CSC110 Project: COVID-19 Discussion Trend Analysis

Azalea Gui & Peter Lin

December 5, 2021

1 Problem Description and Research Question

We have observed that there have been increasingly more voices talking about COVID-19 since the start of the pandemic. However, different groups of people might view the importance of discussing the pandemic differently. For example, we don't know whether the most popular people on Twitter will be more or less inclined to post COVID-related content than the average Twitter user. Also, while some audience finds these content interesting, others quickly scroll through them. **So, we aim to compare people's interests in posting coronavirus content and the audience's interests in viewing them between different groups.** Also, with recent developments and policy changes toward COVID-19, it is unclear how people's discussions would react. Some people might believe that the pandemic is starting to end so that discussing it would seem increasingly like an unnecessary effort, while others might find these policy changes controversial and want to voice their opinions even more. Also, even though COVID-related topics are almost always on the news, some news outlets might intentionally cover them more frequently than others. For the people watching the news, some people might find these news reports interesting, while others can't help but switch channels. So, how people's interest in listening or discussing COVID-related topics changes over time is not very clear. **Our second goal is to analyze how people's interest in COVID-related topics changes and how frequently people have discussed COVID-related issues in the two years since the pandemic started.**

2 Dataset Used

1. A wide range of Twitter users: We used twitter's get friends list API (13) and the follows-chaining technique to obtain a wide range of twitter users. This technique is explained in the Computational Overview section. Due to rate limiting, we ran the program for one day and obtained 224,619 users (852.3 MB decompressed). However, only the username, popularity, post count, and language data are used, and the processed (filtered) user dataset `data/twitter/user/processed/users.json` is only 7.9 MB in total.
2. All tweets from sampled users: We selected two samples of 500 users each (the sampling method is explained in the Computational Overview section), and we used the user-timeline API (14) to obtain all of their tweets. Due to rate limiting, the program took around 16 hours to finish, and we obtained 6.07 GB of raw data (uncompressed). During processing, we reduced the data for each tweet to only its date, popularity (likes + retweets), whether it is a retweet, and whether it is COVID-related. The text of the tweets are not retained, and the processed data directory `data/twitter/user-tweets/processed` is only 107.9 MB in total.
3. Top 100 news twitter accounts by Bremmen (3)
4. COVID-19 daily new cases data by New York Times (1).

3 Computational Overview

Data Gathering & Processing

This section explains the data gathering and processing done in `collect_twitter.py`, `collect_others.py`, and `processing.py`. In this section, raw data will be collected and processed into the `processed_data.7z` that we provided.

To create our samples, we collected a wide range of Twitter users using Twitter's get friends list API endpoint through **tweepy**, using the follows-chaining technique. We specified one single user as the starting point (in this case, we picked `voxdotcom`). The program then obtains the user's friends list, picks 3 random users and 3 most followed users from the friend list, adds them to the queue, and starts the downloading process again from each of the six friends. Because of Twitter's rate limiting on the get friends list endpoint, we can only obtain a maximum of 200 users per minute, with many of them being duplicates. We ran the program continuously for one day and obtained 224,619 users (852.3 MB decompressed). However, only the username, popularity, post count, and language data are kept after processing (filtering). The processed user dataset `data/twitter/user/processed/users.json` is 7.9 MB in total. We selected our samples by filtering the results first based on language, selected the top 500 most followed users as 500-pop, filtered the list again based on post count (¿1000) and followers (¿150), then selected a random sample of 500 users as 500-rand.

We also downloaded all tweets from our sampled users through the user-timeline API also with **tweepy**. Due to rate limiting, the program took around 16 hours to finish, and we obtained 7.7 GB of raw data (uncompressed). During processing, for each tweet, we extracted only its date, popularity (likes + retweets), whether it is a retweet, and whether it is related to the COVID-19 pandemic. The text of the tweets are not retained, and the processed data directory `data/twitter/user-tweets/processed/` is 141.6 MB in total.

To determine whether a post is COVID-related we used keyword matching with three lists of COVID-related keywords for English, Chinese, and Japanese. Tweets with content containing these keywords are marked as COVID-related.

We also used the COVID-19 daily cases data published by New York Times to compare with peaks and through in our frequency over date graph, and the program gathered this data by sending an HTTP requst to New York Times' public github repository using **requests** and then parsing the CSV.

For submission, we packed the processed data into a 7zip archive using **py7zr**. This is necessary because our processed data are placed very close to the raw data in the folder structure, and creating the archive manually requires separating the processed data from the raw data into two separate folders first. We also used py7zr to pack our HTML resources.

We also used **json5** to store the configuration of this program, which contains Twitter API keys.

Statistical Visualization Generation

This section explains the statistical report generation done in `visualization.py`. In this section, specific "elements" used in our report are generated. For example, an element might be an image of the user frequency graph in one of our samples, and another element might be a markdown table showing the amount of users who posted less than 1% or didn't post in our samples. Each element is stored in a separate file, which will be included in the visualized report explained in the next section.

Since the statistical computations of report generation is explained in the interactive report, this section will only focus on the technical aspect of which libraries we used to complete these computations and generate the statistical elements of the report.

We used **matplotlib** to generate images that will be displayed in our report, including histograms and line graphs. We used **scipy** for signal filtering and smoothening the curves so that they are readable (specifically, `scipy.signal.lfilter`). We used **numpy** in our statistical calculations to calculate percentile points and remove outliers. We then used **tabulate** to generate Markdown format tables for report elements.

Interactive Report Generation

This section explains the interactive report creation in `report.py`.

We wrote our report in Markdown format, located in `resources/report_document.md`. However, the default Markdown format doesn't support including the contents of other markdown files generated in the previous step, so we extended the markdown format by adding `@include`, `@include-lines`, and `@include-cut` functionality.

Then, to display the markdown in a webpage, we created a template HTML (`resources/report_page.html`) and used python to inject the markdown content into the HTML template. Then, we used **Marked** (a JS library) to render the Markdown to the webpage. We did not use the python Markdown library because it did not support the Github Markdown table format generated with **tabulate** in the previous step. Then, we used the **Flask** framework to serve the webpage along with the referenced assets like images, js, and fonts on an HTTP server.

On the webpage, we used **jQuery** (a JS library) to make the images enlargeable. We also imported **MathJax** (a JS library) to automatically render LaTeX on the webpage (no code needed to reference this library).

Even though the handout required the project to be purely written in Python, instructors in Piazza allowed us to use web languages as long as all data gathering, processing, computation, visualization, and image rendering are done in pure Python (@1704).

4 Running Instructions

1. Download the submitted files into a new folder called `src`.
2. Extract the archive `src/resources.7z` into the folder `src/resources`.
3. Install `src/requirements.txt`, either with PyCharm or with `pip install -r src/requirements.txt`.
4. If you would like to test out our data collection code or collect data manually, do the following: (We do not recommend collecting all data manually because it took the program two days to gather our data due to rate limiting)
   a. Register for Twitter API keys on their website. For more information, look at the following link: (Getting access to the Twitter API).
   b. Copy the Twitter API keys into the corresponding fields in `src/config.json5`
   c. In `src/main.py`, uncomment all the lines of code for data collection and processing: that is, the steps C1.0-C1.2, P1-P2, C2.1-C2.3, P3.
5. If you would like to use our processed data, you can download the archive from `https://send.utoronto.ca` with the following code, which will expire on December 27.
   Claim ID: 6PPMsHQNTV7TJRmu
   Claim Passcode: 9VPba4YiYx2cetbU
   Alternatively, you can download it from a permanent link: `https://csc110.hydev.org/processed-data.7z`
   Extract the archive into a directory called `data` at the same level as `src`, that is, `data` and `src` should be in the same folder.
   The file `src/constants.py` contains a more detailed directory tree.
6. Run `src/main.py`, either in PyCharm or with `python3 main.py`. Note that the execution directory should be in `src` and not the root directory. If you use PyCharm, you should open `src` in PyCharm instead of the root directory. This file structure is intentionally designed to prevent PyCharm from indexing `data`, which takes an extremely long time.

5 Changes to Proposal

First, we originally planned to include news reports from separate journal websites in our analysis as well. However, when we gather the data, we found that there is no way to identify the popularity of a news report published on a journal website. So, we decided to gather the tweets of news accounts on Twitter instead, which will also have the benefit of having the same data gathering and analysis process for each news channel.

Second, we originally planned to compare people's interests in posting COVID-related topics between different platforms because we thought Chinese people don't rely on Twitter as much since Twitter is blocked in China. However, there isn't any publically available WeChat API that we can use for analysis, and WeChat is also more private, with access to someone's postings limited to only their friends. (It is as if everyone on Twitter has a locked account). Therefore, it is impractical to gather data from WeChat. And, for Telegram channels, the postings does not have a like feature, and might not have commenting feature unless the channel host specifically set up for it using a third-party bot. So there isn't a reliable way to obtain popularity data on Telegram as well. So, instead of comparing between platforms, we compared different groups of people on the Twitter platform.

6 Discussion

To try to answer our research questions, we made 3 user samples: `500-pop`, composed of the 500 most popular users on Twitter, `500-rand`, 500 random users on Twitter, and `eng-news`, the top 100 English news channels on Twitter (3).

Our first research question asks: **how frequently do people post about COVID-related issues, and how interested are people to see COVID-related posts?** After making graphs showing the frequency of COVID-related posts, the histograms showing frequency all tend to be skewed right, or decreasing. In all 3 samples of `500-pop`, `500-rand`, and `eng-news`, for every user, the majority of their posts were not COVID-related. However, news channels tended to post more about COVID than `500-pop`, who tended to post more about COVID than `500-rand`. This suggests that the `500-pop` users tend to post more than the average person about COVID, possibly because they have a large influence and want to share their opinion, but they post less about COVID than news channels, who have to report the latest news about the pandemic, regulations, and slowing the spread. The histograms showing the popularity ratios are also skewed right, except the news, which is almost centered. This shows that for COVID-posts outside of news channels, they usually receive poor engagement. However, for news channels, they receive almost as much engagement as their other posts, suggesting that users are indifferent to COVID-news as regular news.

Our second research question tackled the same thing: but instead, how it changed over time. From the line graphs showing COVID-posting frequency over time, it tends to decrease for the samples of `500-rand` and `eng-news`, although `500-pop` and `eng-news` shared a peak around December 2020 and trough around June 2021, which seems to be related to the rise and fall of COVID cases in the US. A bit surprisingly, the news has stopped posting as many COVID-related posts, maybe because they are starting to incorporate other keywords such as ”Delta variant”. In addition, there is a large spike in May 2021 for the sample `500-pop`, which is also around when the Delta variant started appearing. This could mean that the news started talking about that more, while the popular twitter users talked about its consequences. However, the `500-rand` sample has seen a decrease in frequency, suggesting that they post less about it, possibly because their COVID posts get less engagement, or just a general decrease in the interest in COVID as a topic. We speculate it can possibly be people getting used to seeing COVID-related news all the time, and just like background noise, are starting to ignore it (a process called habituation). The line graphs showing popularity ratio over time, however, were a lot messier. Despite attempts to filter out the noise, there were constants spikes in the graph throughout the entire graph, making it difficult to draw any conclusions. This could possibly be because the sample size is too small, or the method of determining popularity isn't sophisticated enough and prone to random noise. Unfortunately, this means that further research is required to answer our question.

References

[1] Almukhtar, Sarah et al. *Nytimes/COVID-19-data: An ongoing repository of data on coronavirus cases and deaths in the U.S.* 2021. URL: `https://github.com/nytimes/covid-19-data`.

[2] Astanin, Sergey et al. *Tabulate.* 0AD. URL: `https://pypi.org/project/tabulate/`.

[3] Bremmen, Nur. *100 most Influential News Media Twitter Account Archives.* Sept. 2010. URL: `https://memeburn.com/motorburn/tag/100-most-influential-news-media-twitter-account/`.

[4] Community, The SciPy. *SciPy documentation.* 0AD. URL: `https://scipy.github.io/devdocs/index.html`.

[5] Hunter, John et al. *Overview.* Aug. 2021. URL: `https://matplotlib.org/stable/index.html`.

[6] Miura, Hiroshi. *Py7zr.* 0AD. URL: `https://pypi.org/project/py7zr/`.

[7] Numpy. *NumPy v1.21 manual.* 2021. URL: `https://numpy.org/doc/stable/`.

[8] Pranke, Dirke. *JSON5.* 2021. URL: `https://pypi.org/project/json5/`.

[9] Reitz, Kenneth. *HTTP for Humans™.* 0AD. URL: `https://docs.python-requests.org/en/master/index.html`.

[10] Richardson, Leonard. *Beautiful Soup documentation.* 0AD. URL: `https://beautiful-soup-4.readthedocs.io/en/latest/`.

[11] Roesslein, Joshua. *Tweepy documentation.* 2021. URL: `https://docs.tweepy.org/en/stable/`.

[12] Ronacher, Armin. *Flask.* 0AD. URL: `https://pypi.org/project/Flask/`.

[13] Twitter. *Get friends/list — docs — twitter developer platform.* 0AD. URL: `https://developer.twitter.com/en/docs/twitter-api/v1/accounts-and-users/follow-search-get-users/api-reference/get-friends-list`.

[14] Twitter. *Get statuses/user_timeline — docs — twitter developer platform.* 0AD. URL: `https://developer.twitter.com/en/docs/twitter-api/v1/tweets/timelines/api-reference/get-statuses-user_timeline`.