Kevin Esslinger

Professor Eduard Dragut

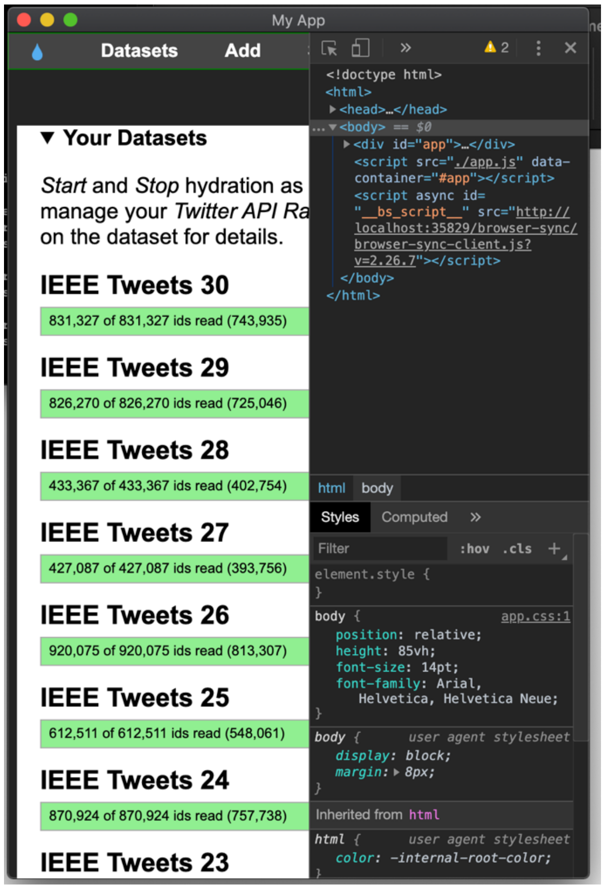
CIS 4517 Data-Intensive and Cloud Computing

27 April 2020

Cleaning a COVID-19 Tweet Dataset

For my project, I downloaded a dataset of tweets all relating to the COVID-19 pandemic. Because Twitter's API Terms of Service do not allow for the tweets to be downloaded directly, I downloaded a list of Tweet IDs corresponding to the exact tweets. After downloading them, I converted the IDs into the actual tweets in the form of JSON, converted those JSON files into CSVs, and cleaned the CSV files. In particular, I removed duplicate tweets found in the dataset files as well as removed retweets (defined as tweets beginning with "RT @"). I have stored the dataset at each step of the cleaning process, and the code to reproduce this work is available online (at https://github.com/kevslinger/CIS4517_Final_Project). This report will go into the details of how the dataset was cleaned.

As I mentioned, the Twitter API Terms of Service does not allow us to share actual tweet content; instead, we need to download the IDs of the tweets and get them from the API ourselves. I downloaded the dataset from https://ieee-dataport.org/open-access/corona-virus-covid-19-tweets-dataset, resulting in over 5 million tweets. After downloading the set of Tweet IDs, I installed and used the hydrator tool (https://github.com/DocNow/hydrator) to convert the Tweet IDs into full tweets in JSON files. A screenshot is attached showing the process of uploading the Tweet ID list to the hydrator tool where it downloads the tweets as a JSONL file (where each line of the file is its own JSON file). From there, I wrote a quick python function (see my json_to_csv.py file on github) to convert the JSONL files to CSV files to make them easier to work with. After that, I was able to start cleaning the tweets.



▼ Your Datasets

Start and *Stop* hydration as
manage your *Twitter API Ra*
on the dataset for details.

IEEE Tweets 30
831,327 of 831,327 ids read (743,935)

IEEE Tweets 29
826,270 of 826,270 ids read (725,046)

IEEE Tweets 28
433,367 of 433,367 ids read (402,754)

IEEE Tweets 27
427,087 of 427,087 ids read (393,756)

IEEE Tweets 26
920,075 of 920,075 ids read (813,307)

IEEE Tweets 25
612,511 of 612,511 ids read (548,061)

IEEE Tweets 24
870,924 of 870,924 ids read (757,738)

IEEE Tweets 23

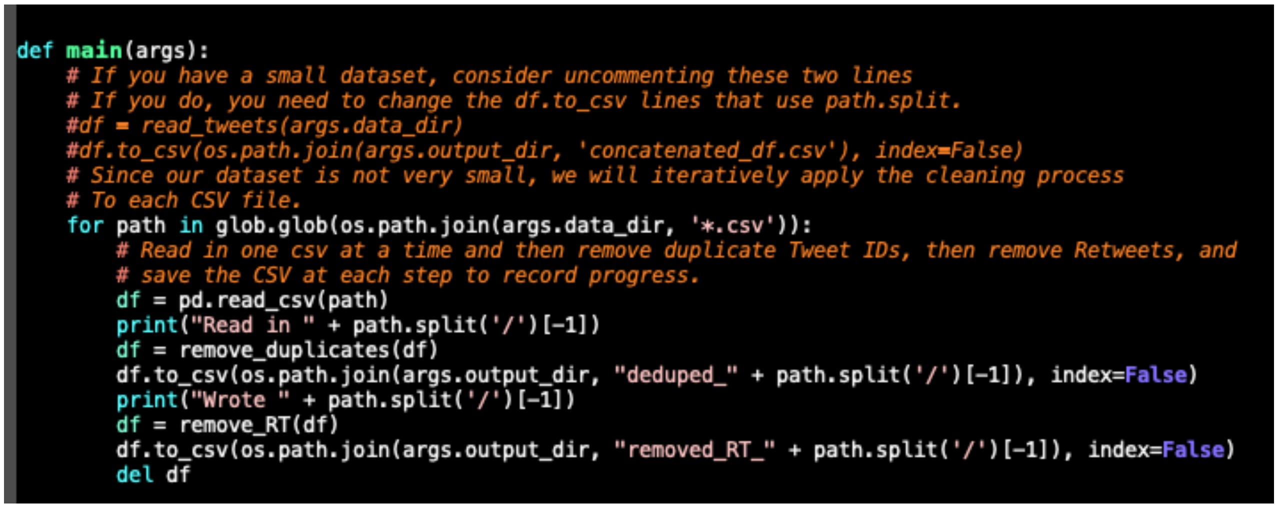
```
<!doctype html>
<html>
 ▸<head>…</head>
…▾<body> == $0
   ▸<div id="app">…</div>
    <script src="./app.js" data-
    container="#app"></script>
    <script async id=
    "__bs_script__" src="http://
    localhost:35829/browser-sync/
    browser-sync-client.js?
    v=2.26.7"></script>
  </body>
</html>
```

```
element.style {
}
body {                      app.css:1
   position: relative;
   height: 85vh;
   font-size: 14pt;
   font-family: Arial,
      Helvetica, Helvetica Neue;
}
body {        user agent stylesheet
   display: block;
   margin: ▸ 8px;
}
Inherited from html
html {        user agent stylesheet
   color: -internal-root-color;
}
```

Screenshot 1.) a view of the hydrator tool which I used to convert the Tweet IDs to full-text tweets in JSON

Because the full dataset of tweets total more than 30 GB of memory, I needed to use some additional computing power to clean the tweets. I started up an xlarge EC2 instance with ubuntu and 100GB of disk space, uploaded my dataset to it, and cloned my GitHub repo. The xlarge instance has 32 GB RAM, allowing me to work with the dataset broken up into smaller files (I used the files as they came from the dataset). I wrote my script in such a way to iteratively handle one file at a time instead of combining each file into one massive CSV file because of the size of my dataset. Now, I will discuss the code used to clean the tweets.

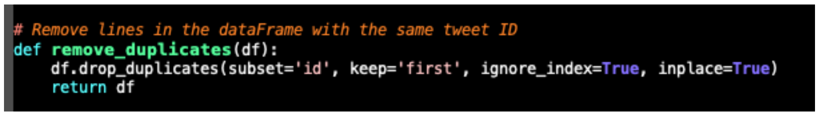
All the code used has been written in python files json_to_csv.py and main.py. I used json_to_csv to convert the JSONL files to CSV files, and used main.py for all cleaning steps. These files are available on my github repo as well as in the google drive folder shared as part of this project. Main.py works by reading all CSV files in an input directory one by one, sending them through 2 cleaning functions, and saving them to an output directory. The screenshot for the main function is posted below.



```python
def main(args):
    # If you have a small dataset, consider uncommenting these two lines
    # If you do, you need to change the df.to_csv lines that use path.split.
    #df = read_tweets(args.data_dir)
    #df.to_csv(os.path.join(args.output_dir, 'concatenated_df.csv'), index=False)
    # Since our dataset is not very small, we will iteratively apply the cleaning process
    # To each CSV file.
    for path in glob.glob(os.path.join(args.data_dir, '*.csv')):
        # Read in one csv at a time and then remove duplicate Tweet IDs, then remove Retweets, and
        # save the CSV at each step to record progress.
        df = pd.read_csv(path)
        print("Read in " + path.split('/')[-1])
        df = remove_duplicates(df)
        df.to_csv(os.path.join(args.output_dir, "deduped_" + path.split('/')[-1]), index=False)
        print("Wrote " + path.split('/')[-1])
        df = remove_RT(df)
        df.to_csv(os.path.join(args.output_dir, "removed_RT_" + path.split('/')[-1]), index=False)
        del df
```

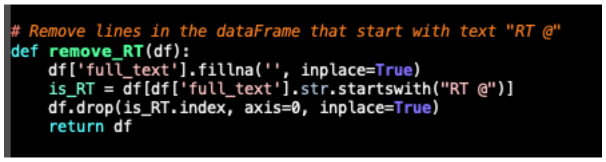
We see that the entire main function is wrapped in a for loop, iterating over the data directory. We read in one CSV at a time, remove duplicate tweets, save that csv, remove retweets, and save that CSV file with a new name. This function takes advantage of the pandas library (abbreviated as “pd”), as well as functions I wrote on my own called remove_duplicates and remove_RT. Next, we should examine those two functions.

The first function to look at is remove_duplicates, which removes duplicate tweets from our dataset. Duplicate tweets are defined as two tweets with the same “id” attribute. We remove all the duplicates by using pandas’ drop_duplicates function, making sure to only keep one copy of the duplicate. The screenshot below shows the entire remove_duplicates function. As a result of the remove_duplicate function, 6, 684 tweets were removed from my dataset.



```python
# Remove lines in the dataFrame with the same tweet ID
def remove_duplicates(df):
    df.drop_duplicates(subset='id', keep='first', ignore_index=True, inplace=True)
    return df
```

Next, we will look at the remove_RT function. The purpose of this function is to remove tweets which are simply retweets. That is, any tweet that starts with "RT @" will be removed since it is just a retweet and provides no additional information. Note that this function is called with the dataframe after all the duplicates have been dropped, so we are working solely with unique tweet IDs. We use pandas' "drop" functionality to drop any row which has text starting with "RT @". That way, any tweet in our dataset must contain its own content. As a result of the remove_RT function, 3,403,928 tweets were removed from our dataset.



```
# Remove lines in the dataFrame that start with text "RT @"
def remove_RT(df):
    df['full_text'].fillna('', inplace=True)
    is_RT = df[df['full_text'].str.startswith("RT @")]
    df.drop(is_RT.index, axis=0, inplace=True)
    return df
```

Challenges

The biggest challenge with this project was computation power and RAM. My dataset was very large and so my laptop alone did not have the RAM I needed to be able to work through all the tweets. Ideally, I would have grouped all the data together into one big dataframe because that would allow me to remove tweets that appear in more than one individual file. However, the conglomerate CSV file was around 30 GB and I could not process it on any resource I used. As a result, I had to use an EC2 instance and process each CSV file individually.

An additional challenge came with removing retweets. My method of removing retweets (removing tweets that begin with "RT @") only accounts for retweets that do not provide additional context, and does not account for something like quote tweets (i.e. Tweets that start with a user's own text and then refer to a particular tweet as a retweet). Should quote tweets be removed? They are retweets, but add a user's thoughts on top. In some sense, they are duplicate tweets, but not always. This provides an additional challenge to this project, and leaves it open for interpretation.