How to do develop sentiment analysis to analyse stocks

Andi Shehu. Byteflow Dynamics

9/25/2018

Contents

1 Overview

1.1 Problem statement in brief

In the world of finance traders and analysts are always vying to stay ahead of the curve when picking stocks using any data sets that they can get their hands on. This can be in the form of time series which can be purchased from the likes of a Bloomberg terminal or it can be an earning reports.

In addition, many of the large institutions deploy an army of analysts to analyze news coverage on particular stocks. In the old fashioned world, this is done by manually going through articles covering a list of stocks and counting positive and negative words for the overall sentiment. A final score is calculated using some internal proprietary algorithm. The score is handed over to the traders who then decide whether to buy, sell, or short a partiular stock.

There are however a few challenges with this methodology:
i it is not a scalable solution, often firms will limit the number of sources and stocks to follow;
ii humans make mistakes, they get tired, and auditing an analyst's work adds to the cost of production.

1.2 Problem solution in brief

Text analytics and Natural Language Processing (NLP) allow for automation of existing scoring methodologies, scaling them up, and quickly iterating on newer models. Once the model is up and running, it can be scaled to score thousands of articles at once. Even though these methods have existed for decades, what is new is the tools from cloud computing and open source community such as R and Python programming languages. We now have the necessary tools to scrape news sources from top blogs and sites.

Here, we present novel approaches to sentiment analysis over one hundred stocks. The data is scraped from ten news sources over the period of a week and the sentiment is calculated for that week.

2 Methodology

Sentiment analysis

We are going to walk through three methodologies to perform sentiment analysis using key words. First, we will start with a more fundamental methodology of using single tokens to analyze the sentiment. Second, we will then move into `n-grams` where we take into consideration negation. Third, we will use finance dictionary developed for more accurate representation of industry specific words.

2.1 The data

The data is scraped from ten news sources using Python, which is then stored in Amazon Web Services. `R` programming language is used for the data cleaning and analysis.

Sample data

In order to demonstrate the scoring process we will select a smaller sample of our full data set.

```
# read raw data
data <- read_csv("fulltext_small_sample.csv")

# column names
names(data)
```

```
## [1] "Link"    "text"    "date"    "company"
```

```
dim(data)
```

```
## [1] 703   4
```

Many of the articles scraped do not actually contain a stock and are labeled as NA (the stock name was probably in metadata). We will remove these articles.

```
df_nona <- data %>%
  filter(is.na(company)==FALSE)

# save the data for future use

#write_csv(df_nona, "articles.csv")
dim(df_nona)
```

```
## [1] 401   4
```

We are left with about 400 articles to analyze.

Select and keep text and stock columns only.

```
df<- df_nona
#df <- df_nona %>%
#  select(company, text)

#head(df)
```

2.2 Steps towards sentiment analysis.

1. Make data tidy
2. Remove stopwords
3. Choose lexicon

4. Add weights
5. Normalize scoring

2.2.1 Step 1. Make data tidy

We will follow the definition of tidy data by Hadley Wickham, Chief Scientist of RStudio (RStudio is the R IDE we are using). The definition of tidy data is:

- Each variable is saved in its own column.
- Each observation is saved in its own row.

- Each observation has its own cell.

Unnest_tokens

`unnest_tokens` is a packages which takes a full text and splits it into individual words, or tokens. We will use it to split each article into individual words.

It works like this:

```
library(tidytext)

text <- "this is a single sentence about sentiment analysis"
writeLines(text)
```

```
## this is a single sentence about sentiment analysis
```

```
# add the text into a dataframe

text_df <- data_frame(text)

# use the unnest_tokens to tokenize the text and turn it into tidy data.
text_df %>%
    unnest_tokens(word, text)
```

```
## # A tibble: 8 x 1
##   word
##   <chr>
## 1 this
## 2 is
## 3 a
## 4 single
## 5 sentence
## 6 about
## 7 sentiment
## 8 analysis
```

We will now apply the same methodology to our entire data set.

```
# first lets add a unique identifier
df_id <- df %>%
  mutate(ID = row_number()) %>%
  select(ID, company,date, text,Link )

# now each word becomes its own entry
df_tidy <- df_id %>%
  unnest_tokens(word, text)
df_tidy
```

```
## # A tibble: 224,399 x 5
##       ID company date       Link                                     word
##    <int> <chr>   <date>     <chr>                                    <chr>
##  1     1 Nvidia  2018-08-29 http://www.marketwatch.com/story/here~ what
##  2     1 Nvidia  2018-08-29 http://www.marketwatch.com/story/here~ a
##  3     1 Nvidia  2018-08-29 http://www.marketwatch.com/story/here~ differ~
##  4     1 Nvidia  2018-08-29 http://www.marketwatch.com/story/here~ a
##  5     1 Nvidia  2018-08-29 http://www.marketwatch.com/story/here~ couple
##  6     1 Nvidia  2018-08-29 http://www.marketwatch.com/story/here~ of
##  7     1 Nvidia  2018-08-29 http://www.marketwatch.com/story/here~ months
##  8     1 Nvidia  2018-08-29 http://www.marketwatch.com/story/here~ can
##  9     1 Nvidia  2018-08-29 http://www.marketwatch.com/story/here~ make
## 10     1 Nvidia  2018-08-29 http://www.marketwatch.com/story/here~ the
## # ... with 224,389 more rows
```

2.2.2 Stopwords

We see that there are a many of common words in the list, often refere to as stopwords, which do not have a lot of value, words such as: “or”, “and”, “the”.

We will remove them using the built in data set for stopwords `stop_words`.

```
data(stop_words)
tail(stop_words, 20)
```

```
## # A tibble: 20 x 2
##    word      lexicon
##    <chr>     <chr>
##  1 whose     onix
##  2 why       onix
##  3 will      onix
##  4 with      onix
##  5 within    onix
##  6 without   onix
##  7 work      onix
##  8 worked    onix
##  9 working   onix
## 10 works     onix
## 11 would     onix
## 12 year      onix
## 13 years     onix
## 14 yet       onix
## 15 you       onix
## 16 young     onix
## 17 younger   onix
## 18 youngest  onix
## 19 your      onix
## 20 yours     onix
```

```
# This can be done simply anti joining the two data sets.
df_tidy <- df_tidy %>%
  anti_join(stop_words, by = "word")
df_tidy
```

```
## # A tibble: 113,660 x 5
```

```
##        ID company date       Link                                   word
##     <int> <chr>   <date>     <chr>                                  <chr>
##  1      1 Nvidia  2018-08-29 http://www.marketwatch.com/story/here~ differ~
##  2      1 Nvidia  2018-08-29 http://www.marketwatch.com/story/here~ couple
##  3      1 Nvidia  2018-08-29 http://www.marketwatch.com/story/here~ months
##  4      1 Nvidia  2018-08-29 http://www.marketwatch.com/story/here~ u.s
##  5      1 Nvidia  2018-08-29 http://www.marketwatch.com/story/here~ stock
##  6      1 Nvidia  2018-08-29 http://www.marketwatch.com/story/here~ market
##  7      1 Nvidia  2018-08-29 http://www.marketwatch.com/story/here~ setting
##  8      1 Nvidia  2018-08-29 http://www.marketwatch.com/story/here~ records
##  9      1 Nvidia  2018-08-29 http://www.marketwatch.com/story/here~ inform~
## 10      1 Nvidia  2018-08-29 http://www.marketwatch.com/story/here~ techno~
## # ... with 113,650 more rows
```

2.2.3 Exploratory data analysis (EDA)

Let‘s see which words are most common.

```
df_tidy %>%
  count(word, sort=TRUE)
```

```
## # A tibble: 14,038 x 2
##    word        n
##    <chr>   <int>
##  1 stock     993
##  2 tesla     891
##  3 company   802
##  4 market    721
##  5 shares    558
##  6 u.s       540
##  7 stocks    536
##  8 apple     514
##  9 amazon    458
## 10 billion   446
## # ... with 14,028 more rows
```

How about most common words by company.

```
df_tidy %>%
  group_by(company) %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 41,774 x 3
## # Groups:   company [45]
##    company word        n
##    <chr>   <chr>   <int>
##  1 Tesla   tesla     766
##  2 Tesla   musk      382
##  3 Amazon  amazon    281
##  4 Tesla   company   250
##  5 Tesla   stock     250
##  6 Apple   apple     237
##  7 Tesla   private   232
##  8 Amazon  stock     216
##  9 Amazon  market    168
```

```
## 10 Tesla   elon      167
## # ... with 41,764 more rows
```

We see that Tesla is very much defined by its chief executive, Elon Musk.

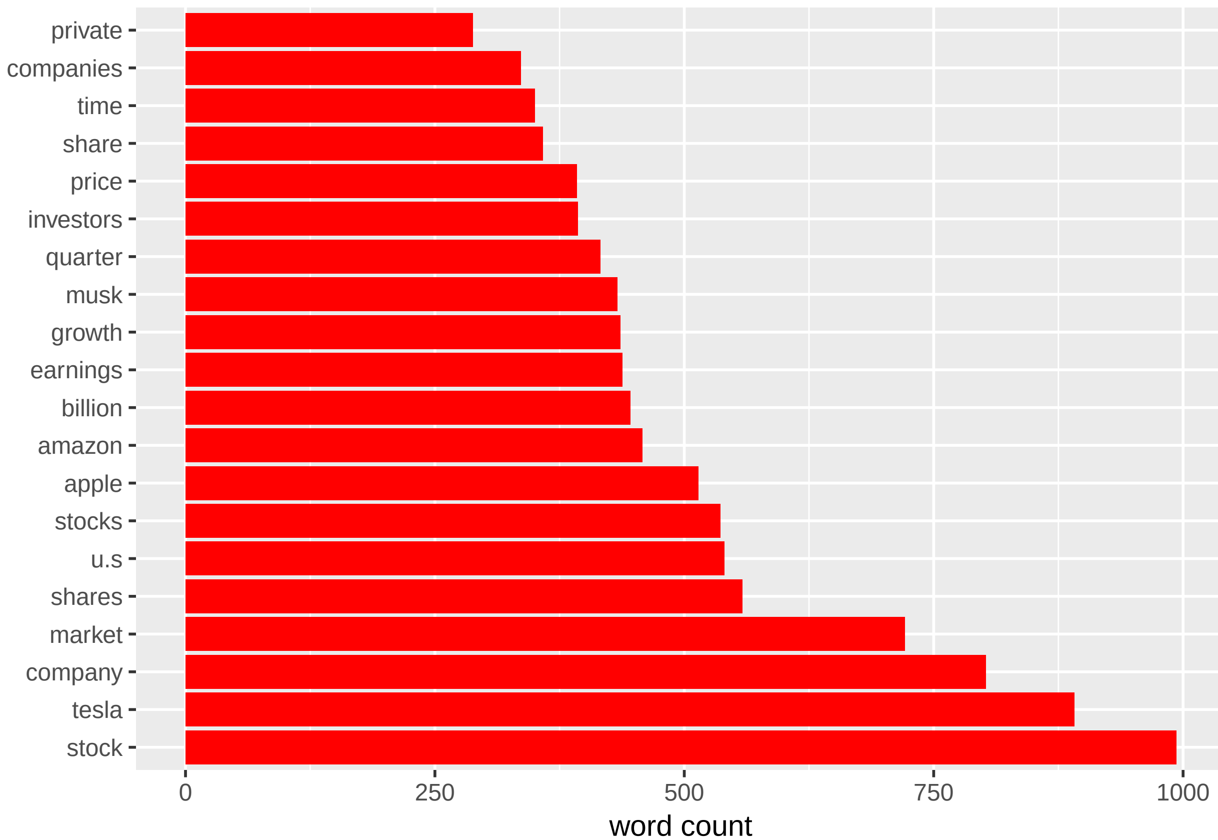
Let's plot the most overall common words.

```r
library(ggplot2)

# Create a dataframe woth count

df_count <- df_tidy %>%
  count(word, sort=TRUE) %>%
  head(20)


ggplot(data = df_count, aes(x = reorder(word, -n), y = n))+
  geom_col(fill = "red")+
  coord_flip()+
  xlab(NULL)+
  ylab("word count")
```



Let's try a wordcoloud just for fun.

```r
library(wordcloud)

df_tidy %>%
  count(word) %>%
      with(wordcloud(word,n, max.words = 100,color = "blue"))
```

2.3 Sentiment analysis, one-gram

The data has been tidying up and is ready for sentiment analysis. There are several built-in packages we can use to get sentiments. Let us try them out.

```
sentiments
```

```
## # A tibble: 27,314 x 4
##    word        sentiment lexicon score
##    <chr>       <chr>     <chr>   <int>
##  1 abacus      trust     nrc        NA
##  2 abandon     fear      nrc        NA
##  3 abandon     negative  nrc        NA
##  4 abandon     sadness   nrc        NA
##  5 abandoned   anger     nrc        NA
##  6 abandoned   fear      nrc        NA
##  7 abandoned   negative  nrc        NA
##  8 abandoned   sadness   nrc        NA
##  9 abandonment anger     nrc        NA
## 10 abandonment fear      nrc        NA
## # ... with 27,304 more rows
```

```
unique(sentiments$lexicon)
```

```
## [1] "nrc"      "bing"     "AFINN"    "loughran"
```

There are four lexicons. We will use `bing` and `loughran` as for now we are interested in positive and negative sentiments.

2.3.1 Bing lexicon

Let's start with the `bing` sentiment

```
bing <- get_sentiments("bing")
bing
```

```
## # A tibble: 6,788 x 2
##    word        sentiment
##    <chr>       <chr>
##  1 2-faced     negative
##  2 2-faces     negative
##  3 a+          positive
##  4 abnormal    negative
##  5 abolish     negative
##  6 abominable  negative
##  7 abominably  negative
##  8 abominate   negative
##  9 abomination negative
## 10 abort       negative
## # ... with 6,778 more rows
```

From our original dataset we are going to keep the words in the lexicon. This can be easily achieved using `inner_join()` to combine the two data sets and keep only the words that appear in both the stocks data and the `bing` data.

```
df_sentiment <- df_tidy %>%
  inner_join(bing, by = "word")
df_sentiment
```

```
## # A tibble: 9,574 x 6
##       ID company date       Link                          word   sentiment
##    <int> <chr>   <date>     <chr>                         <chr>  <chr>
##  1     1 Nvidia  2018-08-29 http://www.marketwatch.com/s~ leadi~ positive
##  2     1 Nvidia  2018-08-29 http://www.marketwatch.com/s~ gloomy negative
##  3     1 Nvidia  2018-08-29 http://www.marketwatch.com/s~ gained positive
##  4     1 Nvidia  2018-08-29 http://www.marketwatch.com/s~ remar~ positive
##  5     1 Nvidia  2018-08-29 http://www.marketwatch.com/s~ decent positive
##  6     1 Nvidia  2018-08-29 http://www.marketwatch.com/s~ suppo~ positive
##  7     1 Nvidia  2018-08-29 http://www.marketwatch.com/s~ miss   negative
##  8     1 Nvidia  2018-08-29 http://www.marketwatch.com/s~ winne~ positive
##  9     1 Nvidia  2018-08-29 http://www.marketwatch.com/s~ cheap  negative
## 10     1 Nvidia  2018-08-29 http://www.marketwatch.com/s~ advan~ positive
## # ... with 9,564 more rows
```

Now we can simply `count` all the positive and negative sentiments for an overall sentiment score. We can do this per article and overall press coverage for this particular time period.

```
df_pos_neg <- df_sentiment %>%
  group_by(ID, company) %>%
  count(sentiment)
df_pos_neg
```

```
## # A tibble: 774 x 4
## # Groups:   ID, company [396]
##       ID company sentiment     n
##    <int> <chr>   <chr>     <int>
```

```
## 1      1 Nvidia  negative     14
## 2      1 Nvidia  positive     15
## 3      2 Walmart negative     55
## 4      2 Walmart positive     38
## 5      3 Tesla   negative     25
## 6      3 Tesla   positive     20
## 7      4 Apple   negative      3
## 8      4 Apple   positive      4
## 9      5 Tesla   negative      3
## 10     5 Tesla   positive      6
## # ... with 764 more rows
```

Spread the data so positive and negative are their own column.

```
df_spread <- df_pos_neg %>%
  spread(sentiment, n)
df_spread
```

```
## # A tibble: 396 x 4
## # Groups:   ID, company [396]
##        ID company  negative positive
##     <int> <chr>       <int>    <int>
## 1       1 Nvidia         14       15
## 2       2 Walmart        55       38
## 3       3 Tesla          25       20
## 4       4 Apple           3        4
## 5       5 Tesla           3        6
## 6       6 Amazon         11       13
## 7       7 Google         18        8
## 8       8 Google          7       10
## 9       9 Facebook       23        6
## 10     10 Netflix         8       13
## # ... with 386 more rows
```

Calculate net sentiment for each article by simply taking the difference in the number of positive and negative words devided by their sum for normalization.

```
df_net_article <- df_spread %>%
  mutate(net = (positive-negative)/(positive+negative))
df_net_article
```

```
## # A tibble: 396 x 5
## # Groups:   ID, company [396]
##        ID company  negative positive     net
##     <int> <chr>       <int>    <int>   <dbl>
## 1       1 Nvidia         14       15  0.0345
## 2       2 Walmart        55       38 -0.183
## 3       3 Tesla          25       20 -0.111
## 4       4 Apple           3        4  0.143
## 5       5 Tesla           3        6  0.333
## 6       6 Amazon         11       13  0.0833
## 7       7 Google         18        8 -0.385
## 8       8 Google          7       10  0.176
## 9       9 Facebook       23        6 -0.586
## 10     10 Netflix         8       13  0.238
## # ... with 386 more rows
```

Calculate net sentiment per company over all the articles for that particular time period.

```
bing_net <-df_net_article %>%
  group_by(company) %>%
  summarise(net_overall = mean(net, na.rm = TRUE)) %>%
  arrange(net_overall)


bing_net %>% head(20)
```

```
## # A tibble: 20 x 2
##    company             net_overall
##    <chr>                     <dbl>
##  1 Pfizer                  -0.487
##  2 American Express        -0.429
##  3 Twitter                 -0.317
##  4 CVS Health              -0.315
##  5 General Electric        -0.296
##  6 IBM                     -0.190
##  7 Chevron                 -0.133
##  8 Charles Schwab          -0.125
##  9 JP Morgan               -0.0942
## 10 Facebook                -0.0902
## 11 Tesla                   -0.0884
## 12 TJX                     -0.0811
## 13 Exxon                   -0.0613
## 14 Morgan Stanley          -0.0455
## 15 Starbucks               -0.0453
## 16 Texas Instruments       -0.0417
## 17 Goldman Sachs           -0.0244
## 18 Cisco                   -0.0208
## 19 Berkshire Hathaway       0
## 20 Salesforce               0.0256
```

```
bing_net %>% tail(20)
```

```
## # A tibble: 20 x 2
##    company               net_overall
##    <chr>                       <dbl>
##  1 EOG Resources               0.102
##  2 Walmart                     0.151
##  3 Nvidia                      0.171
##  4 Oracle                      0.188
##  5 Boeing                      0.2
##  6 Pepsi                       0.208
##  7 Netflix                     0.208
##  8 Intel                       0.213
##  9 Schlumberger                0.23
## 10 Bank of America             0.247
## 11 Caterpillar                 0.374
## 12 Comcast                     0.383
## 13 Home Depot                  0.412
## 14 Qualcomm                    0.44
## 15 Disney                      0.479
## 16 Microsoft                   0.523
```

```
## 17 United Technologies        0.556
## 18 Celgene                    0.581
## 19 Visa                       0.6
## 20 Occidental Petroleum     NaN
```

2.4 Loughran: finance lexicon

Lets try using a different lexicon, something more specific to finance. We want to use this because there are specific words in finance that when taken out of context may give the wrong sentiment. . Words such as tax, cost, capital, board, liability, foreign, and vice appear on many lexicons. In financial statements, vice will often be a title, vice-president.

Loughran et. al. [1]. of University of Notre Dame have developed a domain specific lexicon which is a great improvemnt on more traditional dictionaries.

```
loughran <- get_sentiments("loughran")
dim(loughran)
```

```
## [1] 4149    2
```

There are over 4000 words in the `loughran` dictionary with tho following sentiments.

```
unique(loughran$sentiment)
```

```
## [1] "negative"     "positive"     "uncertainty"  "litigious"
## [5] "constraining" "superfluous"
```

We will focus on the positive and negative sentiments.

Lets inner join with the `loghran` sentiments to keep only the key words.

```
df_loughran <- df_tidy %>%
  inner_join(loughran)
df_loughran
```

```
## # A tibble: 7,257 x 6
##       ID company date       Link                         word    sentiment
##    <int> <chr>   <date>     <chr>                        <chr>   <chr>
##  1     1 Nvidia  2018-08-29 http://www.marketwatch.com/~ leading positive
##  2     1 Nvidia  2018-08-29 http://www.marketwatch.com/~ declin~ negative
##  3     1 Nvidia  2018-08-29 http://www.marketwatch.com/~ closing negative
##  4     1 Nvidia  2018-08-29 http://www.marketwatch.com/~ gained  positive
##  5     1 Nvidia  2018-08-29 http://www.marketwatch.com/~ miss    negative
##  6     1 Nvidia  2018-08-29 http://www.marketwatch.com/~ winners positive
##  7     1 Nvidia  2018-08-29 http://www.marketwatch.com/~ volati~ negative
##  8     1 Nvidia  2018-08-29 http://www.marketwatch.com/~ volati~ uncertai~
##  9     1 Nvidia  2018-08-29 http://www.marketwatch.com/~ smooth~ positive
## 10     1 Nvidia  2018-08-29 http://www.marketwatch.com/~ volati~ negative
## # ... with 7,247 more rows
```

Count the sentiments

```
df_loughran_senti <- df_loughran %>%
  group_by(ID, company) %>%
  count(sentiment) %>%
  spread(sentiment, n) %>%
   replace_na(list(positive = 0, positive = "unknown")) %>%
```

[1] When Is a Liability Not a Liability?

```
  replace_na(list(negative = 0, negative = "unknown")) %>%
  ungroup()
df_loughran_senti
```

```
## # A tibble: 392 x 8
##       ID company constraining litigious negative positive superfluous
##    <int> <chr>          <int>     <int>    <dbl>    <dbl>       <int>
##  1     1 Nvidia            NA        NA       11        5          NA
##  2     2 Walmart           NA        NA       39       19          NA
##  3     3 Tesla             NA         2       14       14          NA
##  4     4 Apple             NA        NA        1        1          NA
##  5     5 Tesla             NA        NA        2        2          NA
##  6     6 Amazon            NA        NA        7       10          NA
##  7     7 Google            NA        NA       17        3          NA
##  8     8 Google             2         2        9        5          NA
##  9     9 Facebo~            7        NA       14        3          NA
## 10    10 Netflix           NA        NA        4        9          NA
## # ... with 382 more rows, and 1 more variable: uncertainty <int>
```

Lets select only the positive and negative sentiments

```
loughran_pos_neg <- df_loughran_senti %>%
  select(ID, company, positive, negative)
loughran_pos_neg
```

```
## # A tibble: 392 x 4
##       ID company  positive negative
##    <int> <chr>       <dbl>    <dbl>
##  1     1 Nvidia          5       11
##  2     2 Walmart        19       39
##  3     3 Tesla          14       14
##  4     4 Apple           1        1
##  5     5 Tesla           2        2
##  6     6 Amazon         10        7
##  7     7 Google          3       17
##  8     8 Google          5        9
##  9     9 Facebook        3       14
## 10    10 Netflix         9        4
## # ... with 382 more rows
```

```
loughran_pos_neg <- loughran_pos_neg %>%
  mutate(net = (positive - negative)/(positive+negative))

loughran_pos_neg
```

```
## # A tibble: 392 x 5
##       ID company  positive negative    net
##    <int> <chr>       <dbl>    <dbl>  <dbl>
##  1     1 Nvidia          5       11 -0.375
##  2     2 Walmart        19       39 -0.345
##  3     3 Tesla          14       14  0
##  4     4 Apple           1        1  0
##  5     5 Tesla           2        2  0
##  6     6 Amazon         10        7  0.176
##  7     7 Google          3       17 -0.7
##  8     8 Google          5        9 -0.286
```

```
## 9      9 Facebook        3       14 -0.647
## 10    10 Netflix         9        4  0.385
## # ... with 382 more rows
```

Combine it with original data

```
df_final <- loughran_pos_neg %>%
  left_join(df_id, by = "ID")
df_final
```

```
## # A tibble: 392 x 9
##       ID company.x positive negative    net company.y date       text
##    <int> <chr>        <dbl>    <dbl>  <dbl> <chr>     <date>     <chr>
##  1     1 Nvidia           5       11 -0.375 Nvidia    2018-08-29 "Wha~
##  2     2 Walmart         19       39 -0.345 Walmart   2018-08-24 "J.C~
##  3     3 Tesla           14       14  0     Tesla     2018-08-27 "The~
##  4     4 Apple            1        1  0     Apple     2018-08-27 "Sha~
##  5     5 Tesla            2        2  0     Tesla     2018-08-27 "Cin~
##  6     6 Amazon          10        7  0.176 Amazon    2018-08-24 "A m~
##  7     7 Google           3       17 -0.7   Google    2018-08-25 "Day~
##  8     8 Google           5        9 -0.286 Google    2018-08-23 "Net~
##  9     9 Facebook         3       14 -0.647 Facebook  2018-08-22 "Tee~
## 10    10 Netflix          9        4  0.385 Netflix   2018-08-24 "Sun~
## # ... with 382 more rows, and 1 more variable: Link <chr>
```

Calculate the net score

```
loughran_net <- loughran_pos_neg %>%
  group_by(company) %>%
  summarise(net_loughran = mean(net, na.rm = TRUE)) %>%
  arrange(net_loughran)
loughran_net
```

```
## # A tibble: 45 x 2
##    company            net_loughran
##    <chr>                     <dbl>
##  1 American Express         -1
##  2 Visa                     -1
##  3 Twitter                  -0.721
##  4 Schlumberger             -0.714
##  5 General Electric         -0.684
##  6 Pfizer                   -0.652
##  7 Comcast                  -0.644
##  8 Texas Instruments        -0.615
##  9 Wells Fargo              -0.558
## 10 Salesforce               -0.524
## # ... with 35 more rows
```

3 Compare lexicons

Lets compare the two lexicons

```
two_lex <- loughran_net %>%
  inner_join(bing_net, by = "company")
```

```r
print.data.frame(two_lex)
```

```
##                  company net_loughran net_overall
## 1       American Express  -1.00000000 -0.42857143
## 2                   Visa  -1.00000000  0.60000000
## 3                Twitter  -0.72058824 -0.31684492
## 4           Schlumberger  -0.71428571  0.23000000
## 5       General Electric  -0.68421053 -0.29574468
## 6                 Pfizer  -0.65217391 -0.48717949
## 7                Comcast  -0.64444444  0.38333333
## 8      Texas Instruments  -0.61538462 -0.04166667
## 9            Wells Fargo  -0.55789474  0.06516291
## 10            Salesforce  -0.52380952  0.02564103
## 11                 Tesla  -0.42234480 -0.08840286
## 12                 Pepsi  -0.42222222  0.20833333
## 13                 Cisco  -0.40714286 -0.02083333
## 14              Facebook  -0.35758149 -0.09016626
## 15                Google  -0.35210219  0.08054935
## 16             JP Morgan  -0.35000000 -0.09415584
## 17                Boeing  -0.33333333  0.20000000
## 18                  Nike  -0.29365079  0.03913630
## 19                 Exxon  -0.26595573 -0.06131082
## 20        Morgan Stanley  -0.26414141 -0.04549431
## 21               Chevron  -0.25851494 -0.13327775
## 22                 Apple  -0.20447318  0.02874533
## 23            CVS Health  -0.10000000 -0.31521739
## 24  United Technologies  -0.10000000  0.55555556
## 25                Amazon  -0.09570771  0.10137063
## 26        Charles Schwab  -0.09090909 -0.12500000
## 27         Goldman Sachs  -0.07578644 -0.02437564
## 28                 Intel  -0.06165414  0.21327228
## 29                   TJX  -0.05882353 -0.08108108
## 30                   IBM  -0.01666667 -0.19007937
## 31           Caterpillar  -0.01111111  0.37362637
## 32               Netflix   0.01133884  0.20835321
## 33                Nvidia   0.03360528  0.17136192
## 34                Disney   0.10317460  0.47938034
## 35             Starbucks   0.13032581 -0.04530651
## 36               Walmart   0.13602763  0.15094375
## 37            Home Depot   0.14285714  0.41176471
## 38       Bank of America   0.20000000  0.24736842
## 39              Qualcomm   0.29411765  0.44000000
## 40               Celgene   0.37500000  0.58139535
## 41         EOG Resources   0.47826087  0.10204082
## 42                Oracle   0.66666667  0.18790850
## 43             Microsoft   0.67777778  0.52272727
## 44    Berkshire Hathaway   1.00000000  0.00000000
## 45 Occidental Petroleum   1.00000000         NaN
```

3.1 Sentiment analysis 2-gram

So far we have calculated the sentiment by simply adding the positive and negative words for a net number of positive and negative words. However, there are negation words for which we have not accounted for.

Consider the following sentence: "Investers are **not** confident in Tesla and Elon Musk is **not** happy with shortsellers" In this sentence the the words "confident" and "happy" would be considered as positive.

3.1.1 Tokenizing by n-gram

We can use the `unnest_tokens()` for two words instead of one.

```
df_bigrams <- df_id %>%
  unnest_tokens(bigrams, text, token = "ngrams", n = 2)
df_bigrams
```

```
## # A tibble: 223,998 x 5
##       ID company date       Link                                    bigrams
##    <int> <chr>   <date>     <chr>                                   <chr>
##  1     1 Nvidia  2018-08-29 http://www.marketwatch.com/story/he~ what a
##  2     1 Nvidia  2018-08-29 http://www.marketwatch.com/story/he~ a differ~
##  3     1 Nvidia  2018-08-29 http://www.marketwatch.com/story/he~ differen~
##  4     1 Nvidia  2018-08-29 http://www.marketwatch.com/story/he~ a couple
##  5     1 Nvidia  2018-08-29 http://www.marketwatch.com/story/he~ couple of
##  6     1 Nvidia  2018-08-29 http://www.marketwatch.com/story/he~ of months
##  7     1 Nvidia  2018-08-29 http://www.marketwatch.com/story/he~ months c~
##  8     1 Nvidia  2018-08-29 http://www.marketwatch.com/story/he~ can make
##  9     1 Nvidia  2018-08-29 http://www.marketwatch.com/story/he~ make the
## 10     1 Nvidia  2018-08-29 http://www.marketwatch.com/story/he~ the u.s
## # ... with 223,988 more rows
```

Now lets separate them into own columns

```
bigrams_separated <- df_bigrams %>%
  separate(bigrams, c("word1", "word2"), sep = " ")
```

Lets check the negated words

```
# create a list of negation words

negation_words <- c("no", "can't", "not", "never", "won't")

negated_words <- bigrams_separated %>%
  filter(word1 %in% negation_words) %>%
  inner_join(loughran, by = c(word2 = "word"))
```

Correcting for negation

```
negated_words <- bigrams_separated %>%
  filter(word1 %in% negation_words) %>%
  inner_join(loughran, by = c(word2 = "word"))

# count the number of positive and negative words which need to be reversed

negated_counted <- negated_words %>%
  group_by(ID, company) %>%
  count(sentiment)%>%
  ungroup() %>%
  filter(sentiment %in% c("positive", "negative")) %>%
  spread(sentiment, n) %>%
  replace_na(list(positive = 0, positive = "unknown")) %>%
```

```
  replace_na(list(negative = 0, negative = "unknown"))

# create a net score

negated_net <- negated_counted %>%
  mutate(net2 = negative - positive) %>%
  select(-c(negative, positive))

# now to correct the original score we need to add net2 to positive column
# subtract net2 from negative column

# this was the original score
loughran_pos_neg
```

```
## # A tibble: 392 x 5
##       ID company  positive negative    net
##    <int> <chr>       <dbl>    <dbl>  <dbl>
##  1     1 Nvidia          5       11 -0.375
##  2     2 Walmart        19       39 -0.345
##  3     3 Tesla          14       14  0
##  4     4 Apple           1        1  0
##  5     5 Tesla           2        2  0
##  6     6 Amazon         10        7  0.176
##  7     7 Google          3       17 -0.7
##  8     8 Google          5        9 -0.286
##  9     9 Facebook        3       14 -0.647
## 10    10 Netflix         9        4  0.385
## # ... with 382 more rows
```

```
# join the two data sets

loughran_pos_neg2 <- loughran_pos_neg %>%
  left_join(negated_net, by = c("ID", "company")) %>%
  replace_na(list(net2 = 0, net2 = "unknown")) %>%
  mutate(positive2 = positive + net2) %>%
  mutate(negative2 = negative - net2) %>%
  select(-c(positive, negative, net2))

# Now calculate net score normalized

loughran_negated <- loughran_pos_neg2 %>%
  mutate(net2 = (positive2 -negative2)/(positive2+negative2))

loughran_negated
```

```
## # A tibble: 392 x 6
##       ID company     net positive2 negative2    net2
##    <int> <chr>     <dbl>     <dbl>     <dbl>   <dbl>
##  1     1 Nvidia   -0.375         6        10 -0.25
##  2     2 Walmart  -0.345        19        39 -0.345
##  3     3 Tesla     0            15        13  0.0714
##  4     4 Apple     0             1         1  0
##  5     5 Tesla     0             2         2  0
##  6     6 Amazon    0.176        10         7  0.176
```

```
## 7      7 Google   -0.7          3        17 -0.7
## 8      8 Google   -0.286        4        10 -0.429
## 9      9 Facebook -0.647        3        14 -0.647
## 10    10 Netflix   0.385        9         4  0.385
## # ... with 382 more rows
```

```
loughran2 <- loughran_negated %>%
  group_by(company) %>%
  summarise(net_loughran2 = mean(net2, na.rm = TRUE)) %>%
  arrange(desc(net_loughran2))
loughran2
```

```
## # A tibble: 45 x 2
##    company              net_loughran2
##    <chr>                        <dbl>
##  1 Berkshire Hathaway           1
##  2 Occidental Petroleum         1
##  3 Microsoft                    0.678
##  4 Oracle                       0.667
##  5 EOG Resources                0.478
##  6 Celgene                      0.375
##  7 Qualcomm                     0.294
##  8 Bank of America              0.2
##  9 Home Depot                   0.143
## 10 Walmart                      0.141
## # ... with 35 more rows
```

```
#library(xlsx)
#library(openxlsx)
#write.xlsx(loughran2, 'company_score.xlsx')

#write.xlsx(loughran2, "company_score.xlsx")
```

4 Conclusion and future works

We have presented a word count approach towards sentiment analysis. In the first part positive and negative words are added up for an overall sense of the sentiment. In the second part we deal we negation and recalculate our score to correct for negation. In future works we will updating the `lexicon` to consider newer and more up to date terminalogy about the stock market.