COVID-19 Case Data Collection for Coronavirus TwitterMap

Tiancheng (Joseph) Zheng

ICS Honors Thesis

June 2020
Abstract

The goal of this project is to collect COVID-19 case numbers in the United States from January 1st, 2020 at both the state and county levels to display on the count map of the Coronavirus TwitterMap. Two different technical approaches are used during the development of the Coronavirus TwitterMap website. The initial approach was to use web crawlers to crawl data from the website containing the COVID-19 case data, and the current approach is to use the provided API to fetch the COVID-19 case data from the data source.
# Table of Contents

1. **Introduction**.................................................................................................................. 5

2. **Determining the Data Source**.......................................................................................... 7

3. **Web Crawler**.................................................................................................................. 9
   3.1. Scrapy ................................................................................................................................. 9
   3.2. Selenium WebDriver ....................................................................................................... 11
   3.3. Politeness ......................................................................................................................... 12
   3.4. Maintenance ..................................................................................................................... 13
   3.5. Deployment and Automation ......................................................................................... 13
   3.6. Application Logging ....................................................................................................... 13

4. **1Point3Acres.com API**.................................................................................................... 14

5. **Conclusion and Future Plans** ........................................................................................ 16

References .................................................................................................................................. 17
Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TwitterMap Interface</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>Coronavirus TwitterMap Interface</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>1Point3Acres.com COVID-19 Website</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>State Data (Collapsed List)</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>County Data (Expanded List)</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>Code for Crawling County Data of a State using Selenium WebDriver</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>Code for Converting the API Data into Required Data Format</td>
<td>15</td>
</tr>
</tbody>
</table>
1. Introduction

Cloudberry\(^1\) is developed by a research team led by Professor Chen Li from the Information Systems Group (ISG) at the University of California, Irvine. It is a general-purpose middleware system to support visualization on large amounts of data. As a middleware, Cloudberry “communicates with backend database management systems via adapters,” and it also “supports various frontend interfaces by providing a RESTful interface” \([1]\).

An example of such a frontend interface is TwitterMap\(^2\), a research prototype powered by Cloudberry and Apache AsterixDB that “supports interactive analytics and visualization on more than 1.9 billion tweets with new data continuously being ingested” \([2]\). Users can type in any keyword and TwitterMap will visualize the distribution of tweets across the United States since November 23\(^{rd}\), 2015 on three different kinds of maps: count map, heat map, and pin map. Users can also adjust the time bar at the bottom of the interface to display the distribution of tweets in a customized time range. Using this interface, users can analyze the spatial, temporal, and textual aspects of different keywords on Twitter and observe the popularity of different topics on social media across the United States.

In the past few months, the hottest topic around the world is COVID-19, the novel coronavirus that has been spreading throughout the entire planet since January 2020. To give the public a general sense of the real-time social media conversations about COVID-19, our team developed and launched a new website called Coronavirus TwitterMap\(^3\). It is a COVID-19 oriented version of TwitterMap which supports “interactive analytics and visualization of large amounts of COVID-19 related Twitter data and case numbers” [3], including the confirmed cases, death count and recovered count of COVID-19.

\(^3\) Coronavirus TwitterMap, [https://coronavirustwittermap.ics.uci.edu](https://coronavirustwittermap.ics.uci.edu).
The most notable difference between these two interfaces is the addition of a trending chart with the aforementioned COVID-19 confirmed, death and recovered counts in the popup window of every state and county in the count map. The goal of my project is to collect the COVID-19 case data in the United States since January 1st, 2020 at both the state and county levels in order to draw these trending charts.

2. Determining the Data Source

The first step of this project is to determine an authoritative data source that contains COVID-19 case numbers for all the states and counties in the United States. Prior to my project, the data we used came from the John Hopkins University COVID-19 Map website. However, the county-level case numbers were outdated and incorrect, so it was urgent that we find a new data source that contained up-to-date county data.
My initial idea was to collect the county data from every state separately via different state healthcare agencies and local news. For example, an article by the *Los Angeles Times*, “Tracking coronavirus in California,” tracks COVID-19 cases in every county and every city in California and is updated very frequently\(^4\). However, not every state has a news outlet that reports this kind of detailed data. In addition, even if there were this kind of data for all the states, it would still be too onerous of a task to track the COVID-19 case numbers site by site on a daily basis.

In the end, I came across this website maintained by 1Point3Acres.com\(^5\), and it contains all the data required by this project. According to the developers, this website “is the only platform providing real-time global case information of more than 4,124 sub-divisions from over 27 countries worldwide with multi-language supports” [4]. It has detailed information on confirmed cases, death count, fatality rate and recovered count for every state and every county in the United States. After discussing with Professor Li and the rest of the team, we were all satisfied with this website, so it became our new data source.

---


\(^5\) 1Point3Acres.com COVID-19 Website, [https://coronavirus.1point3acres.com](https://coronavirus.1point3acres.com).
3. Web Crawler

We sent a request to 1Point3Acres.com asking for access to the COVID-19 data through their API, but we did not receive an immediate response. Since our project was urgent, Professor Li suggested that I should implement a web crawler to get the data from the website.

3.1. Scrapy

This is my first time implementing a web crawler from scratch, and I decided to use Scrapy\(^6\), which is “an open-source and collaborative framework for extracting the data from websites” [5]. I set up my project according to the tutorial provided by Scrapy, and it was really simple. I was able to write a spider in Python and crawled all the state data from 1Point3Acres.com. However, when I tried to retrieve the data for the counties, I encountered a

---

serious issue that I could not expand the collapsible list using Scrapy commands because the feature to expand the list on this website was implemented using JavaScript.

Figure 4: State Data (Collapsed List)

Figure 5: County Data (Expanded List)
As seen in the above two screenshots, the only way to get the county data is to expand the info for each state by clicking on the state’s row in the table. Without the source code of the JavaScript, it is difficult to determine exactly how the action of expanding the list is done in the frontend, so I needed to write a script to simulate the click event in order to expand the list and get the COVID-19 case numbers for the counties.

### 3.2. **Selenium WebDriver**

I found several solutions to this problem online, and the one that stood out to me was the Selenium WebDriver\(^7\). It “drives a browser natively, as a real user would, either locally or on a remote machine using the Selenium server” [6], which is exactly what I intended to do. The other required dependencies to use the Selenium WebDriver include geckodriver v0.26.0 and the Firefox Browser.

The built-in Selenium WebDriver functions that I found particularly useful were `find_element_by_xpath()`, which locates an HTML element in a page by its XPath, `execute_script(script, *args)`, which synchronously executes JavaScript in the current window or frame, and `click()`, which clicks on the designated HTML element. These commands work together to accomplish the purpose of clicking on the row of a specific state and expanding the list to acquire the COVID-19 data for the counties of that state.

---

3.3. Politeness

In my initial design, the crawler code could retrieve COVID-19 data for every state and county from 1Point3Acres.com in less than 5 minutes by sending thousands of requests to the website in quick successions. This did not occur to me as a problem until I showed my design to the team and one of the Ph.D. students pointed out that this crawler was very impolite and could be easily detected and banned by the website. To fix this issue, I added a time.sleep() function after every request to introduce a 0.5-second delay to make the crawler more polite. This additional delay time causes the crawler to take almost 2 hours to finish the execution, but it is a worthy tradeoff as the crawler becomes more conformed to the conventions of web crawlers.
3.4. **Maintenance**

The most significant challenge of maintaining a web crawler is that the targeted website can change its HTML structure very frequently, so the hardcoded HTML element XPaths in the crawler code need to be updated accordingly as well. In order to detect the website changes, I need to constantly keep an eye on my crawler code to make sure that I can update the code immediately once an error occurs. In this way, we can have an up-to-date crawler code that constantly gets COVID-19 data from 1Point3Acres.com. The maintenance stage of the crawler code takes a lot more time than the development stage.

3.5. **Deployment and Automation**

I deployed the crawler code to an Amazon Web Services (AWS) machine and set up a cron job to run the code automatically every three hours. In this way, we can retrieve the newest data and keep the data on our Coronavirus TwitterMap website as updated as possible.

3.6. **Application Logging**

An important practice of maintaining a codebase is to enable logging to keep track of the execution status of the program every time it runs, especially when the program is executed automatically. Using these logs, the developers can discover the potential bugs in their programs. For the logs of my crawler, I suppressed many unnecessary build-in loggings from Scrapy and added additional logging information, including the crawled COVID-19 data and the possible exceptions that could occur during the execution of the crawler. I detect website changes and update my crawler code whenever the logs indicate a failed execution.
4. 1Point3Acres.com API

Two weeks after the initial development of the Coronavirus TwitterMap website, Professor Li received the API access key from 1Point3Acres.com. This API provides access to the United States’ COVID-19 cases in chronological order since January 1st, 2020, and the data is updated daily at 12 AM EST. I wrote a script to aggregate the cases by state and county to convert it into the desired data format used by our website.

During our team meeting, I suggested that since the API data would be updated on a daily basis, we could use its data as the official standard data at the end of each day and use the data obtained by the web crawler as the real-time data. Professor Li objected to this notion and said that the new way of using the API is the “right,” professional way. He added that it would be okay to have data a bit late, which addressed my main concern about the API data.

Apart from being the more professional way to obtain data, the API has three additional advantages over the web crawler. Firstly, compared to the 1Point3Acres.com COVID-19 website, the CSV file provided by their API is more static and less prone to format changes, making it easier to maintain the codebase. Secondly, when using the web crawler, we can only obtain data at the time of the crawling and cannot make changes to any data from the past. If 1Point3Acres.com corrects erroneous data from a previous date, we cannot detect the change and update our data, so we will display incorrect data on our website. With the new technique, since the data is aggregated daily using the updated API data, we will always have the newest data from 1Point3Acres.com for every date we display. Lastly, instead of running the web crawler script 8 times a day for a total of nearly 20 hours, we now only need to run the API script once a day at midnight for merely 12 seconds, so the new technique saves a lot of computing resources.
As a result, we are now using the COVID-19 case numbers obtained using the 1Point3Acres.com API as our backend data for Coronavirus TwitterMap. The web crawler code was deprecated as of April 17th, 2020.

```python
def read_csv_to_json():
    data = {}
    with open('data/state_county_info/state_abbr.txt', 'r') as f:
        state_abbr = json.load(f)
    with open('data/raw_data/raw_data.csv', 'r') as f:
        data_lines = f.readline()
        previous_date = None
        current_date = None
        for line in data_lines[1:]:
            case = line.strip().split(',')
            if len(case) < 6:
                continue
            confirmed_date = case[1]
            if case[2] in state_abbr.keys():
                state = state_abbr[case[2]]
            else:
                logging.warning(f'New State: {case[2]}')
                state = case[2]
                county = case[3]
                confirmed = int(case[4])
                death = int(case[5])
                recovered = int(case[6])
                this_date = datetime.datetime.strptime(confirmed_date, '%Y-%m-%d')
                if current_date is None:
                    current_date = this_date
                    previous_date = this_date
                current_date_string = previous_date.strftime('%Y-%m-%d %H:%M:%S')
                for date in date_range(previous_date, current_date):
                    date_string = date.strftime('%Y-%m-%d %H:%M:%S')
                    dt_string = this_date.strftime('%Y-%m-%d %H:%M:%S')
                    if dt_string not in data.keys():
                        data[dt_string] = {}
                    if state not in data[dt_string].keys():
                        data[dt_string][state] = {'confirmed': 0, 'deaths': 0, 'recovered': 0, 'county': {}}
                    if county not in data[dt_string][state]['county'].keys():
                        data[dt_string][state]['county'][county] = {'confirmed': 0, 'deaths': 0, 'recovered': 0}
                    data[dt_string][state]['confirmed'] += confirmed
                    data[dt_string][state]['deaths'] += death
                    data[dt_string][state]['recovered'] += recovered
                    data[dt_string][state]['county'][county]['confirmed'] += confirmed
                    data[dt_string][state]['county'][county]['deaths'] += death
                    data[dt_string][state]['county'][county]['recovered'] += recovered
    with open('data/raw_data/raw_data_json.txt', 'w') as f:
        json.dump(data, f)
```

*Figure 7: Code for Converting the API Data into Required Data Format*
5. Conclusion and Future Plans

The Coronavirus TwitterMap website is now open to the general public and has moved from the development stage into the maintenance stage. An article on our work has also been published on UCI News\(^8\).

Through this project, I have gained an in-depth understanding of web crawling and its conventions. I also learned that we need to be careful with web crawling and use crawlers to retrieve data only as a last resort.

As COVID-19 continues to rampage in the United States, our work will continue to be of great significance, and it is important for me to make sure that up-to-date case numbers are displayed on our website. Currently, the API data from 1Point3Acres.com contains confirmed cases, death count, and recovered count. If more data becomes available through the API, I will update my codebase accordingly and provide the newest data to the public through our Coronavirus TwitterMap website.

---

References


[4] Yang, Tong, Kai Shen, Sixuan He, Enyu Li, Peter Sun, Lin Zuo, Jiayue Hu et al.
