Natural Language Processing and Sentiment Analysis

Dinesh R Poddaturi

1 Introduction

The purpose of this project is to perform Natural Language Processing and Sentiment Analysis on the most discussed topic in the current economy. Specifically, in this project, I use Twitter data which I access through my API to analyze and plot the general sentiment. In particular, sentiment regarding recession is analyzed and discussed.

The following reproducible code can be easily edited to analyze the sentiments about other keywords.

2 Natural Language Processing

```
install.packages("librarian")

librarian::shelf(tidyverse, reshape2, readxl, data.table, nleqslv, BB, Metrics, ggthemes
                 twitteR, ROAuth, hms, lubridate, tidytext, tm, wordcloud, igraph, glue,
                 rtweet, stringr, ggeasy, plotly, janeaustenr, widyr, textdata)

if (!requireNamespace("httpuv", quietly = TRUE)) {
  install.packages("httpuv")
}


library(openssl)
library(httpuv)
library(rtweet)
```

I use the following credentials to get access to the Twitter data. One must have these secure credentials to get Twitter data.

```
#Note: Replace below with your credentials following above reference
api_key <- "yourAPIKey"
api_secret_key <- "yourAPISecretKey"
access_token <- "yourAccessToken"
access_token_secret <- "yourAccessTokenSecret"


### The following code setup direct authentication to access twitter
setup_twitter_oauth(api_key, api_secret_key, access_token, access_token_secret)
```

The following function cleans the data from images, numbers, control functions, and any other gibberish that cannot be used for analysis.

```
## pre-processing text: This function is to clean the text of the tweet of
## any symbols, and other unnecessary items that are non-readable
cleanText <- function(x){
  # convert to lower case
  x = tolower(x)
  # remove rt
  x = gsub("rt", "", x)
  # remove at
  x = gsub("@\\w+", "", x)
  # remove punctuation
  x = gsub("[[:punct:]]", "", x)
  # remove numbers
  x = gsub("[[:digit:]]", "", x)
  # remove links http
  x = gsub("http\\w+", "", x)
  # remove tabs
  x = gsub("[ |\t]{2,}", "", x)
  # remove blank spaces at the beginning
  x = gsub("^ ", "", x)
  # remove blank spaces at the end
  x = gsub(" $", "", x)
  # some other cleaning text
  x = gsub('https://','',x)
  x = gsub('http://','',x)
  x = gsub('[^[:graph:]]', ' ',x)
  x = gsub('[[:punct:]]', '', x)
  x = gsub('[[:cntrl:]]', '', x)
  x = gsub('\\d+', '', x)
  x = str_replace_all(x,"[^[:graph:]]", " ")
  return(x)
}
```

Here I specify a keyword and use it in the following snippet to get tweets mentioning the keyword with a hash tag.

```
search_term <- "#recession"
by <- 'hour'


tweets <- searchTwitter(searchString = search_term, n=100000, lang="en")


tweetsDF <- twListToDF(tweets)


names(tweetsDF)


 [1] "text"          "favorited"     "favoriteCount" "replyToSN"
 [5] "created"       "truncated"     "replyToSID"    "id"
 [9] "replyToUID"    "statusSource"  "screenName"    "retweetCount"
[13] "isRetweet"     "retweeted"     "longitude"     "latitude"
```

Here I extract the information that will be used for analysis. First I filter the tweets by checking whether the tweet is a re-tweet. I want to analyze the original tweets, so I remove all the re-tweets from my data frame [1]. After filtering the tweets I extract the time stamp, the text, and the hashtags of each individual tweet. The text of the tweets is then cleaned using the *cleanText* function defined above.

```
tweetsDF <- tweetsDF %>% filter(isRetweet == "FALSE")


tweetsSubSet <- tweets %>% select(created, text)


# To ignore graphical Parameters and to avoid input errors
tweetsSubSet$text <- str_replace_all(tweetsSubSet$text,"[^[:graph:]]", " ")

```

[1] This is solely my personal preference. I prefer original content. Re-tweets simply are simply duplicates of the original content.

```
tweetsSubSet$text <- cleanText(tweetsSubSet$text)
```

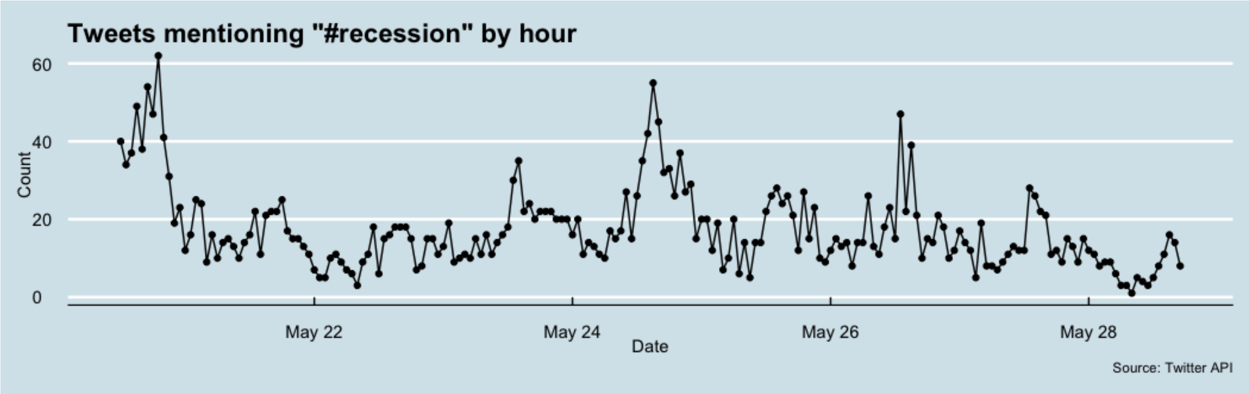
Now I round the tweets by the hour to plot the tweets mentioning the keyword by the hour.

```
tweetsSubSet <- tweetsSubSet %>%
  mutate(Created_At_Round = created %>% round(units = 'hours') %>% as.POSIXct())

tweetsSubSet %>% pull(created) %>% min()

tweetsSubSet %>% pull(created) %>% max()
```

The following code snippet plots the number of tweets mentioning our search keyword.

```
tweetHourPlot <- tweetsSubSet %>%
  count(Created_At_Round) %>%
  ggplot(mapping = aes(x = Created_At_Round, y = n)) +
  theme_economist() +
  geom_point() +
  geom_line() +
  xlab(label = 'Date') +
  ylab(label = 'Count') +
  labs(title = paste0('Tweets mentioning "',
                         search_term,'" by hour') , caption = 'Source: Twitter API')
tweetHourPlot
```

Figure 1: Tweets by Hour



The above plot visualizes the tweets mentioning the keyword for each day by the hour. From the limited data I have I observe that the tweets mentioning recession are decreasing over time. However, it appears that the economy (i.e., people in the economy) tends to tweet about recession between noon and night every day [2].

2.1 Sentiment Analysis

For sentiment analysis, I use pre-existing positive and negative words from the literature [3]. I further add some more words to the existing words.

```
### Retrieving positive and negative words
positive <- scan('OpinionLexiconEnglish/positive-words.txt', what = 'character', comment
negative <- scan('OpinionLexiconEnglish/negative-words.txt', what = 'character', comment

# add our list of words below as you wish if missing in above read lists
positiveWords <- c(positive,'upgrade','Congrats','prizes','prize','thanks','thnx',
                   'Grt','gr8','plz','trending','recovering','brainstorm','leader')

negativeWords <- c(negative,'wtf','wait','waiting','epicfail','Fight','fighting',
```

[2] The peaks in the above plot are indicative of those time-stamps

[3] For more information about lexicon please visit https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

```
              'arrest','no','not')
```

The following function matches the text with pre-determined positive and negative words and returns sentiment score.

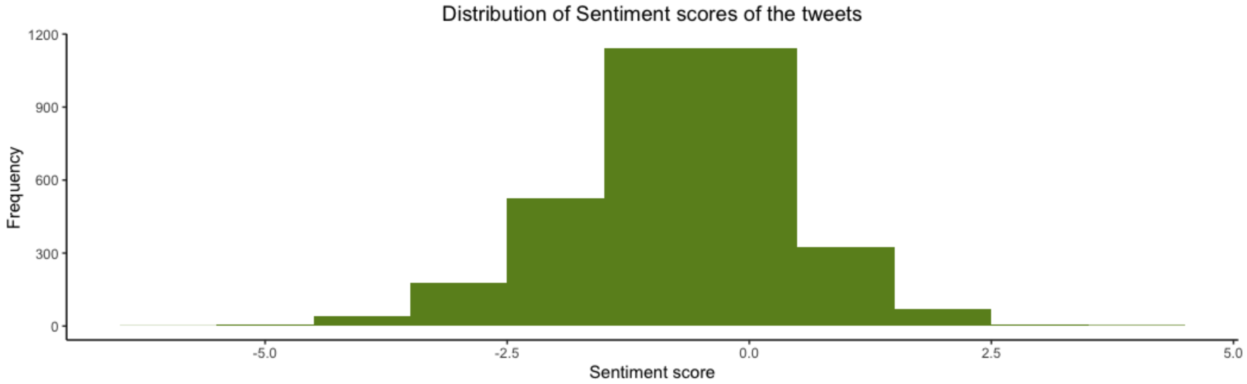
```
sentimentScore <- function(sentences, pos.words, neg.words){
  # require(plyr)
  # require(stringr)

  # we are giving vector of sentences as input.
  # plyr will handle a list or a vector as an "l" for us
  # we want a simple array of scores back, so we use "l" + "a" + "ply" = laply:
  # sentences = tweetsSubSet$text
  # pos.words = positiveWords
  # neg.words = negativeWords

  scores <- lapply(sentences, function(sentence, pos.words, neg.words) {

    # clean up sentences with R's regex-driven global substitute, gsub() function:
    sentence <- gsub('https://','',sentence)
    sentence <- gsub('http://','',sentence)
    sentence <- gsub('[^[:graph:]]', ' ',sentence)
    sentence <- gsub('[[:punct:]]', '', sentence)
    sentence <- gsub('[[:cntrl:]]', '', sentence)
    sentence <- gsub('\\d+', '', sentence)
    sentence <- str_replace_all(sentence,"[^[:graph:]]", " ")
    # and convert to lower case:
    sentence <- tolower(sentence)
    # split into words. str_split is in the stringr package
    word.list <- str_split(sentence, '\\s+')
    # sometimes a list() is one level of hierarchy too much
    words <- unlist(word.list)

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

    # match() returns the position of the matched term or NA
    # we just want a TRUE/FALSE:
    pos.matches <- !is.na(pos.matches)
    neg.matches <- !is.na(neg.matches)

    # TRUE/FALSE will be treated as 1/0 by sum():
    score <- sum(pos.matches) - sum(neg.matches)

    return(score)
  }, pos.words, neg.words)

  scores <- unlist(scores)
  sentences <- unlist(sentences)

  scores.df <- data.frame(score=scores, text=sentences)

  return(scores.df)
```

```
}
```

Here I get the sentiment score from each text and plot the distribution of the sentiment scores in a simple histogram plot.

```
analysisSentiment <- sentimentScore(sentences = tweetsSubSet$text, pos.words = positiveW
                                    neg.words = negativeWords)


analysisSentiment %>%
  ggplot(aes(x=score)) +
  geom_histogram(binwidth = 1, fill = "olivedrab")+
  ylab("Frequency") +
  xlab("Sentiment score") +
  ggtitle("Distribution of Sentiment scores of the tweets") +
  theme_classic() +
  ggeasy::easy_center_title()
```

Figure 2: Distribution of Sentiment scores of the tweets

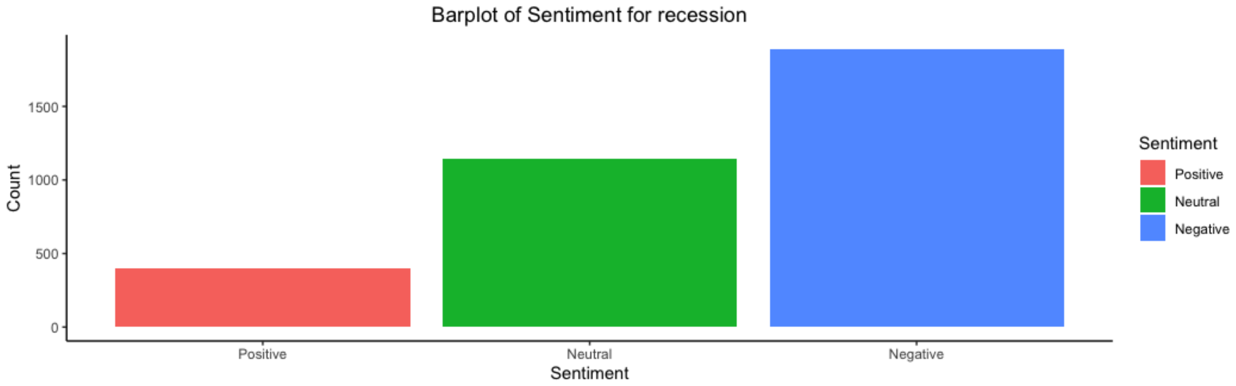


From the above distribution plot, the overall sentiment of the economy regarding the keyword *recession* is skewed left and negative. However, there is a neutral and a small positive sentiment with *recession* but with a thin tail. This is due to the fact that the economy must go through occasional recessions to keep the efficient players in the market and ask the inefficient players to exit from the market.

In the following, I simply plot each sentiment *positive, neutral, negative* in a single plot to further back up the conclusion made from the above distribution plot.

```r
neutral <- length(which(analysisSentiment$score == 0))
positive <- length(which(analysisSentiment$score > 0))
negative <- length(which(analysisSentiment$score < 0))
Sentiment <- c("Positive","Neutral","Negative")
Count <- c(positive,neutral,negative)
output <- data.frame(Sentiment,Count)
output$Sentiment<-factor(output$Sentiment,levels=Sentiment)
ggplot(output, aes(x=Sentiment,y=Count))+
  geom_bar(stat = "identity", aes(fill = Sentiment))+
  ggtitle("Barplot of Sentiment for recession")+
  theme_classic() +
  ggeasy::easy_center_title()
```

Figure 3: Sentiment Bar Plot

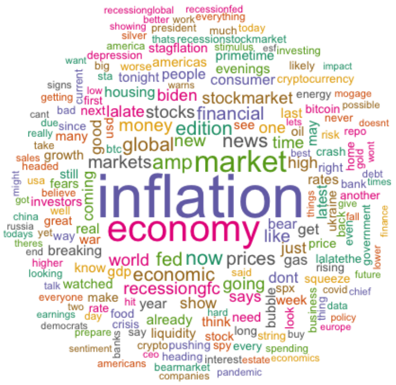


The bar plot above further supports the claim, there is a strong negative sentiment toward recession in the economy. Followed by neutral and positive sentiment. Notice that there is a positive sentiment toward recession. This is crucial in any sentiment analysis. A strong sentiment towards something doesn’t necessarily mean there is no positive sentiment in the economy.

Now, I plot a simple word cloud of the tweets mentioning our keyword. I remove the keyword and other *stop words* from the tweets to observe and plot the cloud.

```r
textCorpus <- Corpus(VectorSource(tweetsSubSet$text))

textCorpus <- tm_map(textCorpus, content_transformer(tolower))

textCorpus <- tm_map(textCorpus, function(x)removeWords(x,stopwords("english")))

textCorpus <- tm_map(textCorpus, removeWords, c("recession", "will", "can"))

termDocMat <- TermDocumentMatrix(textCorpus)

termDocMat <- as.matrix(termDocMat)

termDocMat <- sort(rowSums(termDocMat), decreasing = TRUE)

termDocMat <- data.frame(word = names(termDocMat), freq = tdm)

set.seed(123)

wordcloud(textCorpus, min.freq = 5, max.words = 200,

          colors=brewer.pal(8, "Dark2"), random.color = T, random.order = F)
```

Figure 4: Word Cloud

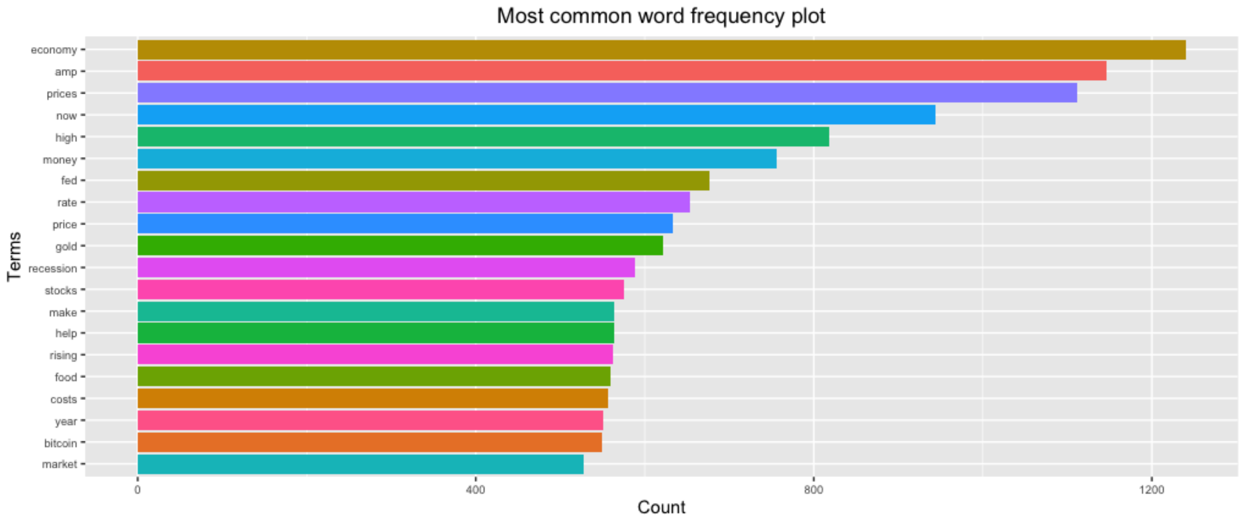


The above word cloud illustrates the most common words associated with the keyword *recession* in the original tweets. They are inflation, economy, market, financial, fed, and global.

Below I visualize the most common words occurring with the keyword.

```
ggplot(tdm[1:20,], aes(x=reorder(word, freq), y=freq, fill = word)) +
  geom_bar(stat="identity") +
  xlab("Terms") +
  ylab("Count") +
  coord_flip() +
  theme(axis.text=element_text(size=7)) +
  ggtitle('Most common word frequency plot') +
  ggeasy::easy_center_title() +
  theme(legend.position="none")
```

Figure 5: Most common word frequency plot



3 Conclusion

Various natural language processing techniques are used to analyze and visualize the current sentiment of *recession* in the economy. The data are gathered from Twitter using API and advanced techniques are used to clean and analyze the sentiment. Although trivial, the data suggest that the overall sentiment about *recession* in the economy is negative. However, the positive sentiment towards *recession* is greater than zero, indicating that the economy views recession in a positive way as well. A word cloud is also plotted to showcase the most used words in the data along with the keyword recession. To elaborate further, a plot with the most frequently occurring common words with the keyword is also plotted.

The code provided in the report is reproducible and can be used by anyone with minor changes.