Sentiment Analysis And Stock Price Movement

Pavan Akula

May 8, 2017

Abstract

The goal of this project is to perform *Sentiment Analysis* of news articles about *Whole Foods Market* company. At the beginning of April 2017 news outlets started publishing news, investors of privately-owned grocery chain Albertsons were considering the *Whole Foods Market* takeover. Some articles favored acquisition while others described the weakness in sales and revenues of the company.

Introduction

As part of the project, I will be presenting multiple ways to perform Sentiment Analysis.

- Basic Sentiment Analysis
- Sentence Sentiment Analysis

There are many R packages available to perform *Sentiment Analysis*, for the scope of the project I will be using three packages `tidytext`, `sentimentr` and `syuzhet`. The main idea is to compare sentiment polarity scores based on `lexicon` and `package`. Finally, sentiment scores and stock prices will be combined to evaluate whether there is a linear relationship between `news sentiment` and `stock price movement`.

Method and Software Usage

The principles of tidy data provided by `Hadley Wickham` are followed throughout the process of cleaning and preparing the data for analysis. The software tool used for the project is `R`. *Basic sentiment analysis* is done using three lexicons `afinn`, `bing` and `nrc` from `tidytext` package. *Sentence Sentiment Analysis* and *Document Sentiment Analysis* is done using `sentimentr` package. Function from technical trading rules `TTR` package is used for calculating of `Moving Averages` and `Volatality` of the stock price. `MongoDB` NoSQL database is used for storing data.

Data Source

For sentiment analysis, all news articles are harvested manually by web scraping. Data harvested from each website is saved as the text file on the local drive and also loaded into `MongoDB` database for future usage. Using functions from `tm` package document `corpus` is created for processing. *Whole Foods Market* stock prices are imported from *Yahoo! finance* website using `quantmod` package.

Libraries used.

```
if (!require('syuzhet')) install.packages('syuzhet')          #Web scraping and text extraction
if (!require('plyr')) install.packages('plyr')                #Data frame and table functions
if (!require('dplyr')) install.packages('dplyr')              #Data frame and table functions
if (!require('stringr')) install.packages('stringr')          #String manuplilation functions
if (!require('tm')) install.packages('tm')                    #create document corpus and DocumentTermMatrix
if (!require('quantmod')) install.packages('quantmod')        #Get stock prrices
if (!require('TTR')) install.packages('TTR')                  #Calculate Moving Averages, RSI, Volatality
if (!require('tidyr')) install.packages('tidyr')              #Tidy data using spread() and gather() functions
if (!require('tidytext')) install.packages('tidytext')        #Word sentiment analysis
if (!require('sentimentr')) install.packages('sentimentr')    #Sentence sentiment analysis
if (!require('ggplot2')) install.packages('ggplot2')          #Graphics display
if (!require('gridExtra')) install.packages('gridExtra')      #Display graphs side by side
if (!require('wordcloud')) install.packages('wordcloud')      #Create word cloud
if (!require('RColorBrewer')) install.packages('RColorBrewer') #Get color palette
if (!require('knitr')) install.packages('knitr')              #Report display, table format
if (!require('quantmod')) install.packages('quantmod')        #Get stock prices
if (!require('jsonlite')) install.packages('jsonlite')        #converting data into JSON
if (!require('mongolite')) install.packages('mongolite')      #connecting to MongoDB
```

Load all articles and stock price data into `MongoDB` NoSQL database.

```
#Function to cleanup data
cleanData <- function(x){
  x <- gsub(":", "", x) # Replace junk with ("")
  x <- iconv(x, "latin1", "ASCII", sub=" ")
  x <- gsub("\\s+", " ", x) # Remove double spaces
  return(x)
}

properDate <- function(x){
  y <- str_sub(x, -4) #Get last 4 digits
  m <- substr(x,1,2)
  d <- paste0(y, "-", m, "-", substr(x,3,4), " 00:00:00")
  d <- as.Date(d)
  return(d)
}

#Create mongoDB connection to load articles
mongoCon <- mongo(collection = "articles", db = "finalProject")
query = '{}'
fields = ' {"_id" : 0}'
mongo.article <- mongoCon$find(query, fields)

#Articles does not exist in MongoDB insert them
if (nrow(mongo.article) < 1){
    #Change path
    rootDir = "D:/CUNY/607/FinalProject/Final/Articles/"
    pattern = "txt$"
    fileFolder <- rootDir
    fileList <- list.files(path = fileFolder, pattern = pattern, all.files = FALSE, full.names = TRUE, recursive = FALSE)

    for(f in fileList){

        #Read first line it contains source of the article
        article.text <- readLines(f)
        article.source <- article.text[1]

        #Get entire text from the file
        article.text <- f %>% get_text_as_string()

        #Cleanup text
        article.link <- cleanData(article.source)
        article.text <- cleanData(article.text)

        #remove source link from the article
        article.text <- str_replace_all(article.text, article.link, "")

        #Article file name
        article.fileName <- gsub(pattern = rootDir, replacement = "", f)

        #Article date
        pattern <- "[[:digit:]]+"
        article.date <- properDate(str_extract(article.fileName, pattern))

        artilce.df <- data.frame(article.fileName, article.source, article.date, article.text)
        colnames(artilce.df) <- gsub("\\.", "_", colnames(artilce.df))

        #Check if article already present in DB
        query = paste0('{"article_fileName": "', article.fileName,'" }')
        fields = ' {"_id" : 1, "article_source" : 1, "article_date" : 1, "article_text" : 1 }'
        mongo.article <- mongoCon$find(query, fields)

        if(nrow(mongo.article) == 0){
            #if file does not exist load it
            mongoCon$insert(artilce.df)
        } else{
              #Source changed update
                if (!grepl(cleanData(mongo.article$article_source), article.link, ignore.case = T)){

                  query = paste0('{"_id": { "$oid" : "', mongo.article$`_id`,'" } }')
                  update = paste0('{ "$set" : { "article_source" : "', article.source, '"} }')

                  a<-mongoCon$update(query, update)
                }
        }
    }

    #After loading data query MongoDB
    query = '{}'
    fields = ' {"_id" : 0}'
    mongo.article <- mongoCon$find(query, fields)

}
```

```
#Close connection
rm(mongoCon)

#Create mongoDB connection to load stock price
mongoCon <- mongo(collection = "stockPrice", db = "finalProject")

query = '{}'
fields = ' {"_id" : 0}'
mongo.stock.price <- mongoCon$find(query, fields)

#Data does not exist connect to Yahoo finance and get data
if (nrow(mongo.stock.price) < 1){
  #Obtain stock price, Whole Foods Market ticker is 'WFM'
  wfm.stock.data <- new.env()
  getSymbols('WFM',src='yahoo',env = wfm.stock.data)

  #Convert to data frame
  wfm.stock.data <-as.data.frame(wfm.stock.data$WFM)
  wfm.stock.data$tradingDay <- as.Date(row.names(wfm.stock.data))
  rownames(wfm.stock.data) <- NULL

  #Add average stock price per day
  wfm.stock.data <- wfm.stock.data %>% mutate(avg_stockprice = (WFM.Open + WFM.High + WFM.Low) / 3)

  #Calculate moving averages
  wfm.stock.data$ma_10 <- round(SMA(wfm.stock.data$WFM.Close, n = 10),2)
  wfm.stock.data$ma_50 <- round(SMA(wfm.stock.data$WFM.Close, n = 50),2)

  #Calculate exponential moving averages
  wfm.stock.data$ema_10 <- round(EMA(wfm.stock.data$WFM.Close, n = 10),2)
  wfm.stock.data$ema_50 <- round(EMA(wfm.stock.data$WFM.Close, n = 50),2)

  #Volatility data
  Vol <- data.frame(Vol = volatility(OHLC(wfm.stock.data), n = 10, N = 260, mean0 = FALSE, calc="garman"))
  wfm.stock.data <- cbind(wfm.stock.data,vol_garman=Vol$Vol)

  mongoCon$insert(wfm.stock.data)

  #After loading data get MongoDB data
  query = '{}'
  fields = ' {"_id" : 0}'
  mongo.stock.price <- mongoCon$find(query, fields)
}

#Close connection
rm(mongoCon)
```

Corpus, Term Document Matrix(TDM) and Sentences Generation

Text inside each articles is broken into words using `TermDocumentMatrix` function from `tm` package. Also, text is seperated into sentences using `unnest_tokens` function from `tidytext` package.

```
#Read files and create Term Document Matrix
filesToTDM <- function(x){
  #Generate corpus for filelist
  wfm.article.corpus <- Corpus(VectorSource(x$article_text), readerControl = list(reader = readPlain, language = "en_US", lo
ad = TRUE))

  #Clean up the corpus
  wfm.article.corpus <- tm_map(wfm.article.corpus, removePunctuation)
  wfm.article.corpus <- tm_map(wfm.article.corpus, removeNumbers)
  wfm.article.corpus <- tm_map(wfm.article.corpus, stripWhitespace)
  wfm.article.corpus <- tm_map(wfm.article.corpus, content_transformer(tolower))
  wfm.article.corpus <- tm_map(wfm.article.corpus, PlainTextDocument)

  #Generate Document Term Matrix
  wfm.tdm <- TermDocumentMatrix(wfm.article.corpus)
  tdmOutput <- list(tdm = wfm.tdm, articleFile = x$article_fileName)
  return(tdmOutput)
}

#Function to cleanup data
cleanData <- function(x){
  x <- ifelse(grepl("http", x), NA, x)
  x <- gsub("[[:punct:]]", "", x) # Replace junk with ("")
  x <- iconv(x, "latin1", "ASCII", sub=" ")
  x <- gsub("\\s+", " ", x) # Remove double spaces
  return(x)
}

properDate <- function(x){
  y <- str_sub(x, -4) #Get last 4 digits
  m <- substr(x,1,2)
  d <- paste0(y, "-", m, "-", substr(x,3,4), " 00:00:00")
  d <- as.Date(d)
  return(d)
}

getSentenceFromText <- function(x){
  #Create empty table to store sentences from articles
  articles.sentence.df <- data.frame(sentence<-NA, filename<-NA, stringsAsFactors = F)
  colnames(articles.sentence.df) <- c("sentence", "fileName")

  for(i in 1:nrow(x)){

    #Get entire text from the file
    fileText <- x$article_text[i]

    #Tibble is used as unnest_tokens expects data in table format
    textToSentence <- tibble(text = fileText) %>% unnest_tokens(sentence, text, token = "sentences") %>% as.data.frame(strin
gsAsFactors = F)
    textToSentence$fileName <- x$article_fileName[i]
    textToSentence$sentence <- cleanData(textToSentence$sentence)

    articles.sentence.df <- rbind(articles.sentence.df, textToSentence)
  }
  articles.sentence.df <- na.omit(articles.sentence.df)
  rownames(articles.sentence.df) <- NULL
  return(articles.sentence.df)
}

#Convert files to Document Term Matrix
wfm.articles.tdm <- filesToTDM(mongo.article)
wfm.articles.matrix <- data.matrix(wfm.articles.tdm[["tdm"]])

#Convert matrix to dataframe
wfm.articles.df <- as.data.frame(wfm.articles.matrix, stringsAsFactors = F)

wfm.articles.filename <- wfm.articles.tdm[["articleFile"]]

#Bind filenames to columns
colnames(wfm.articles.df) <- wfm.articles.filename

#Clean data get words from rownames
wfm.articles.df$fileWords <- cleanData(rownames(wfm.articles.df))
rownames(wfm.articles.df) <- NULL

#Transpose columns to rows
wfm.tidy.data <- gather(wfm.articles.df, fileWords)
colnames(wfm.tidy.data) <- c("fileWord","fileName","wordCount")

#Ignore rows with NA values and wordCount less than 1. Means word does not exist in the article
wfm.tidy.data <- na.omit(wfm.tidy.data)
wfm.tidy.data <- wfm.tidy.data[wfm.tidy.data$wordCount>0, ]
rownames(wfm.tidy.data) <- NULL
```

```
#Get stop words from 'tidytext' package and remove from data frame
lexStopWords <- stop_words
wfm.tidy.data <- wfm.tidy.data %>%
                    anti_join(lexStopWords  , by = c("fileWord" = "word")) %>%
                    filter(!fileWord  %in% c("april", "byteresa", "cfra", "jana","npd", "shopjana","wfm","ihor","amazoncom",
"anayahooyptryahoo","bloomberg","carolinabased","cincinnatibased","cincinnati", "monday", "month","dusaniwsky"))

#Get sentences from each text file
#This data frame will used for sentence analysis
wfm.articles.sentence.df <- getSentenceFromText(mongo.article)

#Attach date
pattern <- "[[:digit:]]+"
wfm.articles.sentence.df$articleDate <- properDate(str_extract(wfm.articles.sentence.df$fileName, pattern))
wfm.tidy.data$articleDate <- properDate(str_extract(wfm.tidy.data$fileName, pattern))
```

Basic Sentiment Analysis

This is a simple sentiment analysis focusing only at the word level. The `tidytext` package contains three sentiment lexicons. All three lexicons are based on unigrams and contain many English words.

- `bing` by Bing Liu, classifies words from each sentence into positive and negative sentiment.
- `nrc` by Saif Mohammad and Peter Turney, classifies words into emotions like positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust.
- `afinn` by Finn Arup Nielsen, assigns words with a score that runs between -5 and 5, with negative scores indicating negative sentiment and positive scores indicating positive sentiment.

`bing` lexicon.

```
#Get words from bing lexicon
bingLexWord <- get_sentiments("bing")

#Apply sentiment to words
wfm.bing <- wfm.tidy.data %>% inner_join(bingLexWord, by = c("fileWord" = "word"))
```

`nrc` lexicon.

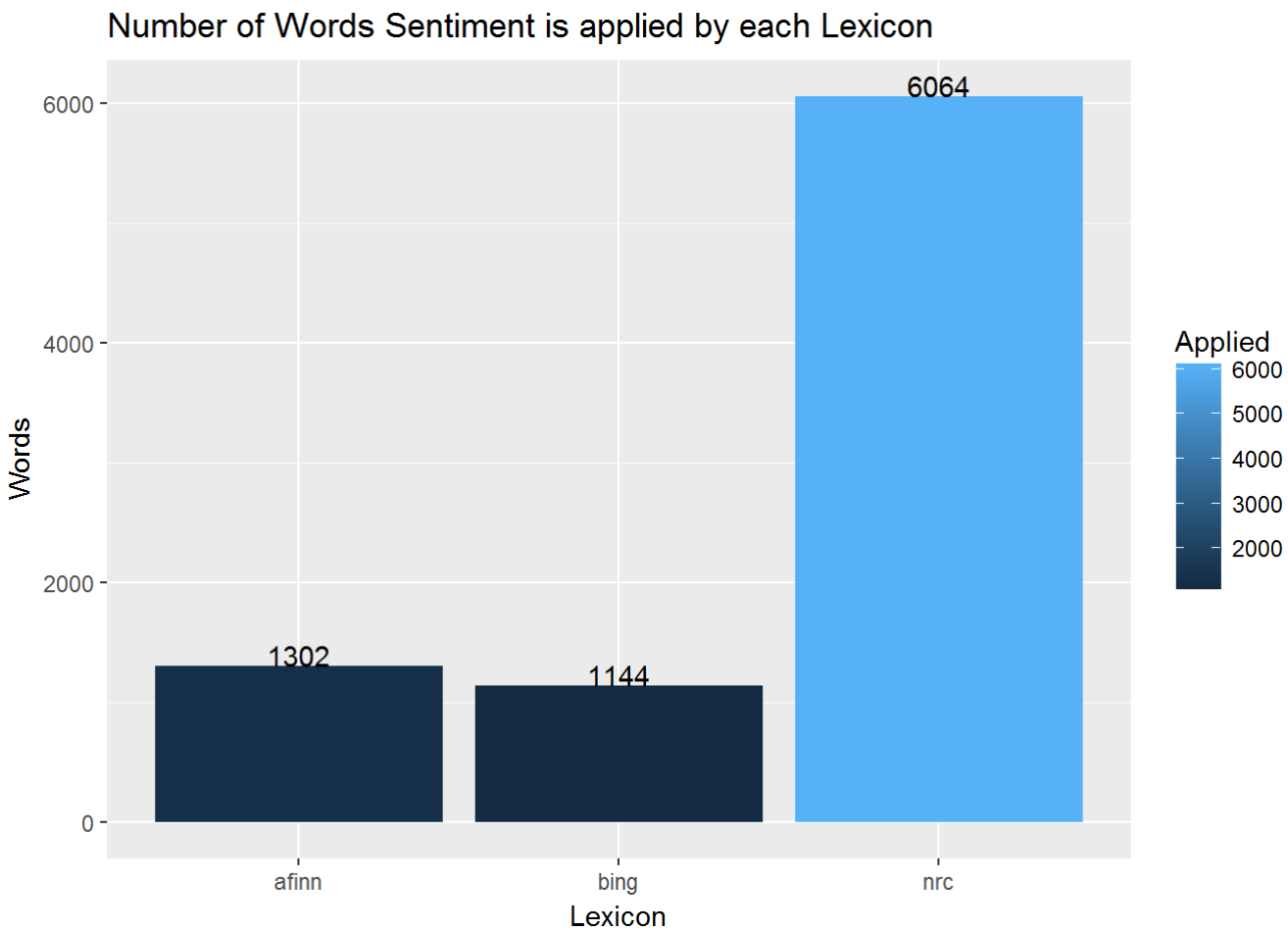
```
#Get words from bing lexicon
nrcLexWord <- get_sentiments("nrc")

#Apply sentiment to words
wfm.nrc <- wfm.tidy.data %>% inner_join(nrcLexWord, by = c("fileWord" = "word"))
```

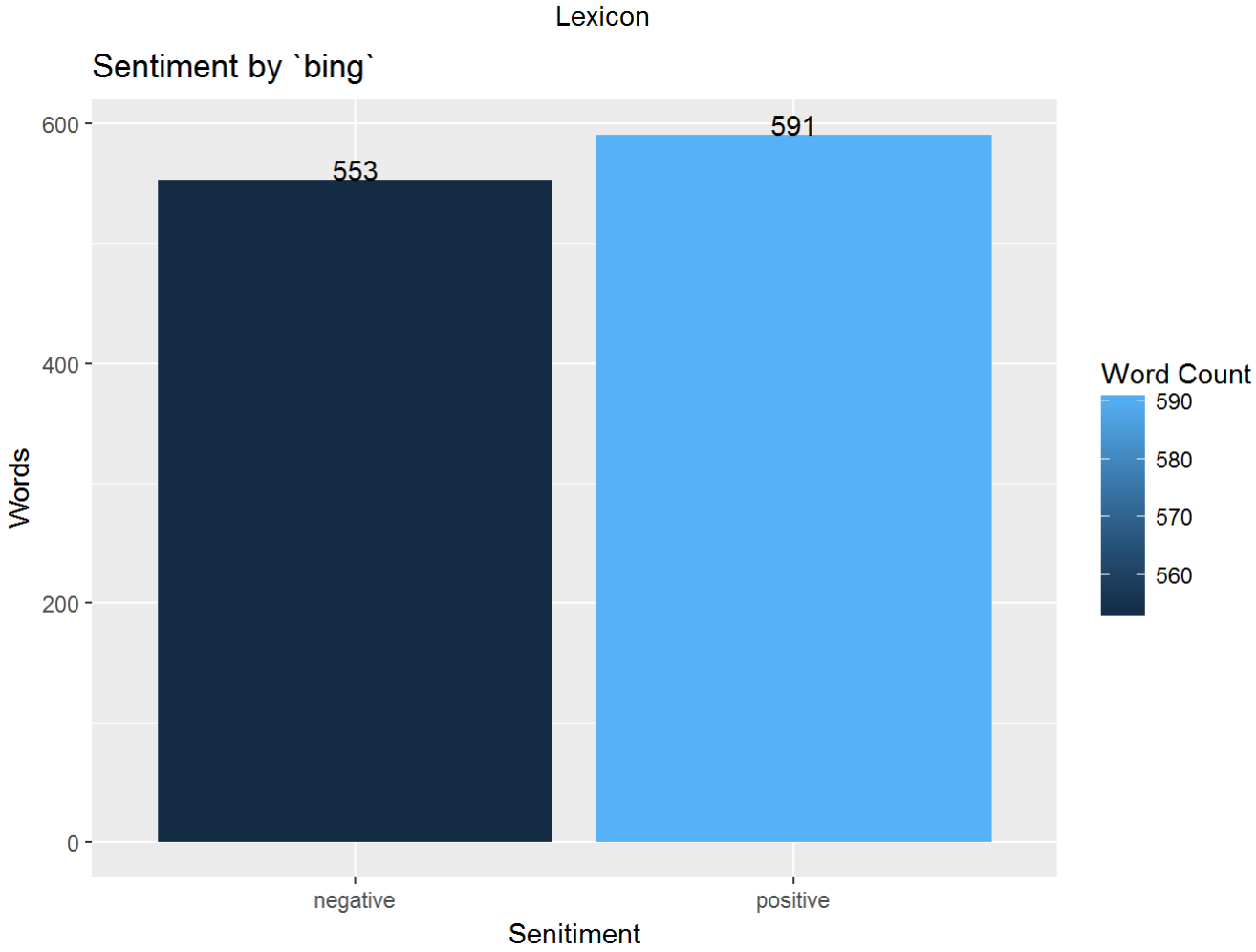
`afinn` lexicon.

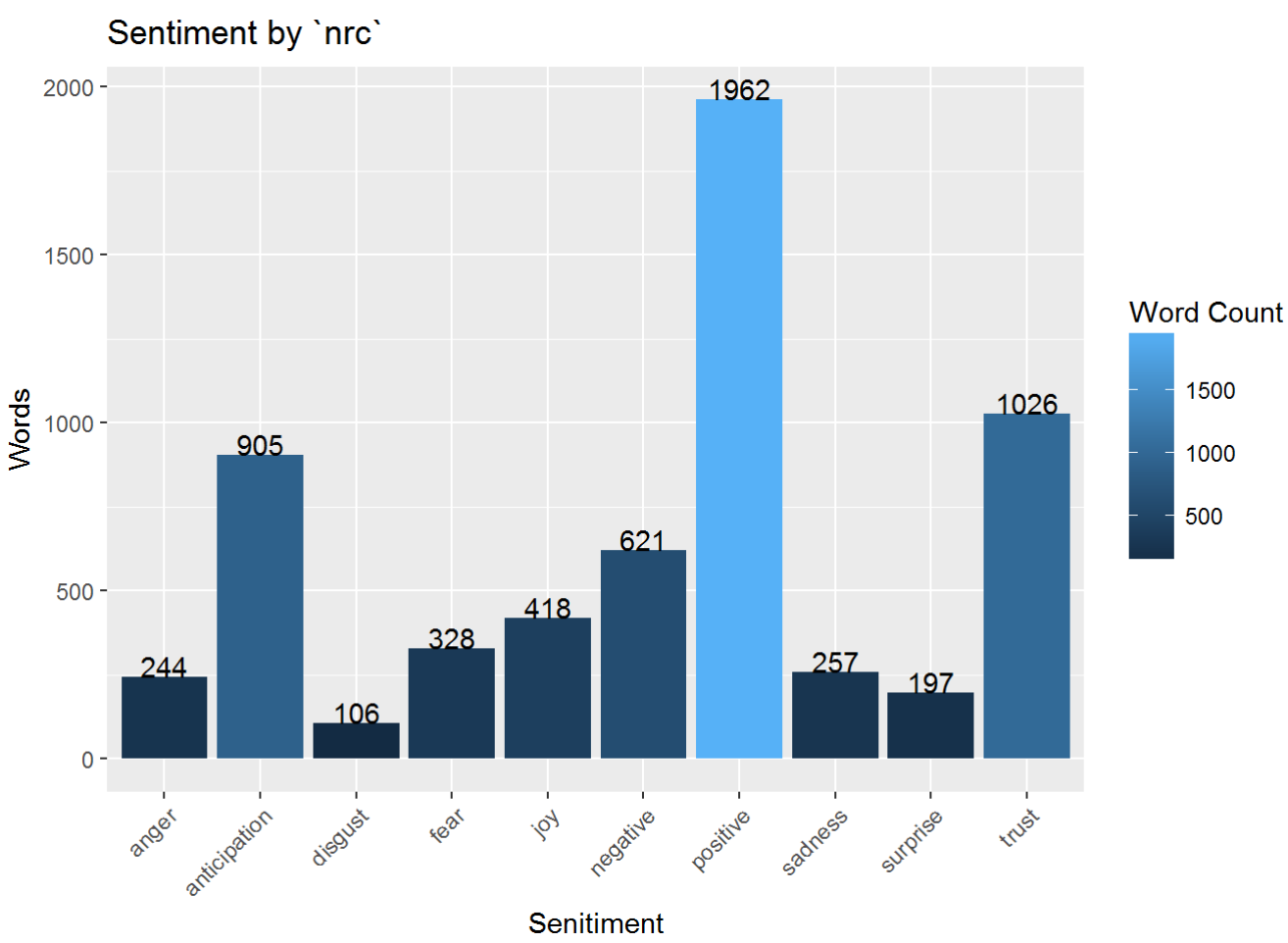
```
#Get words from bing lexicon
afinnLexWord <- get_sentiments("afinn")

#Apply sentiment to words
wfm.afinn <- wfm.tidy.data %>% inner_join(afinnLexWord, by = c("fileWord" = "word"))
```

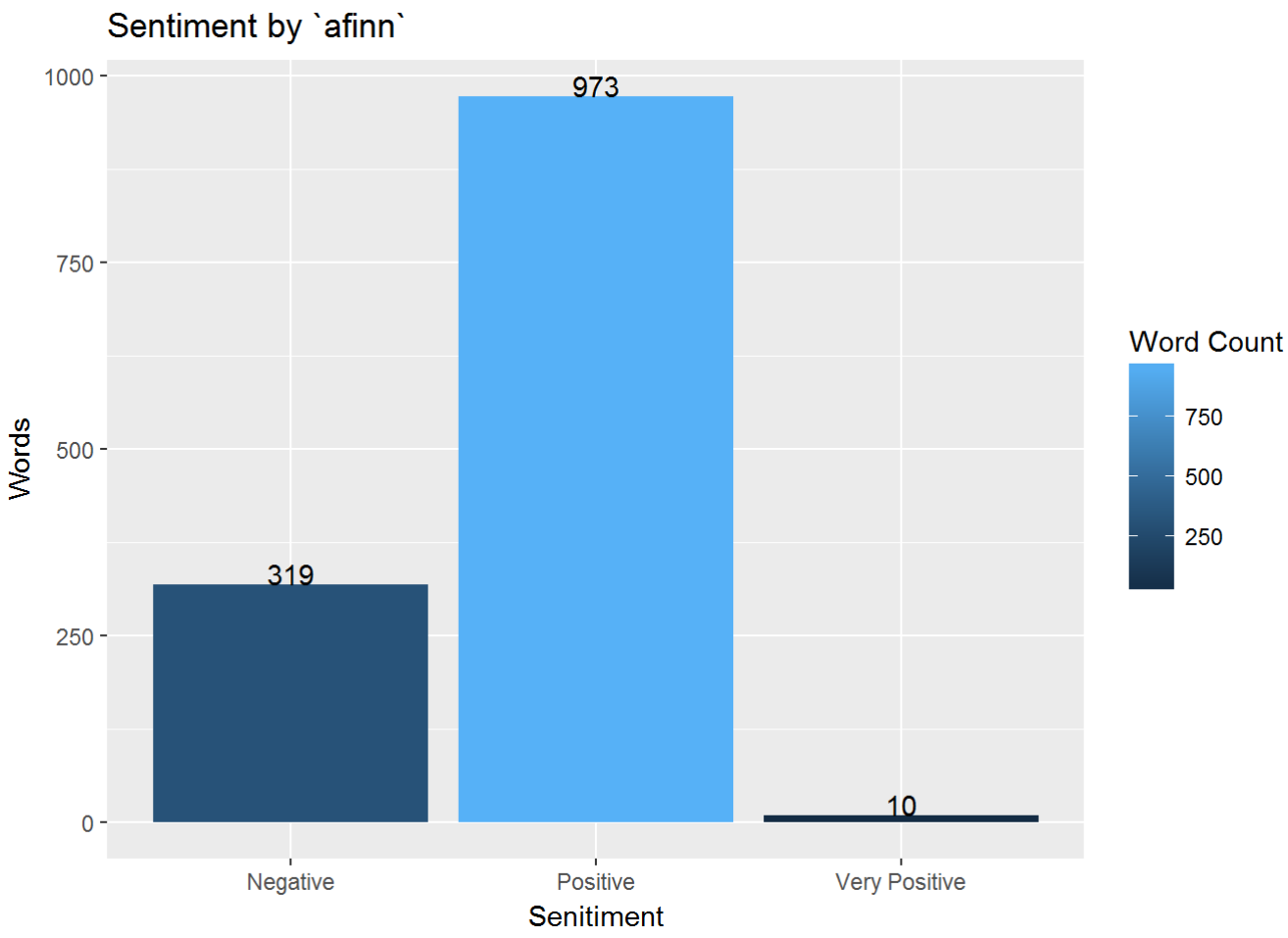
Comparing Lexicons

Following graphs show *Sentiment score* assigned by each lexicon to individual words in all documents in the corpus.


Number of Words Sentiment is applied by each Lexicon
Words
Lexicon
afinn
bing
nrc
1302
1144
6064
Applied

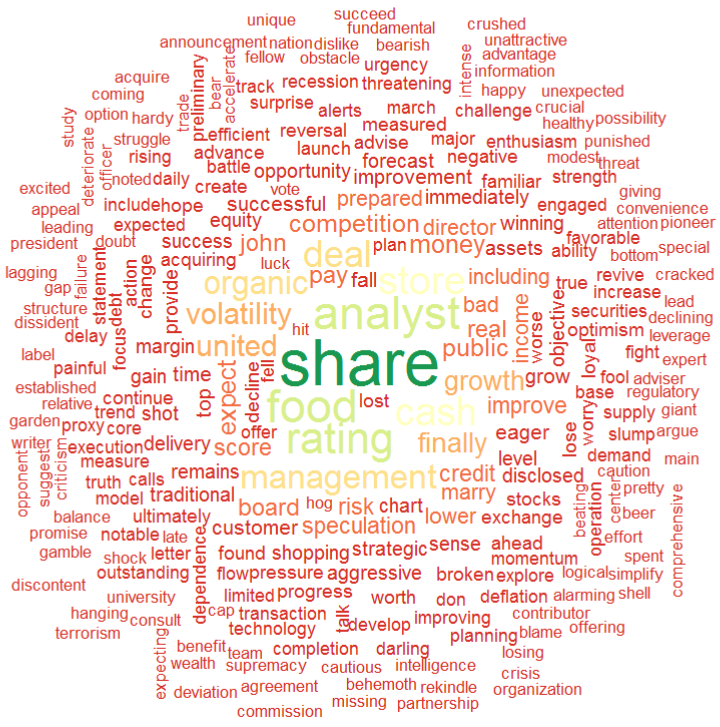

Sentiment by `bing`
Words
Senitiment
negative
positive
553
591
Word Count


Sentiment by `nrc`
Words
Senitiment
anger
anticipation
disgust
fear
joy
negative
positive
sadness
surprise
trust
244
905
106
328
418
621
1962
257
197
1026
Word Count



Word Cloud Using `bing` Lexicon

Word Cloud Using `nrc` Lexicon



Sentence Sentiment Analysis

This method of analysis focuses on entire sentence or phrase and attaches sentiment score. An important feature is it preserves the order of words and valuable information. It computes sentiment based on how words compose the meaning of longer sentences or phrases. Two packages `sentimentr` and `syuzhet` are used to demonstrate the process. Both packages offer many features that can be customized based on the context and needs of a particular data set. For the scope of this project, I will be using `sentiment` function from `sentimentr` package and `get_sentiment` function from `syuzhet` package.

Sentiment score using `sentiment` function from `sentimentr` package. This package uses four lexicons `huliu`, `jockers`, `nrc` and `sentiword`.

```
#polarity of the statement using all four lexicons
wfm.sentimentr.polarity <-
  sentiment(text.var = wfm.articles.sentence.df$sentence, polarity_dt = lexicon::hash_sentiment_huliu, question.weight = 0)
%>%
  inner_join(sentiment(text.var = wfm.articles.sentence.df$sentence, polarity_dt = lexicon::hash_sentiment_jockers, questio
n.weight = 0), by = "element_id") %>%
  select(element_id, huliu = sentiment.x, jockers = sentiment.y) %>%
inner_join(sentiment(text.var = wfm.articles.sentence.df$sentence, polarity_dt = lexicon::hash_sentiment_nrc, question.weigh
t = 0), by = "element_id") %>%
  select(element_id, huliu, jockers, nrc=sentiment) %>%
inner_join(sentiment(text.var = wfm.articles.sentence.df$sentence, polarity_dt = lexicon::hash_sentiment_sentiword, questio
n.weight = 0), by = "element_id") %>%
  select(element_id, huliu, jockers, nrc, sentiword=sentiment)

#Map scores back to document and sentence
wfm.sentimentr.polarity <- cbind(wfm.articles.sentence.df[wfm.sentimentr.polarity$element_id, ], wfm.sentimentr.polarity)
```

Following table shows varying `Polarity Score` based on `lexicon` from `sentimentr` package. Sentence shown in row number `28`, all four lexicons gave the negative score. It actually matches the sentiment, the author is trying to express, why `whole foods market` profit margin is low. However, lines `32` and `33` have mixed sentiment. For line `32` lexicon `huliu` gave low negative score and other lexicons gave a positive score. On the other hand for line `33` lexicon `jockers` gave low negative polarity score, lexicon `nrc` gave neutral score, `huliu` and `sentiword` gave positive score. In other words, `Sentiment Score` varies if `mixed sentiment` is present in the sentence.

```
#Display data in table format
wfm.sentimentr.polarity %>%
  filter(fileName  == "WFM04122017a.txt") %>%
  mutate(rn = row_number()) %>%
  filter(rn %in% c(28,32,33)) %>%
  select (rn, sentence, huliu, jockers, nrc, sentiword) %>%
  kable(digits = 8, col.names = c("Row", "Sentence", "huliu", "jockers", "nrc", "sentiword"), format='pandoc', caption = "Di
fference in Polarity Scores based on Lexicon - `sentimentr` package")
```

Difference in Polarity Scores based on Lexicon - `sentimentr` package

Row	Sentence	huliu	jockers	nrc	sentiword
28	whole foods notoriously high prices put it at risk of margin and food price deflation pressures he said	-0.8485281	-0.5114739	-0.6599663	-0.0677644
32	ryan english a portfolio manager at sycamore township investment advisory firm foster motley said kroger could be a suitor but it s pretty expensive	-0.2041242	0.2020829	0.5062279	0.1837117
33	whole foods could also refocus and become a more formidable competitor to kroger	0.4992302	-0.2995381	0.0000000	0.5130977

Differences are shown using box plot for `sentimentr` package. Box plot for lexicon `huliu` shows `average` sentiment for the article is negative. Whereas, all other lexicons gave `average` sentiment as positive.

```
#Function to calculate mean of the sentiment
fun_mean <- function(x){
  return(data.frame(y=round(mean(x),3),label=round(mean(x,na.rm=T),3)))
}

fill <- "#4271AE"
line <- "#1F3552"

#Generate box plot using score of each sentence by lexicon
wfm.sentimentr.polarity %>%
  filter(fileName  == "WFM04122017a.txt") %>%
  select(fileName, huliu, jockers, nrc, sentiword) %>%
  gather(Lexicon,Score, -fileName) %>%
  ggplot(aes(x = Lexicon, y = Score)) +
        geom_boxplot(fill = fill, colour = line) +
        scale_y_continuous(name = "Polarity Scores") +
        scale_x_discrete(name = "Lexicon") +
        stat_summary(fun.y = mean, geom="point",colour="white", size=3) +
        stat_summary(fun.data = fun_mean, geom="text", vjust=-0.7) +
        ggtitle("Difference in Polarity Scores based on Lexicon - `sentimentr` package")
```

Sentiment score using `get_sentiment` function from `syuzhet` package. This package uses four lexicons `syuzhet`, `bing`, `nrc` and `afinn`.

```
#polarity score using four lexicons
wfm.syuzhet.polarity <- wfm.articles.sentence.df %>%
  mutate(syuzhet = get_sentiment(char_v = sentence, method = "syuzhet")) %>%
  mutate(bing = get_sentiment(char_v = sentence, method = "bing")) %>%
  mutate(afinn = get_sentiment(char_v = sentence, method = "afinn")) %>%
  mutate(nrc = get_sentiment(char_v = sentence, method = "nrc"))
```

Following table shows `Polarity Score` varies based on `lexicon` used from `syuzhet` package. Sentence shown in row number `28`, all four lexicons gave a negative score. Lines `32` and `33` with mixed sentiment, polarity scores varied similarly to `sentimentr` package. For line `32` lexicon `bing` gave negative polarity score and other lexicons gave a positive score. On the other hand for line `33` lexicon `syuzhet` gave a negative score, lexicon `bing` gave a positive score, `afinn` and `nrc` gave neutral polarity score.

```
#Display data in table format
wfm.syuzhet.polarity %>%
  filter(fileName  == "WFM04122017a.txt") %>%
  mutate(rn = row_number()) %>%
  filter(rn %in% c(28,32,33)) %>%
  select (rn, sentence, syuzhet, bing, afinn, nrc) %>%
  kable(digits = 3, col.names = c("Row", "Sentence", "syuzhet", "bing", "afinn", "nrc"), format='pandoc', caption = "Differe
nce in Polarity Scores based on Lexicon - `syuzhet` package")
```

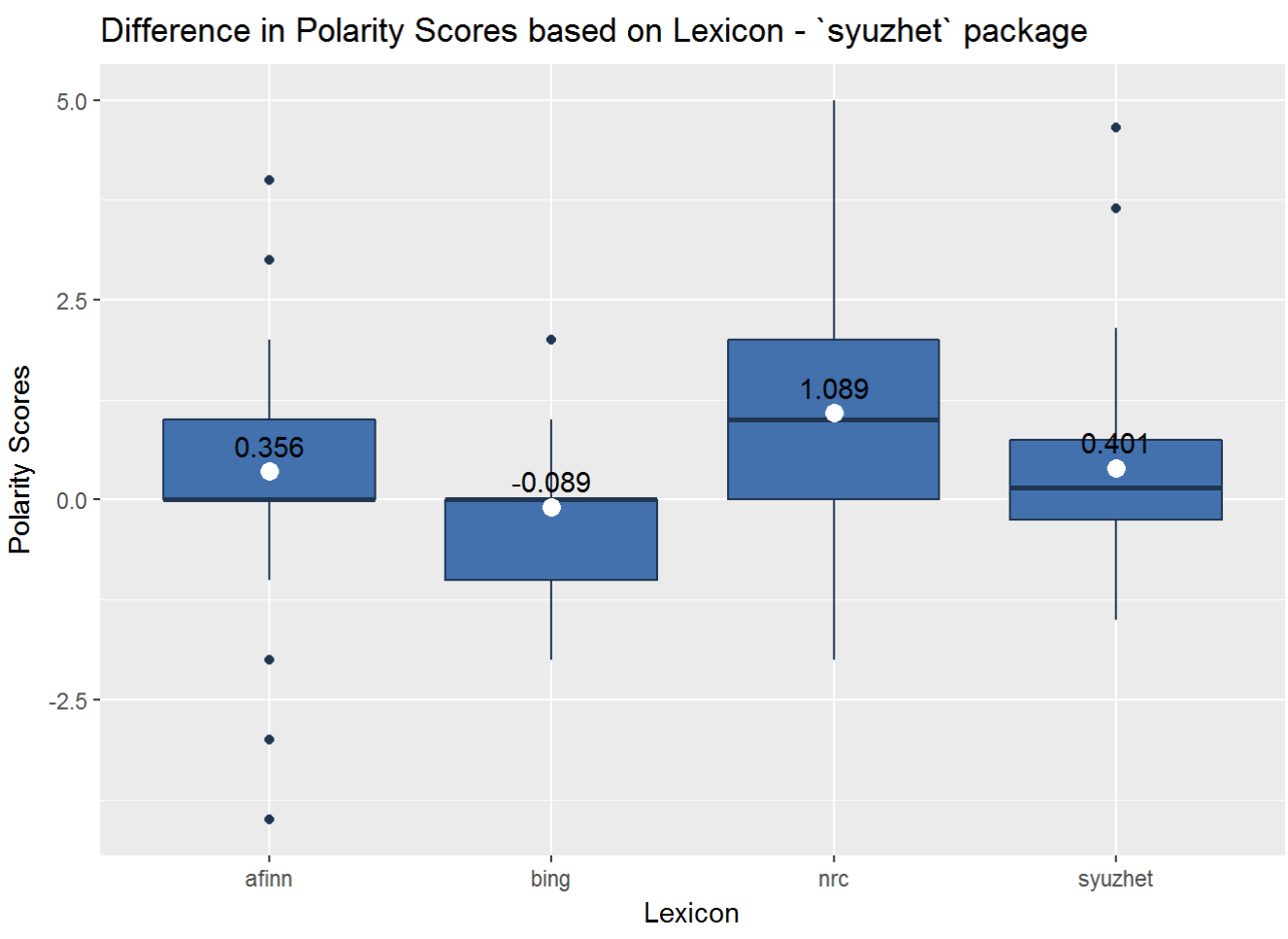
Difference in Polarity Scores based on Lexicon - `syuzhet` package

Row	Sentence	syuzhet	bing	afinn	nrc
28	whole foods notoriously high prices put it at risk of margin and food price deflation pressures he said	-1.25	-2	-2	-2

Row	Sentence	syuzhet	bing	afinn	nrc
32	ryan english a portfolio manager at sycamore township investment advisory firm foster motley said kroger could be a suitor but it s pretty expensive	0.25	-1	1	1
33	whole foods could also refocus and become a more formidable competitor to kroger	-0.60	1	0	0

Differences are shown using box plot for `syuzhet` package. Box plot for lexicon `bing` shows `average` sentiment of the article is negative. Whereas, all other lexicons gave `average` sentiment as positive.

```
#Function to calculate mean of the sentiment
fun_mean <- function(x){
  return(data.frame(y=round(mean(x),3),label=round(mean(x,na.rm=T),3)))
}

fill <- "#4271AE"
line <- "#1F3552"

#Generate box plot using score of each sentence by lexicon
wfm.syuzhet.polarity %>%
  filter(fileName  == "WFM04122017a.txt") %>%
  select(fileName, syuzhet, bing, afinn, nrc) %>%
  gather(Lexicon,Score, -fileName) %>%
  ggplot(aes(x = Lexicon, y = Score)) +
        geom_boxplot(fill = fill, colour = line) +
        scale_y_continuous(name = "Polarity Scores") +
        scale_x_discrete(name = "Lexicon") +
        stat_summary(fun.y = mean, geom="point",color="white", size=3) +
        stat_summary(fun.data = fun_mean, geom="text", vjust=-0.7) +
        ggtitle("Difference in Polarity Scores based on Lexicon - `syuzhet` package")
```



Sentiment score per article between $Apirl\ 10^{th}, 2017$ and $Apirl\ 20^{th}, 2017$. Net sentiment is obtained using `sentimentr` package.

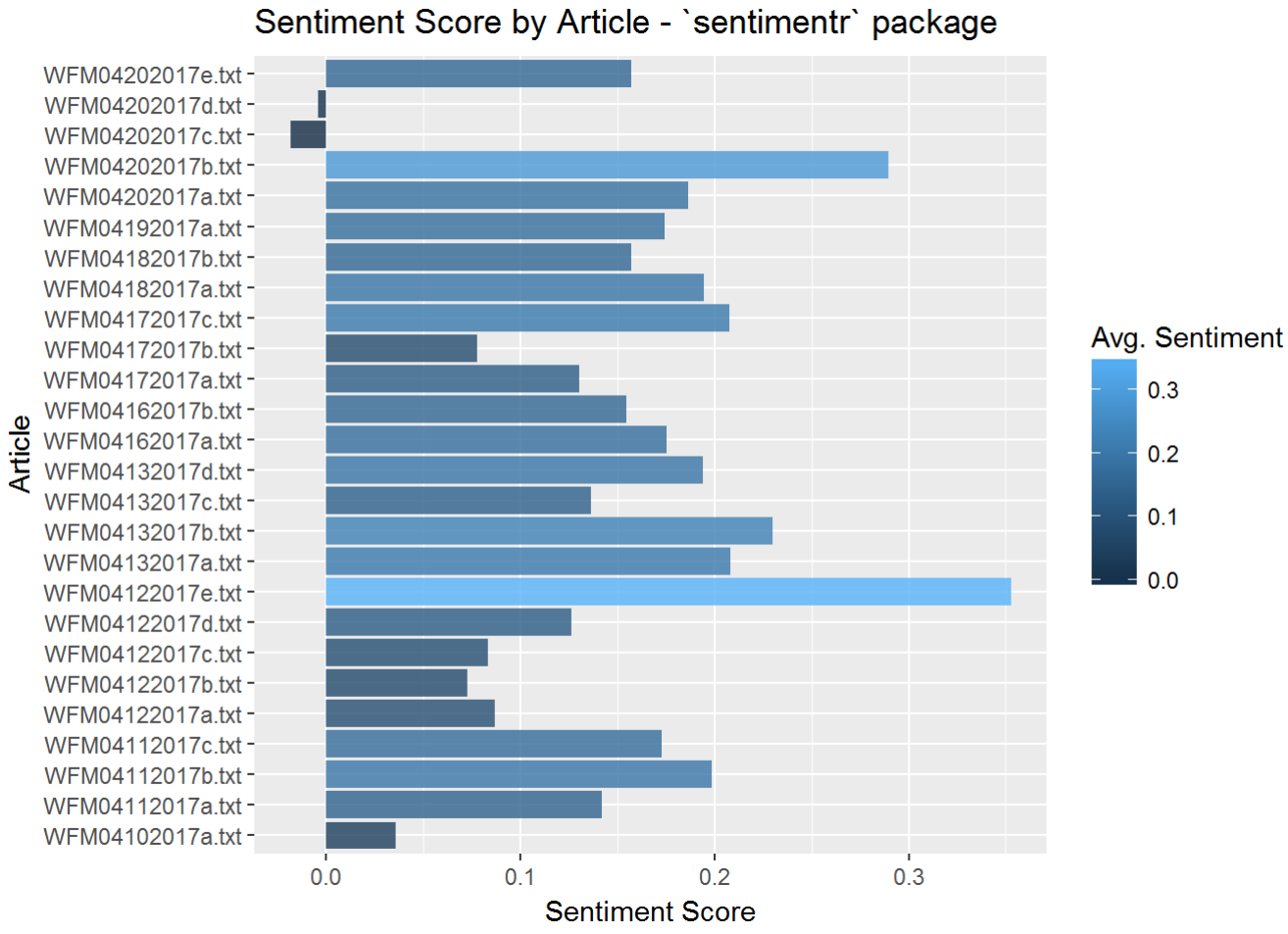
```
#Get sentiment score per article per day
wfm.sentiment.article <- with(wfm.articles.sentence.df, sentiment_by(sentence, list(fileName, articleDate))) %>%
  arrange(articleDate) %>%
  mutate(rn = row_number())

#Order data by date
wfm.sentiment.article$fileName <- factor(wfm.sentiment.article$fileName, levels=wfm.sentiment.article[order(wfm.sentiment.ar
ticle$rn), "fileName"])

#Display score
wfm.sentiment.article %>%
  filter(articleDate <= '2017-04-20') %>%
  select (fileName, ave_sentiment) %>%
  kable(digits = 3, col.names = c("Article", "Sentiment Score"), format='pandoc', caption = "Sentiment Score by Article - `s
entimentr` package")
```

Sentiment Score by Article - `sentimentr` package

Article	Sentiment Score
WFM04102017a.txt	0.036

Article	Sentiment Score
WFM04112017a.txt	0.142
WFM04112017b.txt	0.199
WFM04112017c.txt	0.173
WFM04122017a.txt	0.087
WFM04122017b.txt	0.073
WFM04122017c.txt	0.084
WFM04122017d.txt	0.126
WFM04122017e.txt	0.353
WFM04132017a.txt	0.208
WFM04132017b.txt	0.230
WFM04132017c.txt	0.136
WFM04132017d.txt	0.194
WFM04162017a.txt	0.176
WFM04162017b.txt	0.155
WFM04172017a.txt	0.130
WFM04172017b.txt	0.078
WFM04172017c.txt	0.208
WFM04182017a.txt	0.195
WFM04182017b.txt	0.157
WFM04192017a.txt	0.174
WFM04202017a.txt	0.186
WFM04202017b.txt	0.289
WFM04202017c.txt	-0.018
WFM04202017d.txt	-0.004
WFM04202017e.txt	0.157

```
#Graph score
wfm.sentiment.article %>%
  filter(articleDate <= '2017-04-20') %>%
  ggplot(mapping = aes(x = fileName, y = ave_sentiment, fill = ave_sentiment)) +
    geom_bar(alpha = 0.8, stat = "identity") +
    labs(y = "Sentiment Score", x = "Article", fill = "Avg. Sentiment") +
    ggtitle("Sentiment Score by Article - `sentimentr` package") +
    coord_flip()
```



Sentiment score per day. Net sentiment is obtained using `sentimentr` package.

```
wfm.sentiment.day <- with(wfm.articles.sentence.df, sentiment_by(sentence, list(articleDate)))

wfm.sentiment.day <- wfm.sentiment.day %>%
  mutate(articleDate = as.Date(articleDate)) %>%
  arrange(articleDate) %>%
  mutate(rn = row_number())

#Display score
wfm.sentiment.day %>%
  arrange(articleDate) %>%
  select (articleDate, ave_sentiment) %>%
  kable(digits = 3, col.names = c("Article Date", "Sentiment Score"), format='pandoc', caption = "Sentiment Score by Date -
 `sentimentr` package")
```

Sentiment Score by Date - `sentimentr` package

Article Date	Sentiment Score
2017-04-10	0.036
2017-04-11	0.179
2017-04-12	0.122
2017-04-13	0.197
2017-04-16	0.166
2017-04-17	0.181
2017-04-18	0.178
2017-04-19	0.174
2017-04-20	0.145
2017-04-21	0.184
2017-04-24	0.059
2017-04-25	0.202
2017-04-26	0.120
2017-04-27	0.114
2017-04-28	0.110
2017-04-30	0.129
2017-05-01	0.115
2017-05-02	0.098
2017-05-03	0.178
2017-05-05	0.097
2017-05-07	0.077
2017-05-08	0.161
2017-05-09	0.124
2017-05-10	0.042

```
#Graph score
wfm.sentiment.day %>%
  ggplot(mapping = aes(x = articleDate, y = ave_sentiment, fill = ave_sentiment)) +
    geom_bar(alpha = 0.8, stat = "identity") +
    labs(y = "Sentiment Score", x = "Date", fill = "Avg. Sentiment") +
    ggtitle("Sentiment Score by Date - `sentimentr` package") +
    coord_flip()
```

News Sentiment Vs. Stock Price Movement

Following analyzes shows if `Whole Foods Market` takeover news had any impact on `stock price` . Using `quantmod` package, `stock price` is obtained from `Yahoo! finance` website. Stock `moving average` is calculated using `TTR` package. Details are stored in `MongoDB` .

```
#Convert string to date format
mongo.stock.price$tradingDay = as.Date(mongo.stock.price$tradingDay)
wfm.stock.data <- mongo.stock.price

#Attach sentiment for each day
wfm.stock.data <- wfm.stock.data %>%
  left_join(wfm.sentiment.day, by = c("tradingDay" = "articleDate"))

#Filter days from April 01, 2017, as sentiment data is available from April
wfm.stock.data <- wfm.stock.data %>% filter(tradingDay >= '2017-04-01')

#Replace neutral sentiment for days data is not available
wfm.stock.data[is.na(wfm.stock.data)] <- 0

#Replace column name with (.) to (_)
colnames(wfm.stock.data) <- gsub("\\.", "_", colnames(wfm.stock.data))

#Assuming data meets all the conditions for linear regression
#Model will check impact on closing price
allCols <- colnames(wfm.stock.data)

#Remove column names based that will be not part of regression model
regCols <- allCols[!allCols %in% c("WFM_Close","tradingDay","WFM_Adjusted", "word_count", "sd","rn")]

#Create regression formula
regCols <- paste(regCols, collapse = "+")
regCols <- paste("WFM_Close~",regCols, collapse = "+")
regCols <- as.formula(regCols)

wfm.lm <- lm(regCols, data = wfm.stock.data)
summary(wfm.lm)
```

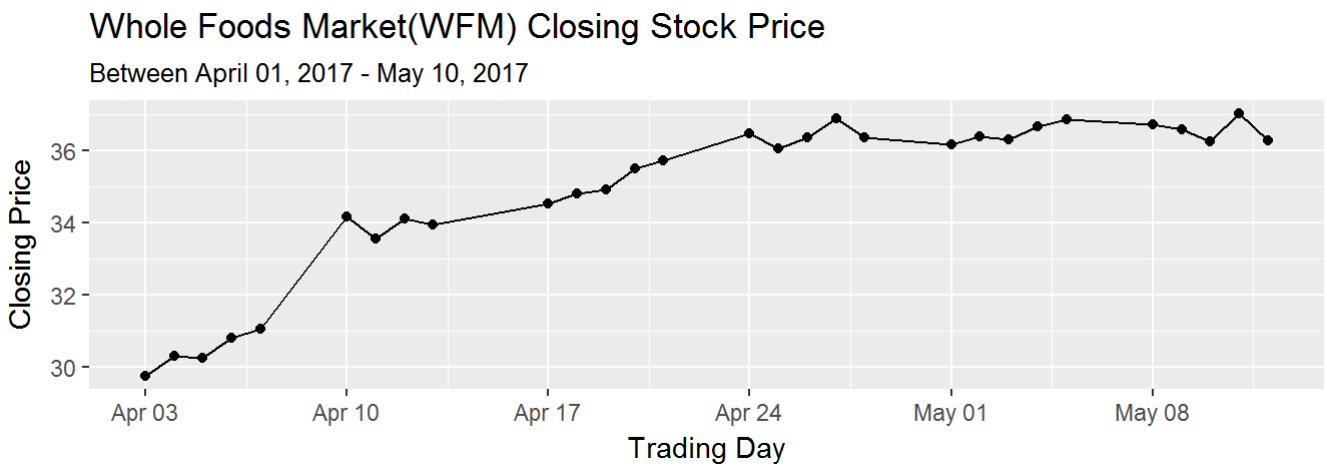
```
##
## Call:
## lm(formula = regCols, data = wfm.stock.data)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.302321 -0.072318 -0.003435  0.108456  0.244167
##
## Coefficients: (1 not defined because of singularities)
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     2.375e+01  2.132e+01   1.114  0.27982
## WFM_Open       -3.163e-01  1.440e-01  -2.197  0.04133 *
## WFM_High        6.671e-01  1.830e-01   3.645  0.00185 **
## WFM_Low         1.331e-01  2.465e-01   0.540  0.59599
## WFM_Volume      1.752e-09  2.071e-08   0.085  0.93351
## avg_stockprice         NA         NA      NA       NA
## ma_10          -2.925e-01  4.102e-01  -0.713  0.48497
## ema_10          7.575e-01  8.010e-01   0.946  0.35682
## vol_garman      1.780e+00  1.213e+00   1.467  0.15950
## ma_50          -2.248e+00  2.594e+00  -0.867  0.39754
## ema_50          1.501e+00  2.165e+00   0.693  0.49709
## ave_sentiment   6.097e-01  9.166e-01   0.665  0.51438
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1825 on 18 degrees of freedom
## Multiple R-squared:  0.9959, Adjusted R-squared:  0.9936
## F-statistic: 432.4 on 10 and 18 DF,  p-value: < 2.2e-16
```

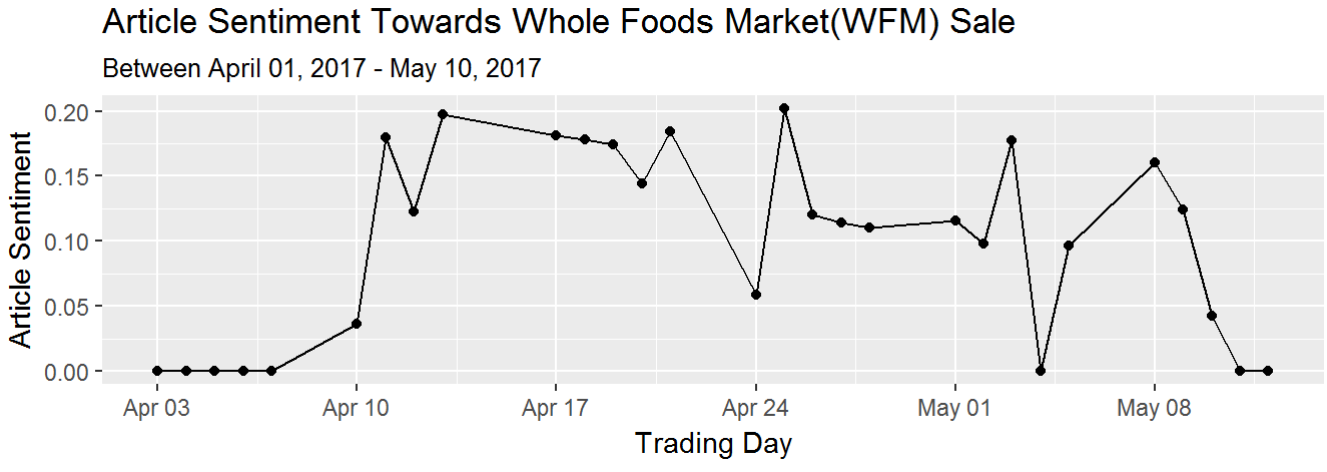
```
#Second go
#Apply backward elimination process, lets remove some columns
regCols <- allCols[!allCols %in% c("WFM_Close","tradingDay","WFM_Adjusted", "word_count", "sd","rn", "WFM_Low", "WFM_High",
"ma_50", "ema_50", "ma_10", "ema_10", "vol_garman")]

#Create regression formula
regCols <- paste(regCols, collapse = "+")
regCols <- paste("WFM_Close~",regCols, collapse = "+")
regCols <- as.formula(regCols)

wfm.lm <- lm(regCols, data = wfm.stock.data)
summary(wfm.lm)
```

```
##
## Call:
## lm(formula = regCols, data = wfm.stock.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.33289 -0.15162  0.00489  0.11194  0.33609
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.586e-01  6.065e-01   0.591    0.560
## WFM_Open       -1.273e+00  1.575e-01  -8.081 2.64e-08 ***
## WFM_Volume      7.167e-09  8.511e-09   0.842    0.408
## avg_stockprice  2.260e+00  1.639e-01  13.791 6.66e-13 ***
## ave_sentiment   4.085e-01  5.482e-01   0.745    0.463
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1935 on 24 degrees of freedom
## Multiple R-squared:  0.9938, Adjusted R-squared:  0.9928
## F-statistic: 959.6 on 4 and 24 DF,  p-value: < 2.2e-16
```

Graphical presentation of the Whole Food Market `sentiment` and `stock price`.

```
plot1 <- wfm.stock.data %>%
ggplot(aes(x=tradingDay, y=WFM_Close, group=1)) +
    geom_line() +
    geom_point() + labs(x="Trading Day", y="Closing Price", title = "Whole Foods Market(WFM) Closing Stock Price", subtitle
= "Between April 01, 2017 - May 10, 2017")

plot2 <- wfm.stock.data %>%
ggplot(aes(x=tradingDay, y=ave_sentiment, group=1)) +
    geom_line() +
    geom_point() + labs(x="Trading Day", y="Article Sentiment", title = "Article Sentiment Towards Whole Foods Market(WFM) S
ale", subtitle = "Between April 01, 2017 - May 10, 2017")

grid.arrange(plot1, plot2, nrow = 2)
```





Conclusions

- Basic Sentiment Analysis focuses on the meanings of the words and phrases. It gives information if words used are positive or negative. It is suitable for analysis of smaller sentences or phrases used in messaging tools, example: Twitter or Facebook data feeds. The major drawback is it does not reflect proper sentiment when `negation` is part of the sentence.
- Sentence Sentiment Analysis focuses on the whole sentence and not just words. It works well with the larger amount of data, example: news articles, reviews, books, etc. Sentiment analysis greatly depends on `package` algorithms and `lexicons` . One has to thoroughly test against data before using `packages` and `lexicons` in production. Packages used on this project demonstrated they fall short if the mixed sentiment is expressed in a single sentence.
- Sentiment aggregation feature `sentiment_by` of `sentimentr` package is very helpful while deriving sentiment for higher grouping other than just sentences. `syuzhet` packages lacks that feature.
- Adjusted R^2 value did improve from `0.9936` to `0.9928` when `moving averages` and `volatility` data was removed suggesting, `news sentiment` may have some impact. This cannot be entirely trusted because `moving averages` , and `volatility` are primary indicators that are used in trading stock.
- Daily `stock price movement` and `sentiment` graphs suggest that they are independent of each other. In other words, there may be many other influential factors that affect `stock price` direction, example: dividends, earning per share, cash flow, etc.

References:

- http://juliasilge.com/blog/Life-Changing-Magic/ (http://juliasilge.com/blog/Life-Changing-Magic/)
- http://uc-r.github.io/sentiment_analysis (http://uc-r.github.io/sentiment_analysis)
- https://github.com/trinker/sentimentr (https://github.com/trinker/sentimentr)
- http://www.matthewjockers.net/2015/02/02/syuzhet/ (http://www.matthewjockers.net/2015/02/02/syuzhet/)
- https://blog.exploratory.io/twitter-sentiment-analysis-scoring-by-sentence-b4d455de3560 (https://blog.exploratory.io/twitter-sentiment-analysis-scoring-by-sentence-b4d455de3560)
- http://stackoverflow.com/questions/19876505/boxplot-show-the-value-of-mean (http://stackoverflow.com/questions/19876505/boxplot-show-the-value-of-mean)