"TailId"] = tail(generalmotors\_dataFrame,n=1)["id"] |
|  |
| if(nrow(generalmotors\_dataFrame) %% 500 == 0){ |
| print("##### Dumping GM Data Frame Checkpoint#####") |
| cat(paste("##### Dumping GM Data Frame Checkpoint#",nrow(generalmotors\_dataFrame) %/% 500, "#####\n"),file = twitter\_log) |
| write.xlsx(generalmotors\_dataFrame,paste("GM",nrow(generalmotors\_dataFrame) %/% 500,".xlsx",sep = "")) |
| } |
| }, |
| { |
| print('Invalid Security') |
| cat("!!!!!!!!!!!Invalid Security!!!!!!!!\n",file = twitter\_log) |
| } |
| ) |
| } |
| } |
|  |
|  |
| #Dump Data frames to Files |
| for(i in 1:nrow(securities\_toCapture)) { |
| data <- securities\_toCapture[i,] |
|  |
| switch(as.character(data$Security), |
| 'Ford'={ |
| print("##### Dumping Ford Data Frame #####") |
| cat("##### Dumping Ford Data Frame #####\n",file = twitter\_log) |
| write.xlsx(ford\_dataFrame,"Ford\_Final.xlsx") |
| #myjson = toJSON(ford\_dataFrame, pretty = TRUE) |
| #save(myjson, file=paste("Ford",Sys.Time(),".xlsx",sep = "")) |
| }, |
| 'Tesla'={ |
| print("##### Dumping Tesla Data Frame #####") |
| cat("##### Dumping Tesla Data Frame #####\n",file = twitter\_log) |
| write.xlsx(tesla\_dataFrame,"Tesla\_Final.xlsx") |
| }, |
| 'General Motors'={ |
| print("##### Dumping GM Data Frame #####") |
| cat("##### Dumping GM Data Frame #####\n",file = twitter\_log) |
| write.xlsx(generalmotors\_dataFrame,"GM\_Final.xlsx") |
| }, |
| { |
| print('Invalid Security') |
| } |
| ) |
| # do stuff with row |
| # print(row$hastags) |
| } |
|  |
| #Close the log file |
| close(twitter\_log) |

## Transformation

|  |
| --- |
|  |
|  |
| `Subj` <- |
| function(x) |
| { |
| if(has.Subj(x)) |
| return(x[,grep('Subjectivity',colnames(x),ignore.case=TRUE)]) |
| stop('subscript out of bounds: no column name containing "Score"') |
| } |
|  |
| `has.Subj` <- |
| function(x,which=FALSE) |
| { |
| loc <- grep('Subjectivity',colnames(x),ignore.case=TRUE) |
| if(!identical(loc,integer(0))) |
| return(ifelse(which,loc,TRUE)) |
| ifelse(which,loc,FALSE) |
| } |
|  |
|  |
|  |
| `Polarity` <- |
| function(x) |
| { |
| if(has.Polarity(x)) |
| return(x[,grep('Polarity',colnames(x),ignore.case=TRUE)]) |
| stop('subscript out of bounds: no column name containing "Polarity"') |
| } |
|  |
| `has.Polarity` <- |
| function(x,which=FALSE) |
| { |
| loc <- grep('Polarity',colnames(x),ignore.case=TRUE) |
| if(!identical(loc,integer(0))) |
| return(ifelse(which,loc,TRUE)) |
| ifelse(which,loc,FALSE) |
| } |
|  |
|  |
|  |
| `Pos\_Refs\_Per\_Ref` <- |
| function(x) |
| { |
| if(has.Pos\_Refs\_Per\_Ref(x)) |
| return(x[,grep('Pos\_Refs\_Per\_Ref',colnames(x),ignore.case=TRUE)]) |
| stop('subscript out of bounds: no column name containing "Pos\_Refs\_Per\_Ref"') |
| } |
| `has.Pos\_Refs\_Per\_Ref` <- |
| function(x,which=FALSE) |
| { |
| loc <- grep(‘Pos\_Refs\_Per\_Ref’,colnames(x),ignore.case=TRUE) |
| if(!identical(loc,integer(0))) |
| return(ifelse(which,loc,TRUE)) |
| ifelse(which,loc,FALSE) |
| } |
|  |
| `Neg\_Refs\_Per\_Ref` <- |
| function(x) |
| { |
| if(has.Neg\_Refs\_Per\_Ref(x)) |
| return(x[,grep('Neg\_Refs\_Per\_Ref',colnames(x),ignore.case=TRUE)]) |
| stop('subscript out of bounds: no column name containing "Neg\_Refs\_Per\_Ref"') |
| } |
|  |
| `has.Neg\_Refs\_Per\_Ref` <- |
| function(x,which=FALSE) |
| { |
| loc <- grep('Neg\_Refs\_Per\_Ref',colnames(x),ignore.case=TRUE) |
| if(!identical(loc,integer(0))) |
| return(ifelse(which,loc,TRUE)) |
| ifelse(which,loc,FALSE) |
| } |
|  |
| `Senti\_Diffs\_Per\_Ref` <- |
| function(x) |
| { |
| if(has.Senti\_Diffs\_Per\_Ref(x)) |
| return(x[,grep('Senti\_Diffs\_Per\_Ref',colnames(x),ignore.case=TRUE)]) |
| stop('subscript out of bounds: no column name containing "Senti\_Diffs\_Per\_Ref"') |
| } |
|  |
| `has.Senti\_Diffs\_Per\_Ref` <- |
| function(x,which=FALSE) |
| { |
| loc <- grep('Senti\_Diffs\_Per\_Ref',colnames(x),ignore.case=TRUE) |
| if(!identical(loc,integer(0))) |
| return(ifelse(which,loc,TRUE)) |
| ifelse(which,loc,FALSE) |
| } |
|  |
| `SentVol` <- |
| function(x, prefix = "SEMNT") |
| { |
| name = paste(prefix, "vol", sep=".") |
|  |
| if(has.SentVol(x, prefix = prefix)) |
| return(x[,grep(name,colnames(x),ignore.case=TRUE)]) |
| stop(paste('subscript out of bounds: no column name containing"', name, '"')) |
| } |
|  |
| `has.SentVol` <- |
| function(x,which=FALSE, prefix = "SEMNT") |
| { |
| name = paste(prefix, "vol", sep=".") |
| loc <- grep(name,colnames(x),ignore.case=TRUE) |
| if(!identical(loc,integer(0))) |
| return(ifelse(which,loc,TRUE)) |
| ifelse(which,loc,FALSE) |
| } |
|  |

## Add Sentiment columns

|  |
| --- |
|  |
| addSubjectivity <- newTA(Subj, col=1) |
| addPolarity <- newTA(Polarity, col=4) |
| addPos\_Refs\_Per\_Ref <- newTA(Pos\_Refs\_Per\_Ref, col=3) |
| addNeg\_Refs\_Per\_Ref <- newTA(Neg\_Refs\_Per\_Ref, col=2) |
| addSenti\_Diffs\_Per\_Ref <- newTA(Senti\_Diffs\_Per\_Ref, col=5) |
| addSentVol <- newTA(SentVol, col=6) |
|  |
| `addSentiment` <- |
| function(fieldname, on = NA){ |
| lchob <- quantmod:::get.current.chob() |
| x <- as.matrix(lchob@xdata) |
|  |
|  |
|  |
| Volumes = NULL |
| if(missing(fieldname)){ |
| Volumes <- Polarity(x) |
| fieldname = "Polarity" |
| }else{ |
| Volumes <- x[,fieldname] |
| } |
|  |
|  |
| max.vol <- max(Volumes, na.rm = TRUE) |
|  |
| bar.col <- ifelse(Volumes > 0, lchob@colors$up.col, lchob@colors$dn.col) |
|  |
| border.col <- ifelse(is.null(lchob@colors$border), bar.col, |
| lchob@colors$border) |
| bar.col <- bar.col[lchob@xsubset] |
| chobTA <- new("chobTA") |
|  |
| if (any(is.na(on))) { |
| chobTA@new <- TRUE |
| } |
| else { |
| chobTA@new <- FALSE |
| chobTA@on <- on |
| } |
|  |
| chobTA@TA.values <- (Volumes)[lchob@xsubset] |
| chobTA@name <- "chartSent" |
| chobTA@call <- match.call() |
| chobTA@params <- list(xrange = lchob@xrange, colors = lchob@colors, |
| color.vol = lchob@color.vol, multi.col = lchob@multi.col, |
| spacing = lchob@spacing, width = lchob@width, bp = lchob@bp, |
| x.labels = lchob@x.labels, log.scale = FALSE, |
| bar.col = bar.col, border.col = border.col, time.scale = lchob@time.scale, vol.scale=list(1, fieldname)) |
| chobTA@params$thin <- ifelse(lchob@type %in% c("bars", "matchsticks"), |
| TRUE, FALSE) |
| if (is.null(sys.call(-1))) { |
| TA <- lchob@passed.args$TA |
| lchob@passed.args$TA <- c(TA, chobTA) |
| lchob@windows <- lchob@windows + ifelse(chobTA@new, 1, |
| 0) |
| FUN <- quantmod:::chartSeries.chob |
| do.call("FUN", list(lchob)) |
| invisible(chobTA) |
| } |
| else { |
| return(chobTA) |
| } |
|  |
| } |
|  |
|  |
| addSentimentVo <- function(fieldname, color = "blue"){ |
| lchob <- quantmod:::get.current.chob() |
| x <- as.matrix(lchob@xdata) |
|  |
| Volumes = NULL |
| if(missing(fieldname)){ |
| Volumes <- SentVol(x) |
| }else{ |
| Volumes <- x[,fieldname] |
| } |
|  |
| max.vol <- max(Volumes, na.rm = TRUE) |
|  |
|  |
| bar.col <- ifelse(Volumes > 0, color, "red") |
|  |
| border.col <- ifelse(is.null(lchob@colors$border), bar.col, |
| lchob@colors$border) |
|  |
| bar.col <- color |
|  |
| chobTA <- new("chobTA") |
| chobTA@new <- TRUE |
| chobTA@TA.values <- (Volumes)[lchob@xsubset] |
| chobTA@name <- "chartVo" |
| chobTA@call <- match.call() |
| chobTA@params <- list(xrange = lchob@xrange, colors = lchob@colors, |
| color.vol = TRUE, multi.col = lchob@multi.col, |
| spacing = lchob@spacing, width = lchob@width, bp = lchob@bp, |
| x.labels = lchob@x.labels, log.scale = FALSE, |
| bar.col = bar.col, border.col = border.col, time.scale = lchob@time.scale, vol.scale=list(1, "Text Documents")) |
| chobTA@params$thin <- ifelse(lchob@type %in% c("bars", "matchsticks"), |
| TRUE, FALSE) |
| if (is.null(sys.call(-1))) { |
| TA <- lchob@passed.args$TA |
| lchob@passed.args$TA <- c(TA, chobTA) |
| lchob@windows <- lchob@windows + ifelse(chobTA@new, 1, |
| 0) |
| FUN <- quantmod:::chartSeries.chob |
| do.call("FUN", list(lchob)) |
|  |
| invisible(chobTA) |
| } |
| else { |
| return(chobTA) |
| } |
|  |
| } |
|  |
| `chartSent` <- |
| function(x) { |
| if(class(x) != "chobTA") stop("chartSentiment requires a suitable chobTA object") |
| Volumes <- x@TA.values |
|  |
| spacing <- x@params$spacing |
| width <- x@params$width |
|  |
| x.range <- x@params$xrange |
| x.range <- seq(x.range[1],x.range[2]\*spacing) |
|  |
|  |
| color.vol <- x@params$color.vol |
| log.scale <- ifelse(x@params$log.scale,"y","") |
|  |
| vol.scale <- x@params$vol.scale |
|  |
| if(x@new) { |
| plot.new() |
| plot.window(xlim=c(1, x@params$xrange[2] \* spacing), |
| ylim=c(min(Volumes,na.rm=TRUE),max(Volumes,na.rm=TRUE)), |
| log=log.scale) |
| coords <- par('usr') |
| rect(coords[1],coords[3],coords[2],coords[4],col=x@params$colors$area) |
| abline(h=axTicks(2), col=x@params$colors$grid.col, lty='dotted') |
| } |
|  |
| x.pos <- 1 + spacing \* (1:length(Volumes) - 1) |
|  |
|  |
| bar.col <- x@params$bar.col |
| border.col <- x@params$border.col |
|  |
| if(x@params$thin) { |
|  |
| segments(x.pos,0,x.pos,Volumes,col=bar.col) |
| } else { |
| rect(x.pos-spacing/3,0,x.pos+spacing/3,Volumes, |
| col=bar.col,border=border.col) |
| } |
| legend.text <- list(list( |
| legend=c(paste("Sentiment Score(",vol.scale[[2]],"):",sep=''),format(last(Volumes)\*vol.scale[[1]],big.mark=',')), |
| text.col=c(x@params$colors$fg.col, last(bar.col)) |
| )) |
| legend("topleft", |
| legend=c(paste("Sentiment Score(",vol.scale[[2]],"):",sep=''),format(last(Volumes)\*vol.scale[[1]],big.mark=',')), |
| text.col=c(x@params$colors$fg.col, last(bar.col)), bty="n", y.inter=0.95) |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| axis(2) |
| box(col=x@params$colors$fg.col) |
| invisible(vector('list',2)) |
| } # }}} |
|  |

## Trend Visualization

|  |
| --- |
| chartSentiment <- |
| function (xts, prefix = "SEMNT", |
| sentname = paste(prefix,"polarity", sep = "."), |
| volname = paste(prefix,"vol", sep = "."), |
| volcolor="blue", |
| na.fill = TRUE, |
| omit.na.leading = TRUE, |
| type = "line", |
| theme='white', |
| name, |
| TA = "", postTA = "", ...) |
| { |
|  |
| counter = 1 |
|  |
| if(omit.na.leading){ |
| firstnona <- min(which(!is.na(xts[,sentname]))) |
| xts <- xts[firstnona:NROW(xts),] |
| } |
|  |
| if(na.fill){ |
| xts[is.na(xts)] <- 0 |
| } |
|  |
| if(missing(name)){ |
| name = "" |
| if(has.Cl(xts)) |
| name <- strsplit(colnames(Cl(xts)), "\\.", perl = TRUE)[[1]][1] |
| else |
| name <- strsplit(colnames(xts[,1]), "\\.", perl = TRUE)[[1]][1] |
|  |
| name <- paste("Sentiment", name) |
| } |
|  |
| if(sentname != ""){ |
| TA <- paste(TA, "addSentiment('", sentname, "');", sep = "") |
| counter = counter + 1 |
| } |
|  |
| if(has.SentVol(xts)){ |
| TA <- paste(TA, "addSentimentVo('", volname, "');", sep = "") |
| counter = counter + 1 |
| } |
|  |
| if(has.Vo(xts)){ |
| TA <- paste(TA, "addVo();", sep = "") |
| counter = counter + 1 |
| } |
|  |
| if(all(has.Subj(xts), |
| has.Neg\_Refs\_Per\_Ref(xts), |
| has.Pos\_Refs\_Per\_Ref(xts), |
| has.Senti\_Diffs\_Per\_Ref(xts))){ |
| range <- range(na.omit(rbind(Pos\_Refs\_Per\_Ref(xts), |
| Neg\_Refs\_Per\_Ref(xts), |
| Senti\_Diffs\_Per\_Ref(xts), |
| Subj(xts)))) |
|  |
| addSubjectivityAll <- newTA(Subj, col=1, yrange = range) |
| assign("addSubjectivityAll", addSubjectivityAll, envir = .GlobalEnv) |
|  |
| TA <- paste(TA, "addSubjectivityAll();", sep = "") |
| counter = counter + 1 |
|  |
| TA <- paste(TA, "addPos\_Refs\_Per\_Ref(on=", counter, ");", sep = "") |
| TA <- paste(TA, "addNeg\_Refs\_Per\_Ref(on=", counter, ");", sep = "") |
| TA <- paste(TA, "addSenti\_Diffs\_Per\_Ref(on=", counter, ");", sep = "") |
| }else{ |
| if(has.Subj(xts)){ |
| TA <- paste(TA, "addSubjectivity();", sep = "")# |
| } |
|  |
| if(has.Pos\_Refs\_Per\_Ref(xts)){ |
| TA <- paste(TA, "addPos\_Refs\_Per\_Ref();", sep = "") |
| } |
|  |
| if(has.Neg\_Refs\_Per\_Ref(xts)){ |
| TA <- paste(TA, "addNeg\_Refs\_Per\_Ref();", sep = "") |
| } |
|  |
| if(has.Senti\_Diffs\_Per\_Ref(xts)){ |
| TA <- paste(TA, "addSenti\_Diffs\_Per\_Ref();", sep = "") |
| } |
| } |
|  |
| TA <- paste(TA, postTA, sep = ";") |
|  |
|  |
| chartSeries(xts, type = "line", theme = "white", |
| name = name, |
| TA = TA, ...) |
| } |

## Corpus

|  |
| --- |
| "getField" <- function(x, ...) UseMethod("getField", x) |
| "getField.VCorpus" <- |
| function(x, fieldname){ |
| field <- do.call("c",lapply(x, function(y) meta(y, fieldname))) |
| names(field) <- NULL |
| field |
| } |
|  |
|  |
| "setField" <- function(x, ...) UseMethod("setField", x) |
|  |
| "setField.VCorpus" <- |
| function(x, fieldname, value){ |
| for(i in 1:length(x)){ |
| meta(x[[i]], fieldname) <- value[i] |
| } |
| x |
| } |

## Scrapping Relevant data

|  |
| --- |
|  |
| getRelevant <- |
| function(td, items, boundaries, matches.only = FALSE, fieldname = "matches"){ |
| if(missing(boundaries)){ |
| boundaries = "\\." |
| } |
| itempatt <- paste("\\W(",paste("(", items, ")", collapse = "|", sep = ""),")\\W", sep = "") |
| matches <- gregexpr(itempatt, Content(td), ignore.case = TRUE, perl = TRUE, fixed = FALSE, useBytes = FALSE) |
| if(!matches.only){ |
| startend <- range(matches) |
| delimmatches <- gregexpr(boundaries, Content(td), ignore.case = TRUE, perl = TRUE, fixed = FALSE, useBytes = FALSE) |
| delimmatches <- as.integer(delimmatches[[1]]) |
| textlen <- nchar(Content(td)) |
| sentstart <- max(c(delimmatches[delimmatches < startend[1]], 0)) |
| sentend <- min(c(delimmatches[delimmatches > startend[2]], textlen)) |
| subtext <- substr(Content(td), sentstart+1, sentend) |
| Content(td) <- substr(Content(td), sentstart, sentend) |
| } |
| if(any(matches[[1]] == -1)){ |
| meta(td, fieldname) <- 0 |
| } |
| else{ |
| meta(td, fieldname) <- length(matches[[1]]) |
| } |
| getRelevant <- td |
| } |

## Meta XTS

|  |
| --- |
| metaXTS <- function(corpus, |
| fieldnames, |
| period = "days", |
| k=1, |
| aggFUN = mean, |
| symbol, |
| prefix = "SEMNT", |
| join = "inner", |
| na.omit = TRUE){ |
|  |
| if(missing(fieldnames)){ |
| fieldnames <- colnames(meta(corpus)) |
| } |
|  |
| idx <- do.call(c, lapply(corpus, meta, "datetimestamp")) |
| df <- data.frame(idx, meta(corpus)[, fieldnames]) |
|  |
| if(na.omit){ |
| df <- na.omit(df) |
| } |
|  |
| xts <- xts(df[, -1], order.by = df[, 1]) |
| volxts <- xts(rep(1,NROW(xts)), order.by = index(xts)) |
|  |
| ep <- endpoints(xts, on = period, k = k) |
|  |
| xts\_agg = NULL |
| if(length(ep) > 2){ |
| ldata <- lapply(1:NCOL(xts), function(x) period.apply(xts[,x], ep, aggFUN)) |
| ldata\_all <- do.call("cbind", ldata) |
| xts\_agg <- xts(ldata\_all, order.by = index(xts)[ep]) |
| volxts\_agg <- period.apply(volxts, ep, sum) |
| }else{ |
| #ep <- ep[ep > 0] |
|  |
| xts\_agg <- xts(t(as.data.frame(apply(xts, 2, aggFUN))), order.by = index(xts)[ep]) |
| volxts\_agg <- xts(sum(volxts), order.by = index(xts)[ep]) |
| } |
|  |
| colnames(xts\_agg) <- colnames(xts) |
| colnames(volxts\_agg) <- "vol" |
|  |
| scorevolxts\_agg <- cbind(xts\_agg, volxts\_agg) |
| colnames(scorevolxts\_agg) <- paste(prefix,colnames(scorevolxts\_agg), sep=".") |
| if(period == "days") |
| index(scorevolxts\_agg) <- as.Date(index(scorevolxts\_agg)) |
|  |
| if(!missing(symbol)){ |
| names <- colnames(symbol) |
| symbol <- to.period(symbol, period = period, k = k) |
| colnames(symbol) <- names |
| scorevolxts\_agg <- merge(symbol, scorevolxts\_agg, join = join) |
| } |
|  |
| return(scorevolxts\_agg) |
| } |
|  |

## Pre Process the Corpus

|  |
| --- |
| preprocessCorpus <- |
| function(corpus, control, verbose = FALSE){ |
|  |
| if(missing(control)){ |
| control = list( removePunctuation = list(), |
| removeNumbers = list(), |
| tolower = list(), |
| removeWords = list(stopwords("english")), |
| stripWhitespace = list(), |
| stemDocument = list()) |
|  |
| } |
|  |
| if(verbose) |
| cat("Starting Preprocessing...\n") |
|  |
| for(n in names(control)){ |
| if(verbose) |
| cat(n, " ...") |
| args <- control[[n]] |
| if(length(args) == 0){ |
| corpus <- eval(call("tm\_map", corpus, n)) |
| }else{ |
| corpus <- eval(call("tm\_map", corpus, n, unlist(args))) |
| } |
| if(verbose) |
| cat("Done\n") |
| } |
| Corpus |
| } |
|  |

## Score Function

|  |
| --- |
| `score` <- function(corpus, |
| control, |
| scoreFUNS){ |
|  |
| if(missing(control)){ |
| control = list( |
| tolower = TRUE, |
| removePunctuation = TRUE, |
| removeNumbers = TRUE, |
| removeWords = list(stopwords("english")), |
| stripWhitespace = TRUE, |
| stemDocument = TRUE, |
| minWordLength = 3, |
| weighting = weightTf) |
| } |
| if(missing(scoreFUNS)){ |
| scoreFUNS = list( |
| polarity = list(), |
| subjectivity = list(), |
| pos\_refs\_per\_ref = list(), |
| neg\_refs\_per\_ref = list(), |
| senti\_diffs\_per\_ref = list() |
| ) |
| } |
|  |
| tdm <- TermDocumentMatrix(corpus, control = control) |
|  |
|  |
| res <- list() |
| for(n in names(scoreFUNS)){ |
| args <- unlist(scoreFUNS[[n]]) |
| if(is.null(args)){ |
| res[[n]] <- eval(call(n, tdm)) |
| }else{ |
| res[[n]] <- eval(call(n, tdm, args)) |
| } |
| } |
|  |
| dfres <- as.data.frame(res) |
| meta <- meta(corpus) |
|  |
| for(n in colnames(dfres)) { |
| meta(corpus, n) <- dfres[, n] |
| } |
| Corpus |
| } |
|  |
| polarity <- function(x, positive, negative) UseMethod("polarity", x) |
|  |
| polarity.TermDocumentMatrix <- function(x, positive = posterms\_GI(), negative = negterms\_GI() ){ |
| pos <- tm\_term\_score(x, positive) |
| neg <- tm\_term\_score(x, negative) |
| all <- (pos+neg) |
| ifelse(all != 0, (pos-neg)/all, 0) |
| } |
|  |
| polarity.DocumentTermMatrix <- function(x, ...) polarity(t(x), ...) |
|  |
| subjectivity <- function(x, positive, negative) UseMethod("subjectivity", x) |
|  |
| subjectivity.TermDocumentMatrix <- function(x, positive = posterms\_GI(), negative = negterms\_GI() ){ |
| pos <- tm\_term\_score(x, positive) |
| neg <- tm\_term\_score(x, negative) |
| all <- col\_sums(x) |
| ifelse(all != 0, (pos+neg)/all, 0) |
| } |
|  |
| subjectivity.DocumentTermMatrix <- function(x, ...) subjectivity(t(x), ...) |
|  |
| pos\_refs\_per\_ref <- function(x, positive) UseMethod("pos\_refs\_per\_ref", x) |
|  |
| pos\_refs\_per\_ref.TermDocumentMatrix <- function(x, positive = posterms\_GI()){ |
| pos <- tm\_term\_score(x, positive) |
| all <- col\_sums(x) |
| ifelse(all != 0, (pos)/(all), 0) |
| } |
|  |
| pos\_refs\_per\_ref.DocumentTermMatrix <- function(x, ...) pos\_refs\_per\_ref(t(x), ...) |
|  |
| neg\_refs\_per\_ref <- function(x, negative) UseMethod("neg\_refs\_per\_ref", x) |
|  |
| neg\_refs\_per\_ref.TermDocumentMatrix <- function(x, negative = negterms\_GI()){ |
| neg <- tm\_term\_score(x, negative) |
| all <- col\_sums(x) |
| ifelse(all != 0, (neg)/(all), 0) |
| } |
|  |
| neg\_refs\_per\_ref.DocumentTermMatrix <- function(x, ...) neg\_refs\_per\_ref(t(x), ...) |
|  |
| senti\_diffs\_per\_ref <- function(x, positive, negative) UseMethod("senti\_diffs\_per\_ref", x) |
|  |
| senti\_diffs\_per\_ref.TermDocumentMatrix <- function(x, positive = posterms\_GI(), negative = negterms\_GI()){ |
| pos <- tm\_term\_score(x, positive) |
| neg <- tm\_term\_score(x, negative) |
| all <- col\_sums(x) |
| ifelse(all != 0, (pos-neg)/all, 0) |
| } |
|  |
| senti\_diffs\_per\_ref.DocumentTermMatrix <- function(x, ...) senti\_diffs\_per\_ref(t(x), ...) |
|  |
| posterms\_GI <- function(){ |
| data("dic\_gi") |
| data <- get("dic\_gi", pos=globalenv()) |
| data[["positive"]] |
| } |
|  |
| negterms\_GI <- function(){ |
| data("dic\_gi") |
| data <- get("dic\_gi", pos=globalenv()) |
| data[["negative"]] |

**}**