**A Social Media Analysis of COVID-19: Sentiment, Influence, and more on Twitter**

By Joseph D’Ambrosio

Introduction: Topic

[write an introduction to your topic that establishes context; likely a paragraph or two and less than 2 pages]

Beginning in 2019, documented on December 12. “A cluster of patients in Wuhan, Hubei Providence, China begin to experience shortness of breath and fever.” (CDC, COVID 19 Timeline) On January 20th, 2020 the CDC confirms the first U.S Laboratory confirmed case of COVID-19. On March 15th, 2020 U.S States begin to shut down to prevent the spread of COVID-19. Businesses, Schools, and Government Agencies begin to close and limit services. The pandemic outbreak lasts for 2+ years, and affects the daily lives of all, disrupting nearly every industry worldwide.

Introduction: Dataset

The dataset used in this analysis consists of 59,203 tweets collected via twitter from 3/4/2020 to 3/29/2021. However, for this analysis, a pseudorandom seed of 304 was used, and the dataset was reduced to 1500 tweets for analysis. Tweets were archived via the DMI Twitter Capturing and Analysis Toolset (DMI-TCAT) located at Kutztown University, and managed by Dr. Keith Massie.

Stage One: Finding average sentiment

At the start of this project, the DMI-TCAT provided dataset was read via the *fread* command and stored into the *SUPLdf*. This allowed R to have a native dataset to work with for the remainder of the project. Initial average sentiment analysis was achieved by using the *get\_sentences* function on the *SUPLdf* dataset. These values were then stored into the S1prep\_df data frame which allowed for the “*sentiment*” function from sentimentr to be ran and values to be stored in the *S1sent\_df* data frame. To finish this stage of analysis, both the mean and range functions were performed on the *sentiment* column of the S*1sent\_df* data frame.

1. file.choose()
2. SUPLdf <- fread("/Users/josephdambrosio/Dropbox/AA-ACADEMICS/AA-Kutztown/2021/Spring/SMS Analytics/Final Project/SMS314 Final Project/DAMBROSIOSEEDCOV.csv")
3. S1prep\_df <- get\_sentences(as.vector(SUPLdf$text))
4. S1sent\_df <- sentiment(S1prep\_df)
5. mean(S1sent\_df$sentiment)
6. range(S1sent\_df$sentiment)

Stage Two: Generating an influence score

For the purpose of this analysis, a basic “influence score” was created for each user This score was calculated by dividing the posting user’s follower count (*from\_user\_followercount*) by the number of twitter “friends” the user had (*from\_user\_friendcount*). Where a user’s followers were other users that strictly was following the user, and “friends” were followers of the posting user who the posting user also followed back. The initial data frame *SUPLdf* was stored into a new *S2infl\_df* data frame which the influence score was also added into through use of dplyr’s mutate command.

|  |
| --- |
| 1. S2infl\_df <- SUPLdf %>% 2. mutate(influence = from\_user\_followercount / from\_user\_friendcount) |

Stage Three: Individualized sentiment

With the basic sentiment analysis in stage one being completed on a by sentence basis, it is prudent to merge those values into their respective original tweets rather than just by sentence within the entire dataset. This allows for sentiment per each respective tweet to be analyzed correctly. This was accomplished by storing the values from the *S1sent\_df* into the *S3recmb\_df* and then grouping by the *element\_id* value of the data frame. Then the *summarize* function is called to summarize by the mean score of the sentiment for the tweet.

1. S3recmb\_df <- S1sent\_df %>%
2. group\_by(element\_id) %>%
3. summarize(mean(sentiment))

Stage Four: Expanding the data frame to have both influence and sentiment

Through the use of the data.frame command the *S2infl\_df* data frame from stage 2 (influence score) and the column *mean(sentiment)* in the *S3recmb\_df* data frame into a new data frame called *S4newest\_df*.

1. S4newest\_df <- data.frame(S2infl\_df, S3recmb\_df$'mean(sentiment)')

Stage Five: Renaming and improving the dataframe

[Explain why changing a column name may be useful; how did you change it?]

With the new scores for each individual piece of content being calculated and stored in data frames, it was time to clean up the data frame and rename the data frame to something easier to use for the remainder of the project. To begin, the *S4newest\_df* was stored into a new dataframe called *S5clean\_df* and then the *filter* command was used to filter out only tweets by users who had more than 0 friends and 0 followers. This *S5clean\_df* was then stored to a new dataframe called *S5finalClean\_df* which than through the *mutate* function, stored the mean sentiment value in the *S3recmb\_df* into a column called *sentiment\_score* and then removed the original *mean.sentiment* column.

1. S5clean\_df <- S4newest\_df %>%
2. filter(from\_user\_friendcount > 0, from\_user\_followercount > 0)
3. S5finalClean\_df <- S5clean\_df %>%
4. mutate(sentiment\_score = S3recmb\_df..mean.sentiment..) %>%
5. select(-c(S3recmb\_df..mean.sentiment..))

Stage Six: Working with time

DMI\_TCAT’s CSV output of tweets stores the value that the tweet was created as a factor. This is an issue because visual plots to show data over time can not be made with this factor. To combat this, I used the as.Date function to format the date correctly in a way that can be used in the further stages of this project. This is accomplished by bringing in the *created\_at* column of the *S5finalClean\_df* data frame and storing it to a data frame called *x6*. Another data frame is created with the name of *x6\_2* and is filled with data outputted by the *as.Date(x6)* function. To finalize this stage of cleaning up time, a new dataframe is created called *S6\_FinalDF*  which has the *S5finalClean\_df* imported. Through the use of the mutate command, a new column called Date is created with the values of the *x6\_2* dataframe and the *created\_at* dataframe is removed.

1. x6 <- S5finalClean\_df$created\_at
2. x6\_2 <- as.Date(x6)
3. S6\_FinalDF <- S5finalClean\_df %>%
4. mutate(Date = x6\_2) %>%
5. select(-c(created\_at))

Stage Seven: Tweets across time

The *table* function can now be used to store the values from the *Date* column in the *S6\_FinalDF* data frame to a new data frame called BPtbl. This allows the *barplot* function to run successfully and output the following visualization of the number of tweets over time.

1. BPtbl <- table(S6\_FinalDF$Date)
2. barplot(BPtbl, col='blue', main= 'Barplot of Tweets about COVID-19')

Image 1: Tweets across time

Chart

Description automatically generated

Stage Eight: Examining changes across time

We’ll analyze the influence of the users tweeting over time to determine relevance. This is done by bringing in the *S6\_FinalDF* into a new data frame called *S8splitINFL\_df* and then using *group\_by* to group by values in the *Date* column. Next the data frame is summarized by the mean values of the *influence* column. The same is done for the *sentiment\_score* with values being stored in the *S8splitSNTMNT\_df*.

1. S8splitINFL\_df <- S6\_FinalDF %>%
2. group\_by(Date) %>%
3. summarize(mean(influence))
4. S8splitSNTMNT\_df <- S6\_FinalDF %>%
5. group\_by(Date) %>%
6. summarize(mean(sentiment\_score))

Image 2: Influence on each day

Table

Description automatically generated

One can also look at sentiment in the same way.

Image 3: Sentiment on each day

Table

Description automatically generated

Stage Nine: Visualizing changes over time

One can visualize the outputs noted in images 2 and 3. Plotting each output aids in interpretation. One finds that the influence score starts high, drops low for the month of May, increases towards July, where it again drops however over a less steep rate then May. Sentiment matches the trend pattern as well. It appears that there is high influence as a higher sentiment score is present.

1. plot(S8splitINFL\_df, main = "Change over time: Influence", type = "b")
2. plot(S8splitSNTMNT\_df, main = "Change over time: Sentiment", type = "b")

Image 4: Influence across time

Chart, line chart

Description automatically generated

Additionally, we can examine visually the sentiment for each day.

Image 5: Sentiment across time

Chart, line chart

Description automatically generated

Stage Ten: Influencers and their attributes

The top influencer in my dataset was @algerie360. This user no longer exists on Twitter, so no data could be collected on demographics.

2/5 was 24/7 Italian news organization @repubblica.

Joining Twitter in January 2009, @repubblica has 3.3M followers at the time of writing and tweets about every 15/30 minutes.

3/5 was an account that “retweets all tweets talking about Senegal”.

Joining Twitter in March 2018, @kebetubot has 25.9K followers at the time of writing and tweets about every 30 minutes.

4/5 was “Senegalese Television Broadcasting” RTS Senegal.

Joining Twitter in August 2012, @RTS1\_Senegal has 187.9K followers at the time of writing and tweets in 5+ hour increments.

5/5 was French television personality William Midi.

Joining twitter in August 2012, @WilliamAMidi has 71.5k followers at the time of writing and tweets in hour increments.

Stage Eleven: Top positive and negative sentiment

[Explain how you found the top positive and negative user for sentiment. Does the sentiment make sense? Did the sentimentr package make any error in coding their sentiment? Etc.]

1. round(mean(S6\_FinalDF$sentiment), 3)
2. round(mean(S6\_FinalDF$influence), 3)

Stage Twelve: Looking for relationships

One way to look for relationships between variables is to run a correlation to see the strength and direction of their relationship. Given that the data frame has several numeric values, it would be inefficient take each individually and compare individually. To save time, we will run a correlation plot matrix, which will allow us to compare all numeric values to each other simultaneously. To begin, we condense the original data frame called *S6\_FinalDF* into a data frame called *corrplot\_df* that has removed all columns that are not numeric. We, then, create a correlation plot matrix, and its outcome can be seen in Image 6.

Image 6: Correlation plot matrix for *corrplot\_df*

Table

Description automatically generated with medium confidence

You can see from the correlation plot that the users that have a higher number of followers correlate to a higher influence score. This makes sense as our mathematical equation for calculating influence is dependent on the number of followers that a user has.

Stage Thirteen: Advanced sentiment of top influencers

When looking at the analysis of the top influencers, sentiment is both **positive** and **trust.** I believe that this is because of the time from which my dataset focuses on was around the time that COVID-19 restrictions were beginning to ease up around the summer season and COVID case numbers were beginning to lower.

1. S12topINFLR\_df <- S6\_FinalDF %>%
2. filter(influence > 2461)
3. text <- as.vector(S12topINFLR\_df$text)
4. text\_df <- data\_frame(line = 1:5, text = text)
5. text\_df2 <- text\_df %>% unnest\_tokens(word, text)
6. text\_df3 <- text\_df2 %>% anti\_join(stop\_words) %>% count(word, sort = TRUE)
7. text\_df\_NRC <- text\_df3 %>% inner\_join(get\_sentiments("nrc"))
8. text\_df\_NRC
9. mean(text\_df\_NRC$value)

Stage Fourteen: Comparing bigrams and trigrams

One can compare the bigrams and trigrams of *S5finalClean\_df* to the bigrams and trigrams found in the full, original data frame for coronavirus with over 59,000 observations. Table 1 below shows the comparison of bigrams. Table 2 below highlights the differences in trigrams.

Table 1: Bigram comparison

|  |  |
| --- | --- |
| Top 5 ‘meaningful’ bigrams for *S5finalClean\_df* | Top 5 ‘meaningful’ bigrams found in full covid dataset |
| rt barackobama | rt bbcscotlandnews |
| barackobama protect | bbcscotlandnews scotland’s |
| protect yourself | scotland’s test |
| yourself and | test and |
| and your | and protect |

Table 2: Trigram comparison

|  |  |
| --- | --- |
| Top 5 ‘meaningful’ bigrams for *S5finalClean\_df* | Top 5 ‘meaningful’ bigrams found in full covid dataset |
| rt barackobama protect | rt bbcscotlandnews scotland’s |
| barackobama protect yourself | bbcscotlandnews scotland’s test |
| protect yourself and | scotland’s test and |
| yourself and your | test and protect |
| and your community | and protect system |

1. S13bigram\_df <- S5finalClean\_df %>%
2. filter(lang == "en") %>%
3. unnest\_tokens(bigram, text, token = "ngrams", n = 2)
4. S13trigram\_df <- S5finalClean\_df %>%
5. filter(lang == "en") %>%
6. unnest\_tokens(trigram, text, token = "ngrams", n = 3)

Stage Fifteen: Making a ngram function

One way that a researcher or analyst can save energy and make work more efficient is by creating a function. The below images demonstrate how one would create a function that always located the 5-gram within any dataset where the tweets were in a column called text. The first image is a function that is unfiltered and produces all 5-word combinations. The second image is a function that automatically filters the original data frame as to isolate English-speaking users only.

Image 6: The unfiltered function for a 5-word ngram

1. fivegram <- function(x) {
2. library(dplyr)
3. library(tidytext)
4. library(tidyr)
5. bigram\_number <- x %>%
6. unnest\_tokens(bigram, text, token = "ngrams", n = 5)
7. z <- table(bigram\_number$bigram)
8. z2 <- as.data.frame(z)
9. View(z2)}

Image 7: The function for a 5-word ngram that filters for English

1. fivegram\_eng <- function(x) {
2. library(dplyr)
3. library(tidytext)
4. library(tidyr)
5. bigram\_number <- x %>%
6. filter(lang == "en") %>%
7. unnest\_tokens(bigram, text, token = "ngrams", n = 5)
8. z <- table(bigram\_number$bigram)
9. z2 <- as.data.frame(z)
10. View(z2)}

Conclusion

In summary, this project analyzed 1500 tweets that were collected via the DMI-TCAT software. Tweets were then analyzed for relevance, influence, sentiment, and top ngrams. The findings were documented in multiple various stages. The most interesting finding to me was that the top influencer account from my dataset was deleted and no longer available. Second to that was the fact that my 2nd highest influencer was an Italian news agency. I would have predicted a news agency from a different region to have been top influence.

Bibliography

Borra, E., & Rieder, B. (2014, May 19). *Programmed method: Developing a toolset for capturing and analyzing tweets*. Aslib Journal of Information Management. Retrieved March 28, 2022, from https://www.emerald.com/insight/content/doi/10.1108/AJIM-09-2013-0094/full/html

Rinker, T. W. (2021). sentimentr: Calculate Text Polarity Sentiment version 2.9.0. <https://github.com/trinker/sentimentr>

Hadley Wickham, Romain François, Lionel Henry and Kirill Müller (2022). dplyr: A Grammar of Data Manipulation. R package version 1.0.8. <https://CRAN.R-project.org/package=dplyr>

Matt Dowle and Arun Srinivasan (2021). data.table: Extension of `data.frame`. R package version 1.14.2. <https://CRAN.R-project.org/package=data.table>

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

Taiyun Wei and Viliam Simko (2021). R package 'corrplot': Visualization of a Correlation Matrix (Version 0.92). Available from <https://github.com/taiyun/corrplot>

Silge J, Robinson D (2016). “tidytext: Text Mining and Analysis Using Tidy Data Principles in R.” \_JOSS\_, \*1\*(3). doi: 10.21105/joss.00037 (URL: <https://doi.org/10.21105/joss.00037>), <http://dx.doi.org/10.21105/joss.00037>

Hadley Wickham and Maximilian Girlich (2022). tidyr: Tidy Messy Data. R package version 1.2.0. <https://CRAN.R-project.org/package=tidyr>

Centers for Disease Control and Prevention. (2022, January 5). *CDC Museum Covid-19 Timeline*. Centers for Disease Control and Prevention. Retrieved April 13, 2022, from https://www.cdc.gov/museum/timeline/covid19.html

The NRC dataset was published in Saif M. Mohammad and Peter Turney. (2013), ``Crowdsourcing a Word-Emotion Association Lexicon.'' Computational Intelligence, 29(3): 436-465.