

Misinformation Represented in Tweets Throughout the COVID-19 Pandemic

Muhan (Elsie) Zhu, Ron Varshavsky

Tuesday, December 14, 2021

Problem Description and Research Question

Throughout the COVID-19 Pandemic, there has been an influx of rumours and misinformation related to the virus and how different countries responded. The World Health Organization has dubbed this an “Infodemic” alongside the current worldwide pandemic. This widespread misinformation has been catastrophic in many areas. A prime example is the “cure” for COVID-19 which circulated near the beginning of the pandemic. This “cure” involved injecting bleach into your arms (van der Linden et al., 2020). Such rumours are not only false, but they are dangerous. In fact, this false fact was so widespread that the reported cases of accidental poisoning due to bleach in Americans doubled in March and April of 2020 (Bowden, 2020). The effects of a singular rumour were so detrimental that it compromised the lives of many people; such widespread misinformation is quite literally deadly. Such misinformation has changed over time, however it has never been fully eradicated from word of mouth, such as the rumour that taking a potentially dangerous dose of Ivermectin could reduce COVID-19 symptoms (U.S. Food and Drug Administration, 2021). This rumour, like the previous one mentioned, can put numerous lives at risk. We researched how public conversation surrounding the virus and misinformation have been impacted over time during the pandemic, and if these time frames coincided with any major milestones within the pandemic (i.e lockdown, vaccine distribution). We found evidence relating to the periods of time within the pandemic when influxes of misinformation took place. This allowed us to hypothesize the mistakes that we, as a society, made during the course of the pandemic which caused this spread of false, dangerous information. In order to quantify our research, we asked the question: **In what way, quantitatively, has misinformation in discourse on Twitter surrounding the COVID-19 pandemic changed over the course of the pandemic?**

Dataset Descriptions

The source for our myths is the Newsguard Covid Myths website. Here, we simply scraped each myth from their site, and removed the characters “MYTH: ” from the beginning of each scraped line. The NewsGuard website can be found here: <https://www.newsguardtech.com/special-reports/coronavirus-misinformation-tracking-center/>.

Another source we used for our myths was creating a separate file where we would input known misinformation that was skipped over in the NewsGuard website because they were removed as the site updated. We then combined both this source and the previous source into one file for a single dataset of “COVID-19 myths”, which we used to detect misinformation.

To find the tweets relating to COVID-19, we used the dataset “COVID-19 Twitter Dataset with Latent Topics, Sentiments and Emotions Attributes”. It can be found here: <https://www.openicpsr.org/openicpsr/project/120321/version/V11/view;jsessionid=EB090760F70409E47B82FC0137D7B1A1>. This csv dataset contains 198 million Twitter tweet ID’s, but does not contain the tweets themselves. Evidently, this would be very time consuming to compute on, so we took a smaller sample size by only using 1 in 30 000 lines of the dataset, resulting in about 6600 tweet id’s to compute on. From here, we had to collect our own tweet dataset, as the dataset provided only contains tweet ID’s, due to Twitter’s API policy. As such, we used the Twitter API to get the text of each tweet by its ID. We then gathered each of these tweets into a separate file and used this as a dataset to compute on.

Finally, we compiled a list of common negation words from Grammarly into a text file, to use for the false-positive portion of our misinformation detection algorithm. The data we scraped can be found here: <https://www.grammarly.com/blog/negatives/>.

Computational Overview

Data Filtering

After downloading the Twitter dataset, we immediately isolated just the ID's from the dataset, as it contained other information not relevant to our project. We did this in `pull_tweets.py` inside the function `isolate_ids()`. Here, it opens the full dataset csv file, and then splits the csv by its commas and writes the `twitter_id` columns into a new file generated called `twitter_ids` with no file extension, in the directory `./data/twitter_ids`. From here, we used `pull_direct_tweets()` in the same file to use the IDs we isolated to request the Twitter API and actually get a response containing the text of the tweet. Unfortunately, due to the structure of Twitter's API, there is a rate limit of 900 requests per 15 minutes, so we had to run this function multiple times, storing data in multiple files, and then combining it in the next function we used, `compile_into_file()`, which took all of the files we first generated, and wrote them into a single file named `filtered_twitter_ids_compiled`, with no file extension, stored in the directory `./data/filtered_twitter_ids_compiled`. From here, we ran `sort_by_month` to sort the compiled file into multiple files separated by the month and year that each tweet was posted. This function accomplished this by creating a list of open files for each month/year combination in the dataset, and then reading through each line of the compiled Twitter API responses, and then running a helper function `isolate_text_write_to_file`, which isolated the text, and based on what year and month combination the Tweet was posted, wrote the tweet text into that file. After all of this was completed, we successfully had thousands of tweets each sorted into a file based on the month and year they were written, waiting to be computed on.

Data Computing

Once the data was filtered, the data was computed on. This was done in the file `compute_similarity.py`. In this module, a function named `output_myths_count_into_file` is called, with a filename as its parameters. This filename is a file of combined myths that were assembled through web-scraping. Essentially, this file calls a helper function and then writes the returned value from the helper function into a file. The helper function is named `get_myths_for_every_month`, and it computes the misinformation counts that were generated for every month available in the dataset. It does this by going through every month and year combination, and then running another helper function, then appending the returned value to a list. This helper function is titled `compute_misinformation_month`, and it returns the number of misinformation detected for the inputted month as well as the rate of misinformation. Essentially, using a list of myths, this function checks every line of the dataset for the month it was inputted through the misinformation detection algorithm, and then writes it to a file. To determine if a string is misinformation, it runs the `is_misinformation` function from the `similarity` module which we wrote. To determine if a string is misinformation, the function first runs `check_similarity_score`. This function uses FuzzyWuzzy's `token_sort_ratio()` function, which returns a score from 0–100 on how similar two strings are. By setting a threshold at 70%, it is able to identify when a string mostly contains a myth, although some are missed. The second part of the `is_misinformation` function is a second function called `is_false_positive`, which uses a list of negation words to determine if a sentence is actually NEGATING a myth, thus would not be misinformation. This function simply checks if there is a negation word, and returns True if there is. Putting both of these together, `is_misinformation` returns True if a function both has a similarity score above the threshold, and is not a false positive. Otherwise, the function returns False.

Data Visualization

In order to visualize this data, we settled on using the Pygame library. Specifically, in `visualize_data.py`, the `visualize()` function does most of the work. The function draws 3 rows of buttons, with 7 buttons per row, accounting for every month of the dataset. The function also has an empty gray bar which can be likened to a “progress bar”. By checking the mouse position on mouse clicks, we can turn the buttons functional. Simply, when a specific part of the screen is pressed, which is tied to a certain month/year combination, the progress bar will smoothly fill to a certain percentage for that month. This was chosen to be the way to visualize the data, because it allows for all of the data to be seen easily, and it is interactive. It gives the user more control over which data they want to see, as they are the ones who select which month/year combination to open.

Results

On our computations on the data, we found the following results:

Month/Year	% Tweets w/ Misinformation	Hard Values Tweets w/ Misinformation
Jan 2020	4.8%	20
Feb 2020	19.5%	143
Mar 2020	28.5%	154
Apr 2020	24.9%	173
May 2020	10.2%	37
Jun 2020	18.9%	88
Jul 2020	26.2%	95
Aug 2020	5.7%	22
Sep 2020	10.8%	27
Oct 2020	10.9%	56
Nov 2020	36.6%	134
Dec 2020	8.8%	34
Jan 2021	27.0%	97
Feb 2021	6.5%	18
Mar 2021	34.6%	79
Apr 2021	13.9%	50
May 2021	8.6%	23
Jun 2021	6.0%	12
Jul 2021	5.8%	16
Aug 2021	23.8%	70
Sep 2021	0.0%	0

Instructions to Reproduce

Obtaining Datasets:

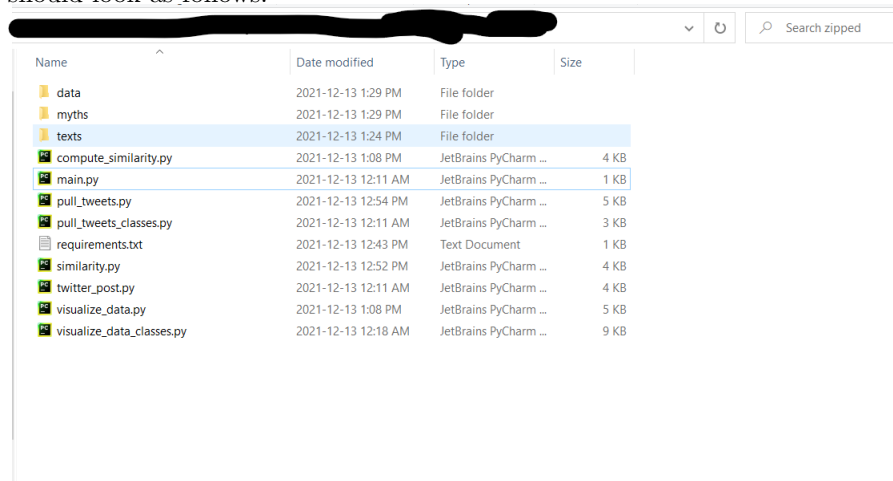
Step 1: Download the necessary files from **UTSend**:

Claim ID: [Redacted]

Claim Passcode: [Redacted]

Note, UTSend will download the pre-processed version of this tweets dataset. Unfortunately, due to the Twitter API's rate limiting, it is difficult to reproduce gathering tweet texts from their IDs.

Once you have downloaded the necessary data, as well as downloaded the Python files from MarkUs, your directory should look as follows:



Please note, when you unzip the file downloaded from UTSend, it will extract into a new folder. We do not want this. Instead, cut all the folders inside of the extracted folder and move them into the same folder as all of the Python files.

In the main directory, there are `filtered_twitter_ids_compiled` is the main dataset of filtered Twitter IDs, generated using `twitter_post.py`, `pull_tweets_classes.py`, and `pull_tweets.py`.
`twitter_post.py` requests the server and gets tweet IDs to search on the API.
`pull_tweets_classes.py` generates a list of needed month-year combinations.
`pull_tweets.py` randomly pulls valid tweets using their IDs from the Twitter dataset and writes them into `filtered_twitter_ids_compiled`.

`requirements.txt` is the required file that lists all the external libraries that we used.

`extracted_myths.txt` is the file that contains all of the myths that we extracted from the NewsGuard website using our NewsGuard Spider that will be used to compute misinformation.

`extra_added_myths.txt` is the file that contains our added myths that were not listed on the website that will be used to compute misinformation.

`negation_words.txt` is the file that contains negated words that will be used to compute similarity.

`compute_similarity.py` computes the amount of misinformation is found in tweets from each month and outputs it into a file.

`similarity.py` is a module that checks if two strings are similar - this is used by `compute_similarity.py`.

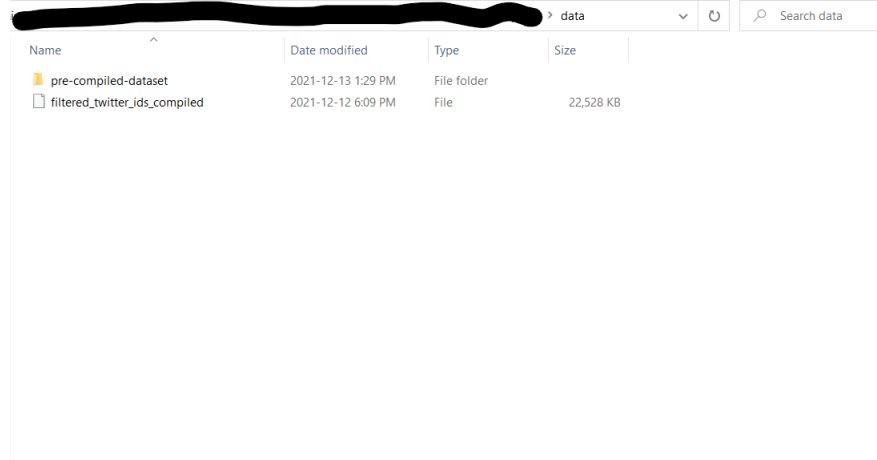
`visualize_data_classes.py` provides the necessary information to `visualize_data.py` to create a row of buttons in Pygame.

`visualize_data.py` visualizes all of our data.

`main.py` is the required main program file.

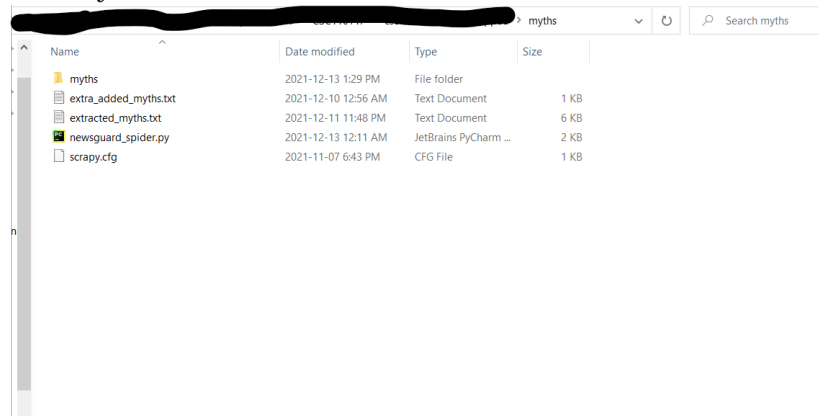
Inside of each folder, there should be data, and inside of `myths`, there should also be a web-scraper named `newsguard_spider.py`. Each folder should look as follows:

The data folder:



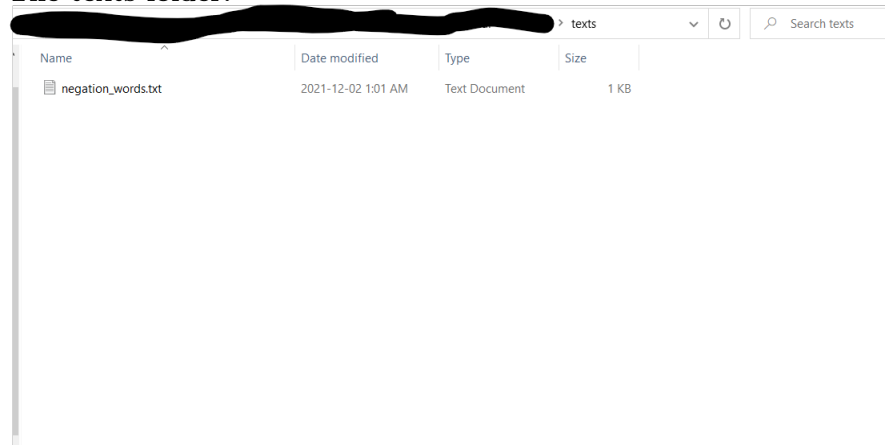
Name	Date modified	Type	Size
pre-compiled-dataset	2021-12-13 1:29 PM	File folder	
filtered_twitter_ids_compiled	2021-12-12 6:09 PM	File	22,528 KB

The myths folder:



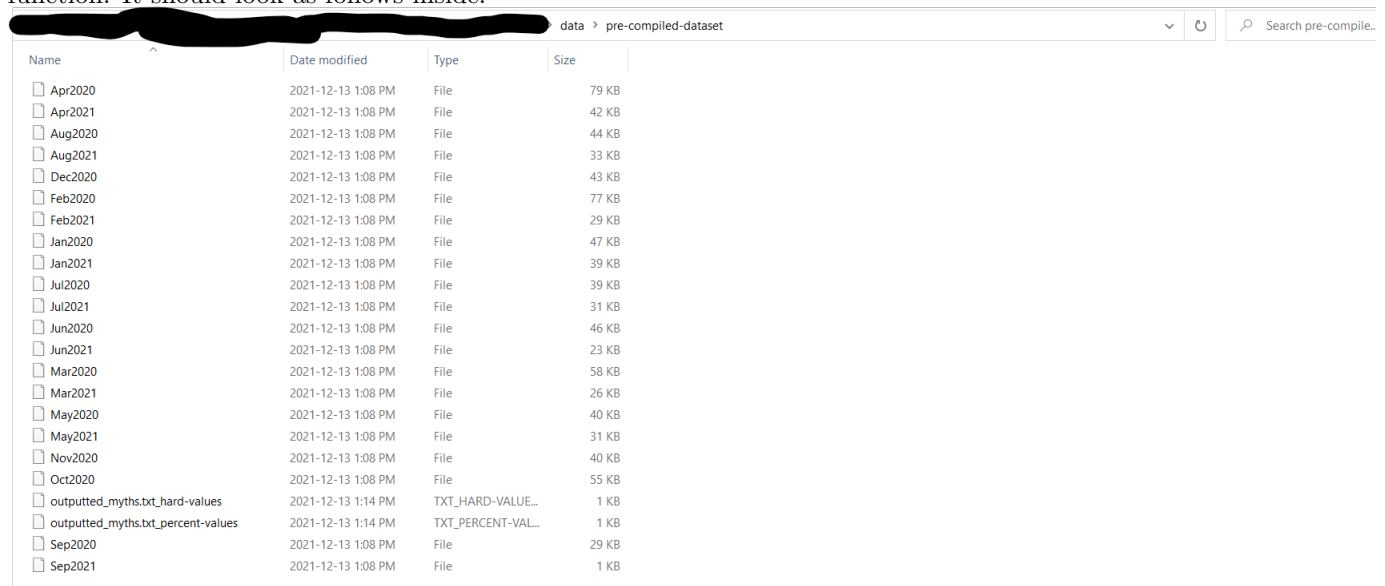
Name	Date modified	Type	Size
myths	2021-12-13 1:29 PM	File folder	
extra_added_myths.txt	2021-12-10 12:56 AM	Text Document	1 KB
extracted_myths.txt	2021-12-11 11:48 PM	Text Document	6 KB
newsguard_spider.py	2021-12-13 12:11 AM	JetBrains PyCharm ...	2 KB
scrapy.cfg	2021-11-07 6:43 PM	CFG File	1 KB

The texts folder:



Name	Date modified	Type	Size
negation_words.txt	2021-12-02 1:01 AM	Text Document	1 KB

Notice that there is also a subfolder inside of **data** named **pre-compiled-dataset**. This folder is only necessary if the computations on the data are too time consuming to reproduce, and the grader wishes to test the visualization function. It should look as follows inside:



Name	Date modified	Type	Size
Apr2020	2021-12-13 1:08 PM	File	79 KB
Apr2021	2021-12-13 1:08 PM	File	42 KB
Aug2020	2021-12-13 1:08 PM	File	44 KB
Aug2021	2021-12-13 1:08 PM	File	33 KB
Dec2020	2021-12-13 1:08 PM	File	43 KB
Feb2020	2021-12-13 1:08 PM	File	77 KB
Feb2021	2021-12-13 1:08 PM	File	29 KB
Jan2020	2021-12-13 1:08 PM	File	47 KB
Jan2021	2021-12-13 1:08 PM	File	39 KB
Jul2020	2021-12-13 1:08 PM	File	39 KB
Jul2021	2021-12-13 1:08 PM	File	31 KB
Jun2020	2021-12-13 1:08 PM	File	46 KB
Jun2021	2021-12-13 1:08 PM	File	23 KB
Mar2020	2021-12-13 1:08 PM	File	58 KB
Mar2021	2021-12-13 1:08 PM	File	26 KB
May2020	2021-12-13 1:08 PM	File	40 KB
May2021	2021-12-13 1:08 PM	File	31 KB
Nov2020	2021-12-13 1:08 PM	File	40 KB
Oct2020	2021-12-13 1:08 PM	File	55 KB
outputted_myths.txt_hard-values	2021-12-13 1:14 PM	TXT_HARD-VALUE...	1 KB
outputted_myths.txt_percent-values	2021-12-13 1:14 PM	TXT_PERCENT-VAL...	1 KB
Sep2020	2021-12-13 1:08 PM	File	29 KB
Sep2021	2021-12-13 1:08 PM	File	1 KB

!!!OPTIONAL!!! Refreshing the list of myths scraped from NewsGuard:

If you would like to refresh the myths scraped from the NewsGuard website, run `newsguard_spider.py` and then go into the Command Line and change the directory to the project folder, in the `myths` subfolder. Relative to the main project folder, it should be in `./myths/`. Then, run the command `scrapy runspider newsguard_spider.py` in the command line. This should update the `extracted_myths.txt` file with an updated set of scraped myths from the NewsGuard website.

Running The Program:

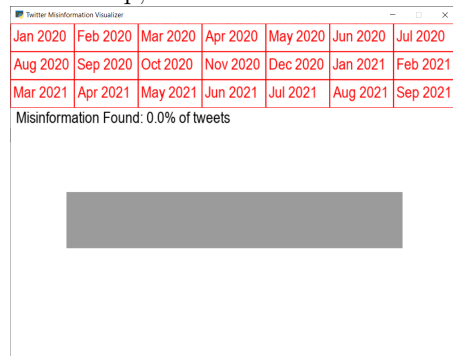
Step 1: Open `main.py` and run it.

Step 2: Wait for the hard values and percent values for misinformation in each month in our analysis to generate. This may take a long time depending on your computer's processing speed.

NOTE: If Step 2 takes too long, we have provided a sample of data that we have already generated. You can use this instead of waiting for the program to generate a new set of values. To use this, please comment out line 20 in `main.py`, move every file in `./data/pre-compiled-dataset` up a folder into `./data`. Then, re-run `main.py`.

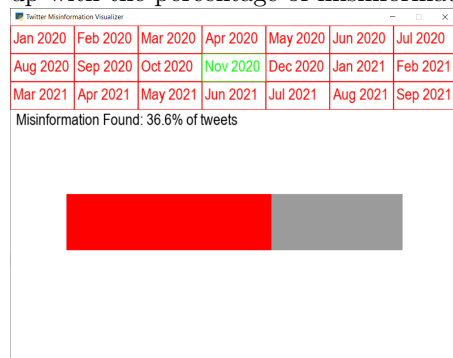
Step 3: When the Pygame window opens up, you will see a few rows of buttons at the top of the page, labelled with the month and year; the percent value of the misinformation found for that month (this will be empty at first); and the bar representing this percentage (this will be empty until you select a month). Simply click on the desired month to see the rate of misinformation found in that month in percentage value, and the bar that visually represents this percentage value.

On startup, the visualizer should look as follows:



If it does not look like the screenshot, there is a **glitch with *PYGAME***, and to fix it, you should **minimize, not close** the window and then reopen the window, and the glitch should be fixed.

When you click on a month/year combination, said button should turn green, and the progress bar will slowly fill up with the percentage of misinformation it detected. It will look as follows:



Changes Between Proposal and Final Submission

Between the proposal and the final submission, we made a few changes. First, we did not include the original dataset of myths we outlined in the proposal. This was because the dataset only contained myths for about three months worth of the pandemic, which made it largely irrelevant. Due to the immense number of computations performed by our algorithm to check for misinformation, we concluded that the few extra myths would not outweigh the extra time it would take to compute on the tweet data.

Another change we made was switching from searching for misinformation within news articles, to searching for misinformation within tweets on Twitter. We settled on this change because most news agencies perform intense fact checking for their articles, and if they do accidentally release misinformation, many will retract said articles making the entire project idea rather cumbersome, as there would be little misinformation found. On the other hand, we found that on Twitter, many people would simply post without fact checking, which allowed for widespread misinformation. Although Twitter does its due diligence to try to flag, and in some cases remove misinformation, they are bound to miss tweets, which was quite noticeable based on the results of our computations. We also came to the conclusion that discussions on Twitter have an immense impact on real world news, alongside the media. In fact, according to Pew Research, about 50% of adults in the United States say they either sometimes, or often get their news from social media. As such, misinformation within social media plays a very big role in the world, and we decided it would be beneficial to research this.

Discussion

The results from our computations show a very interesting trend: real world news impacted misinformation. The months that yielded the highest rates of misinformation among tweets on Twitter were April 2020, November 2020, January 2021, March 2021, April 2021, May 2021, and August 2021. These highlighted months coincide with the dates where worldwide COVID-19 cases spiked or the month following those months, where many areas of the world went into lockdown as an attempt to lessen the spread of the virus. Similarly, on the months where COVID-19 vaccines first became widely available and encouraged by multiple countries' governments, such as January 2021, the rate of misinformation in tweets also spiked. Both lockdowns and vaccines pertaining to COVID-19 are considered 'controversial' - they give rise to public disagreement and they tend to trigger strong opinions or emotions in many individuals. This may explain the spikes in rates of misinformation in Twitter. Moreover, the months where the lowest rates of misinformation were found on Twitter were August 2020, February 2021, June 2021, and July 2021. These months coincide with the points during the COVID-19 pandemic where worldwide cases were the lowest and when many regions of the world, as a result of these low number of cases, lifted or began to lift their lockdown restrictions. This data definitely helps answer the research question. Our data explicitly shows how misinformation within tweets on Twitter changed throughout the pandemic, and during which months spikes in misinformation rates occurred.

There were a number of limitations encountered throughout the development of this project. Due to Twitter terms of service, the dataset we used for the tweets did not contain any text from the tweets, instead storing the ID linking to the tweet. This caused a limitation with how accurate our data can be, as some of the tweets we attempted to analyze were deleted. Another limitation is with the FuzzyWuzzy library. Unfortunately, calculating Levenshtein distance is rather time consuming with this library, so once all the Twitter data is generated and filtered, computing on said data to find misinformation can take upwards of 15 minutes for all 21 months. Finally, a limitation with the algorithm is the way misinformation itself was detected. Using fuzzy string matching unfortunately does not account for words that are in different tenses, or different variations of the same word. For instance, 'consult' versus 'consultation' or 'consulting'. With our implementation of fuzzy string matching, only one of these three would be picked up by the algorithm. Another limitation involves synonyms. To illustrate, with words such as 'sickness' and 'illness', the algorithm would struggle detecting misinformation where the sample text does not match the inputted string.

To further explore the topic of the Change in Misinformation in Tweets on Twitter, we can improve the misinformation algorithm so that it is faster or more accurate. At this moment, the misinformation algorithm that calculates the amount and rate of misinformation in tweets is relatively slow - fuzzy string matching is calculated several times. To improve this project, we can incorporate machine learning instead of using fuzzy string matching, which would improve the accuracy of the misinformation algorithm by matching words by their root meaning, instead of their exact composition. For example, the algorithm would have the ability to identify two words with the same root as having the same relative meaning, such as "combine" and "combination." The algorithm could also be improved so that it could recognize two words that are synonyms as having the same meaning, such as 'sick' and 'ill.' Additionally, to further explore this topic, we can analyze the main topics that contribute the most to each month's rate of misinformation - for example, we can calculate the rate of each trigger word for misinformation: whether that be "lockdown", "vaccine", etc.

References

- Bowden, J. (2020, May 12). Accidental poisonings from bleach and other disinfectants spiked amid coronavirus. TheHill. Retrieved November 4, 2021, from <https://thehill.com/policy/healthcare/497312-accidental-poisonings-from-bleach-and-othr-disinfectants-spiked-amid>.
- Bryner, J. (2020, April 24). Disinfectant injections are a really bad idea. LiveScience. Retrieved November 4, 2021, from <https://www.livescience.com/disinfectant-injections-coronavirus-dangerous.html>.
- Edward, A. (2021, June 13). An Extensive Guide to collecting tweets from Twitter API v2 for academic research using Python 3. Retrieved December 12, 2021, from <https://towardsdatascience.com/an-extensive-guide-to-collecting-tweets-from-twitter-api-v2-for-academic-research-using-python-3-518fcb71df2a>
- Gupta, Raj, Vishwanath, Ajay, and Yang, Yiping. (2021). COVID-19 Twitter Dataset with Latent Topics, Sentiments and Emotions Attributes [Data set]. <https://www.openicpsr.org/openicpsr/project/120321/version/V11/view;jsessionid=EB090760F70409E47B82FC0137D7B1A1>
- NewsGuard. (2021, November 1). Coronavirus Misinformation Tracking Center. NewsGuard. Retrieved November 4, 2021, from <https://www.newsguardtech.com/special-reports/coronavirus-misinformation-tracking-center/>.
- Office of the Commissioner. (2021, September 7). Why you should not use ivermectin to treat or prevent COVID-19. U.S. Food and Drug Administration. Retrieved November 4, 2021, from <https://www.fda.gov/consumers/consumer-updates/why-you-should-not-use-ivermectin-treat-or-prevent-covid-19>.
- van der Linden, S., Roozenbeek, J., & Compton, J. (2020). Inoculating against fake news about COVID-19. *Frontiers in Psychology*, 11. <https://doi.org/10.3389/fpsyg.2020.566790>
- World Health Organization: WHO. (2021, April 21). Fighting misinformation in the time of covid-19, one click at a time. World Health Organization. Retrieved November 4, 2021, from <https://www.who.int/news-room/feature-stories/detail/fighting-misinformation-in-the-time-of-covid-19-one-click-at-a-time>.