TWITTER ANALYSIS TOOL

(A TWITTER API TOOL FOR ANALYSIS)

By

Litha Thampan

PROJECT REPORT

Course : *CSIT 691 Independent Study- Spring 2020*

Instructor : *Dr. Dawei Li*

Table of Contents

1. Introduction

Twitter Analysis Tool is a standalone application to help data scientists gather and analyse tweets on a particular topic. This abstracts the complexity of finding the relevant hashtags and automatically detects and provides data for further data processing.

1.1. Purpose of the Document

The purpose of the document is to provide a detailed report of the project (Twitter Analysis Tool) to develop a tweet gathering tool.

1.2. Purpose of the System

Twitter is a highly popular social media site and are one of the best platforms for understanding trends of social sentiment on a particular topic. However, the data gathering process on a topic is very tedious as the trends and hashtags relevant to certain topic change on a daily basis. It is also possible that popular trends overpower the searches and makes the gathered data lean towards trending topics. Hashtags and User mentions of a particular topic might not be relevant in a reciprocal manner. For example, while searching for a less popular topic, a trending hashtag might appear a lot. But if the trending hashtag is searched, the original search word may not appear as much.

The purpose of this application is to programmatically gather only the relevant tweets of a particular topic and systematically adjust the gather process to capture new trends as it appears in twitter feeds. This application will also stop gathering tweets when a particular hashtag fades away which was at one point related to a topic. This happens when a celebrity mentions about the search topic. A lot of tweets will be gathered related to that person while that person tweets are trending. Gradually, the gather will be stopped, since other tweets from that id may not be relevant to the search word.

The application provides an easy interface to the data scientists to work with relevant tweets of a particular topic. Major insights can be made based on the several meta information each tweet contains. The data of the application is always kept in the pure raw form to allow users to take full use of the tweet data. The design is kept simple by keeping gathering, data preparation and analysis as three separate modules. This helps users to choose only the functionality according to their requirement.

2. Design

Following are design specifications of the system.

2.1. Overall Design

The system provides three major interactions through commands.

1. Gather – This gathers tweets based on a keyword.
2. Prepare – Deduplicated the data gathered up to that time
3. Analyse – Provides basic visualizations on the prepared data.

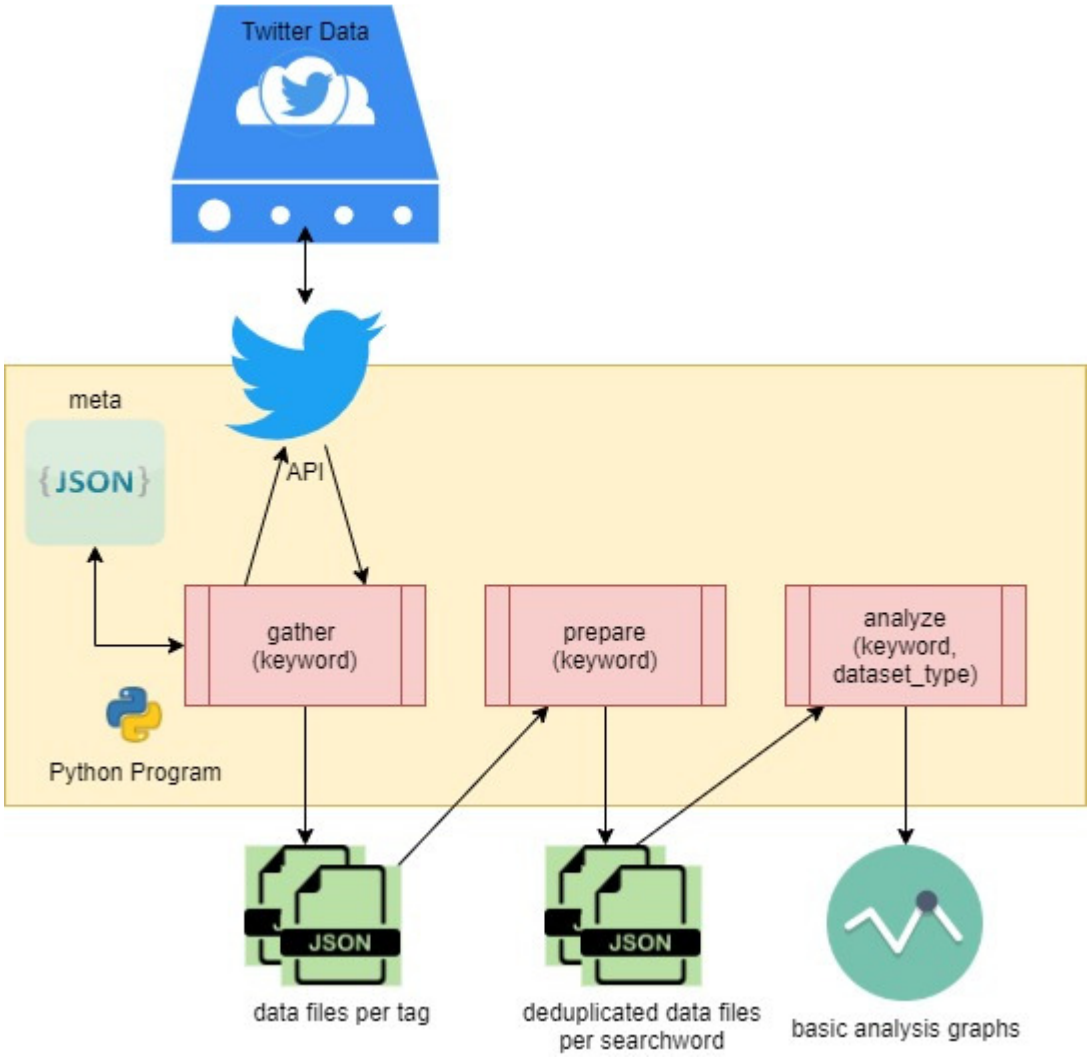

Figure 1: Overall Design

Figure 1 provides an overall design. The system interacts with Twitter API Host server using and API call implemented within the application. This interaction only happens during the gather phase. System keeps both data and metadata in json files and is named on the keyword used for the gather. For each keyword used during the execution a single meta file will be used to keep track of already gathered data and N number of files will be kept in data folder with the name starting with the keyword. These filenames will be also stored in the meta file. Prepare action creates deduplicated files with keyword as a prefix. Analyse action performs basic analysis and visualizations by looking at the prepared files starting with the keyword as the name.

Below section goes in detail about the algorithms used by each command.

2.2. Algorithms

Figure 2 shows the shapes used in the flowchart to explain each algorithm.

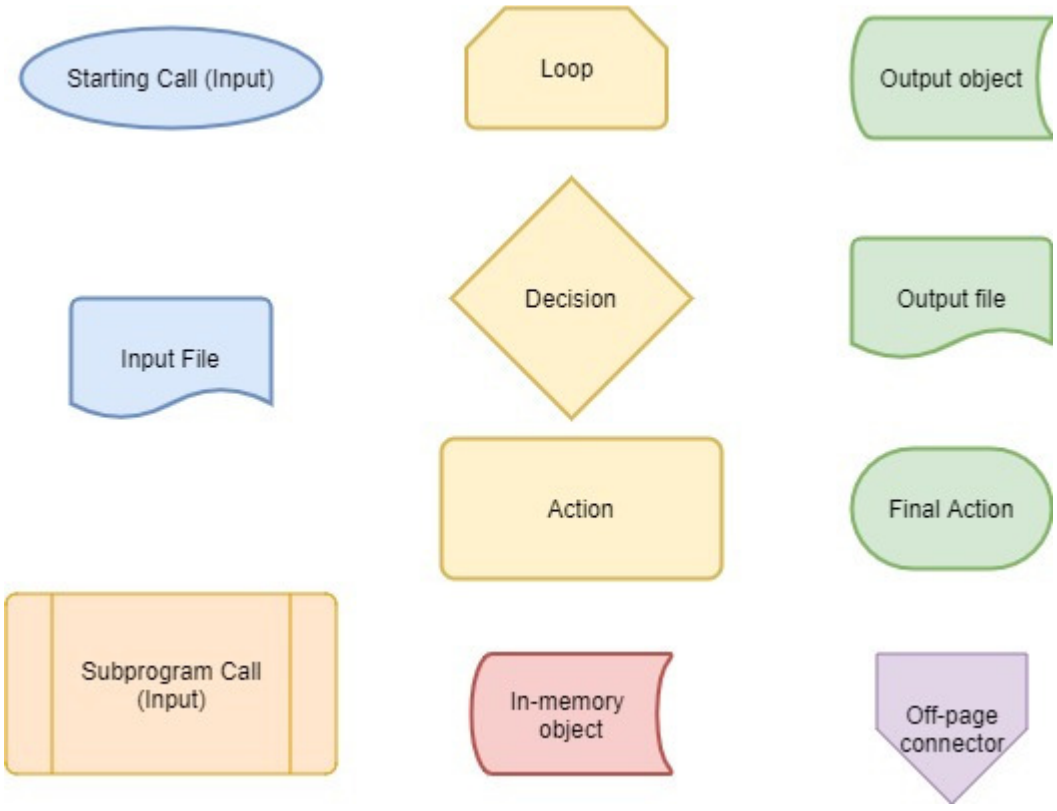

Figure 2: Legend

2.2.1. Gather

Figure 3 shows overall data flow in the gather algorithm. Each keyword meta is preserved in separate meta files which gets updated each time the keyword is gathered for more data. This helps in seamless search on a particular keyword.

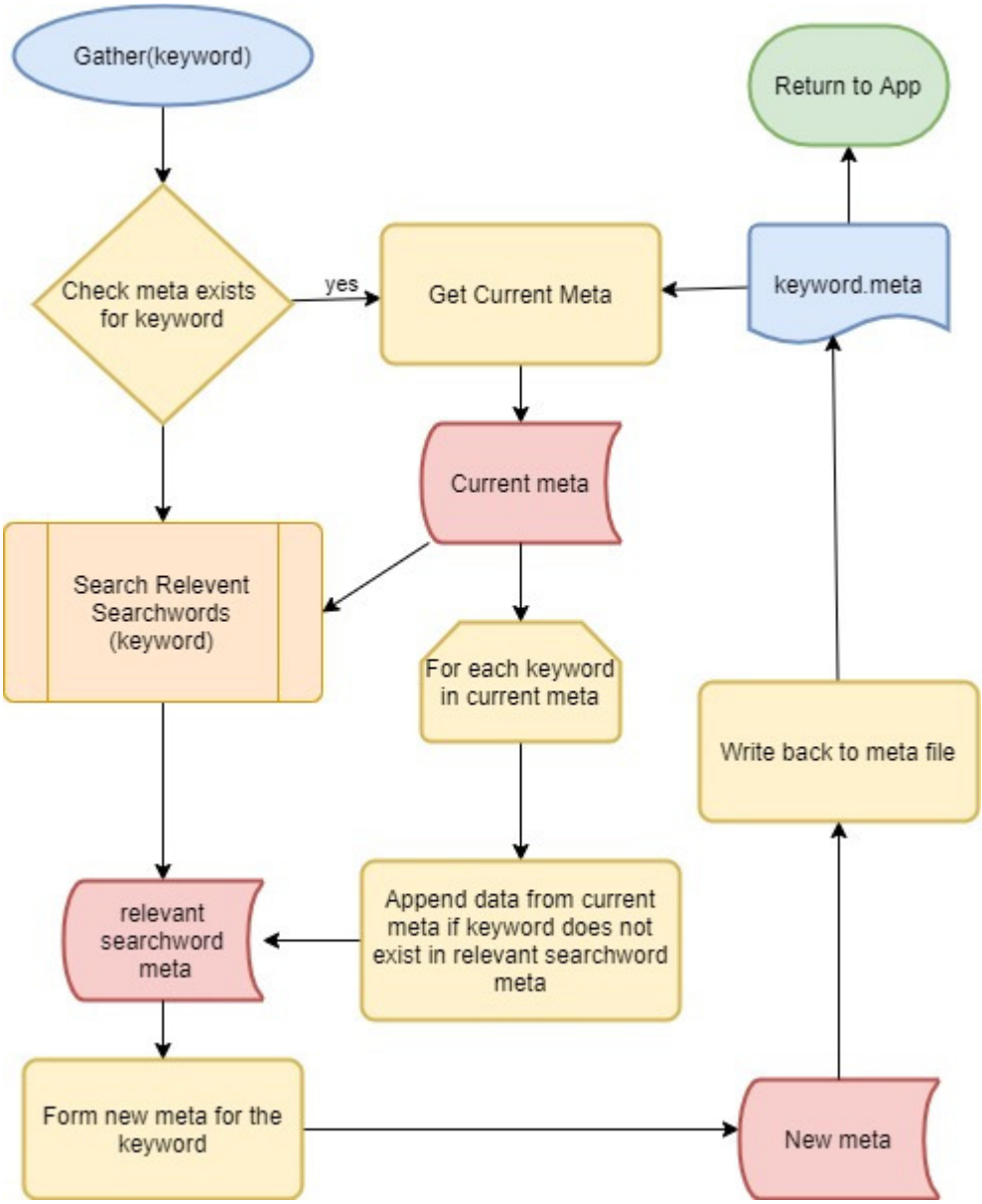

Figure 3: Gather Algorithm

Figure 4 and Figure 5 provides the logic of the submodule in which different searchwords (hashtags and user mentions) are selected for a keyword and the criteria on which the data is selected for the dump files.

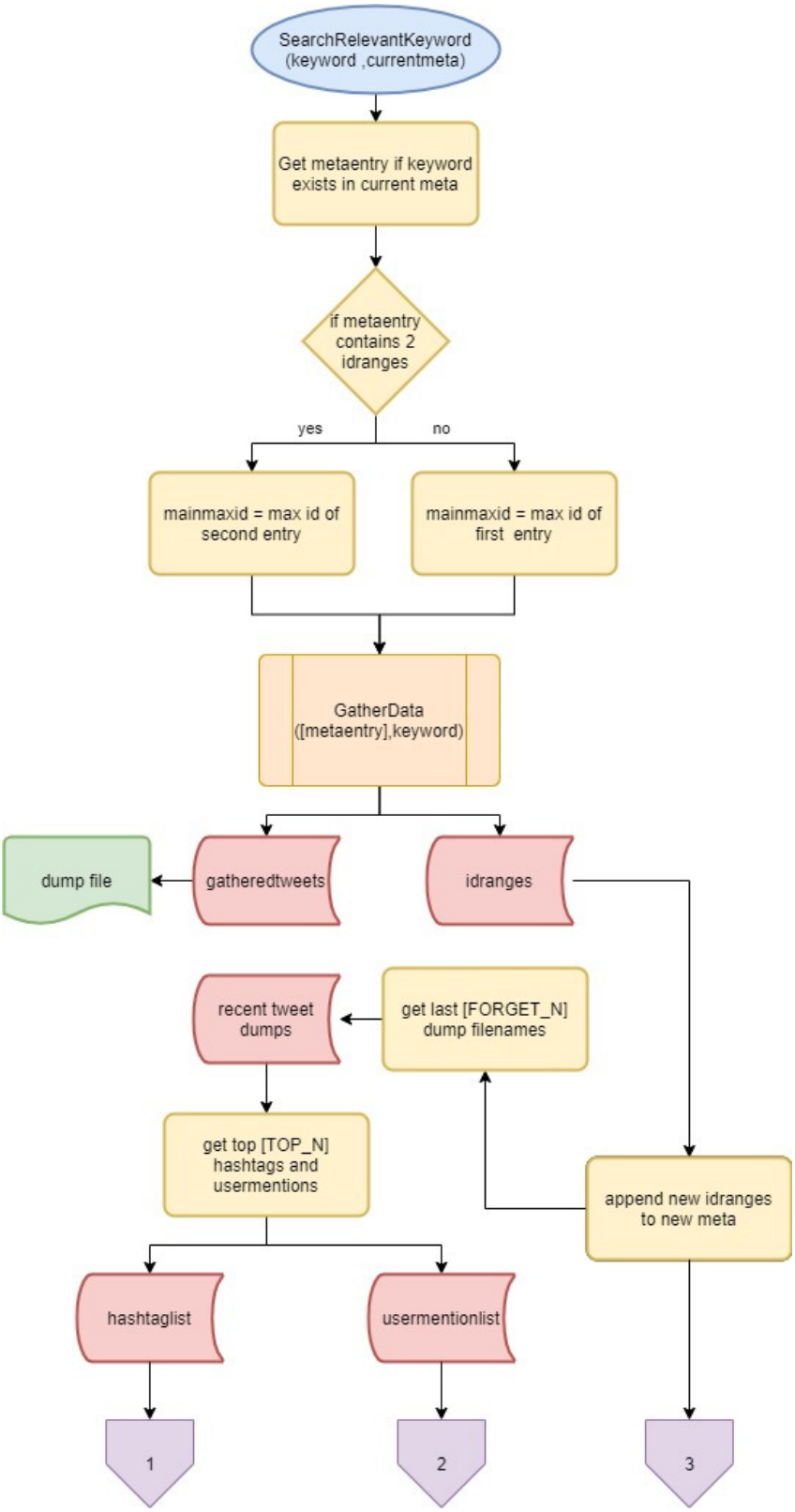

Figure 4: Search Relevant Searchwords Module (Part1)

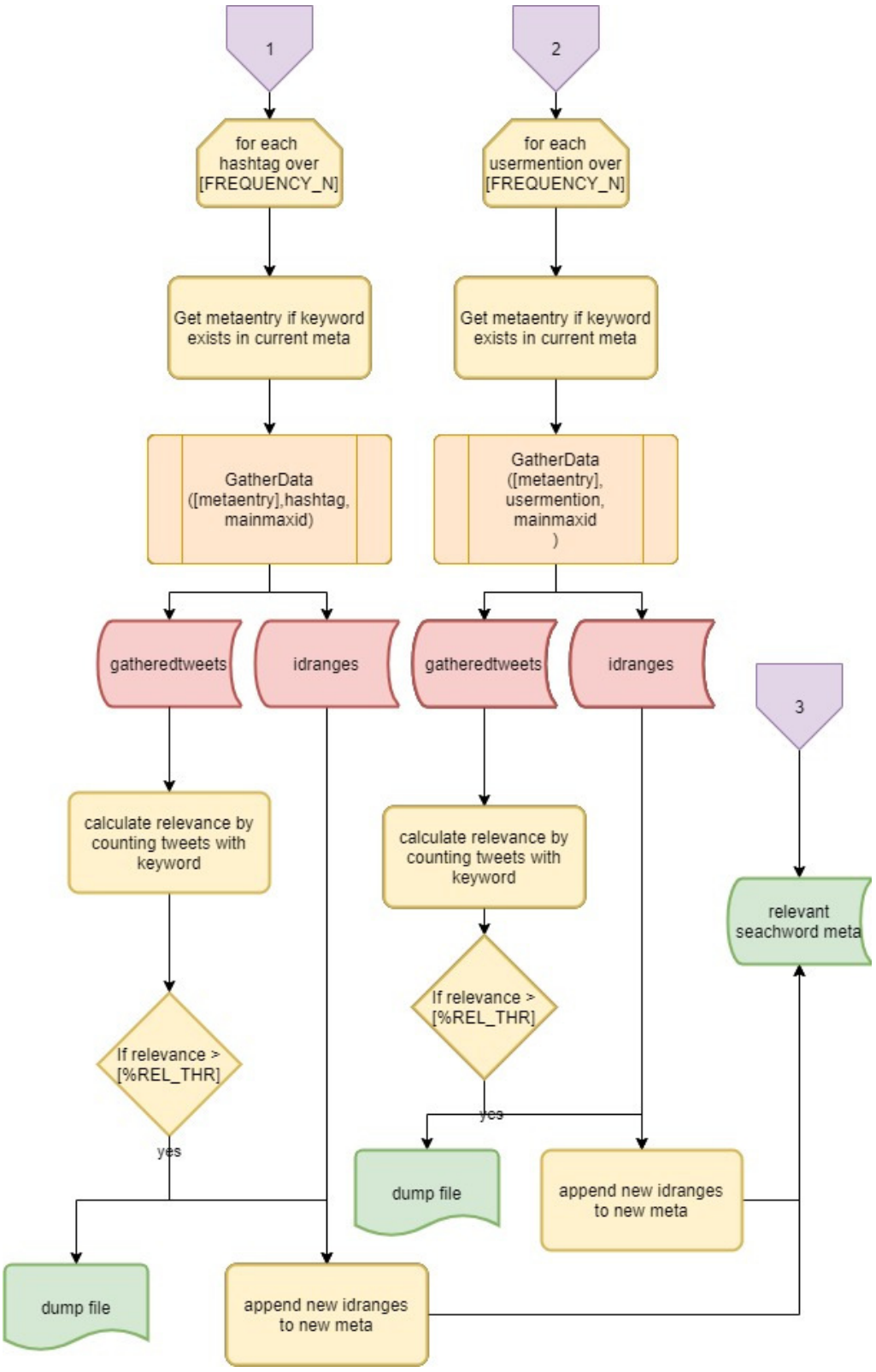

Figure 5: Search Relevant Searchwords Module (Part2)

Figure 6 showcases the submodule which handles the max_id and since_id logic of twitter api and making sure the maximum tweets are collected for handling searches by the user during irregular intervals.

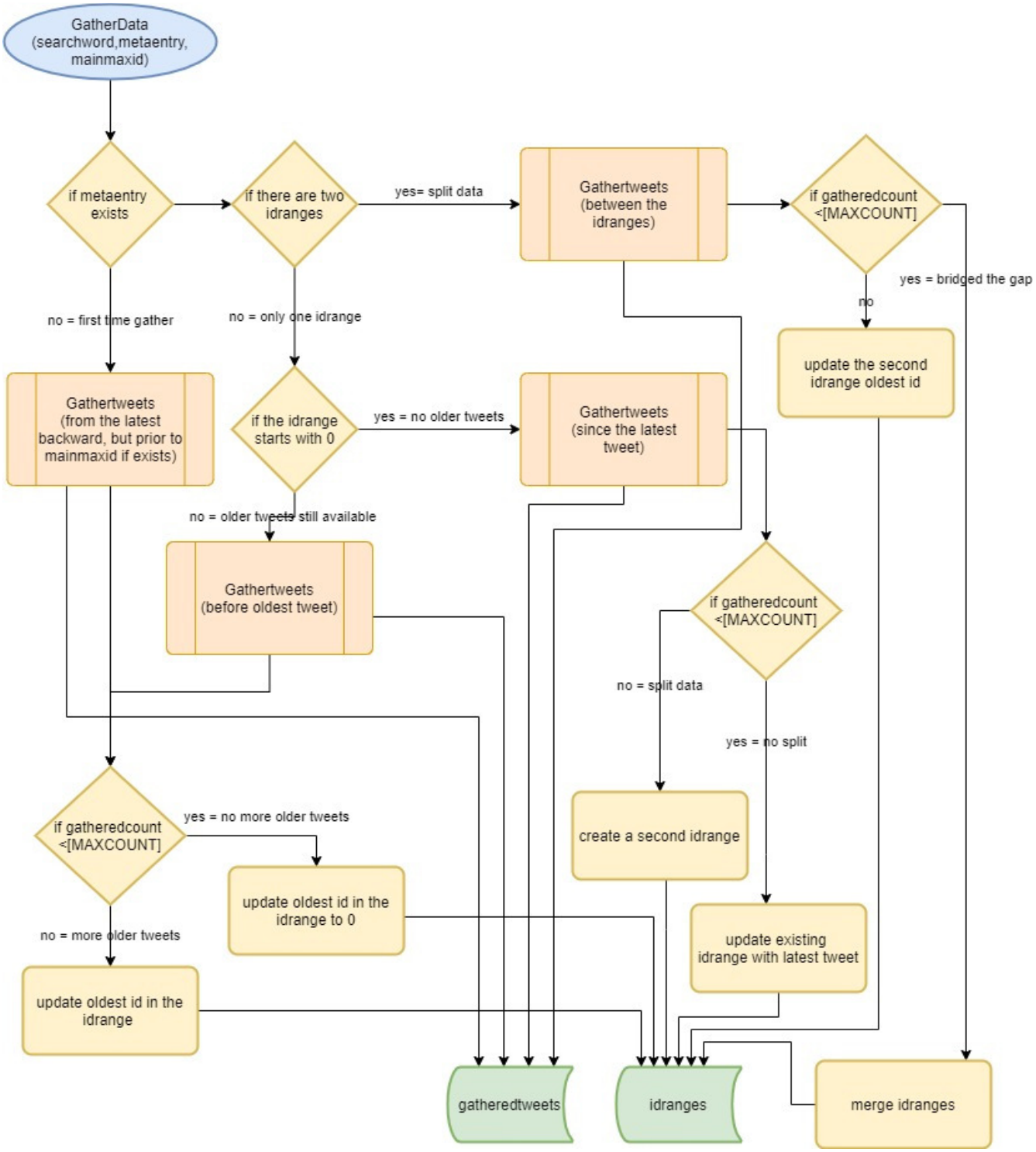

Figure 6: Gather Data Module

The concept of this logic revolves around timeline logic from twitter API. As shown in Figure 7, When the search word is queried for the first time, the latest tweets are gathered. The Meta File stores the max_id, which is calculated as the id prior to the oldest tweet gathered. In the diagram, it is calculated as 6 (Oldest TweetID) – 1 = 5.

When the data is queried again, the max_id parameter is passed so that it gathers available tweets less than or equal to max_id. This helps in a continuous gather of the tweet and reduces any duplicate gather. In the below scenario, max_id = 5 is passed to gather tweets older than Tweet5.The gather algorithm will continue to gather older tweets until there are no older tweets available.

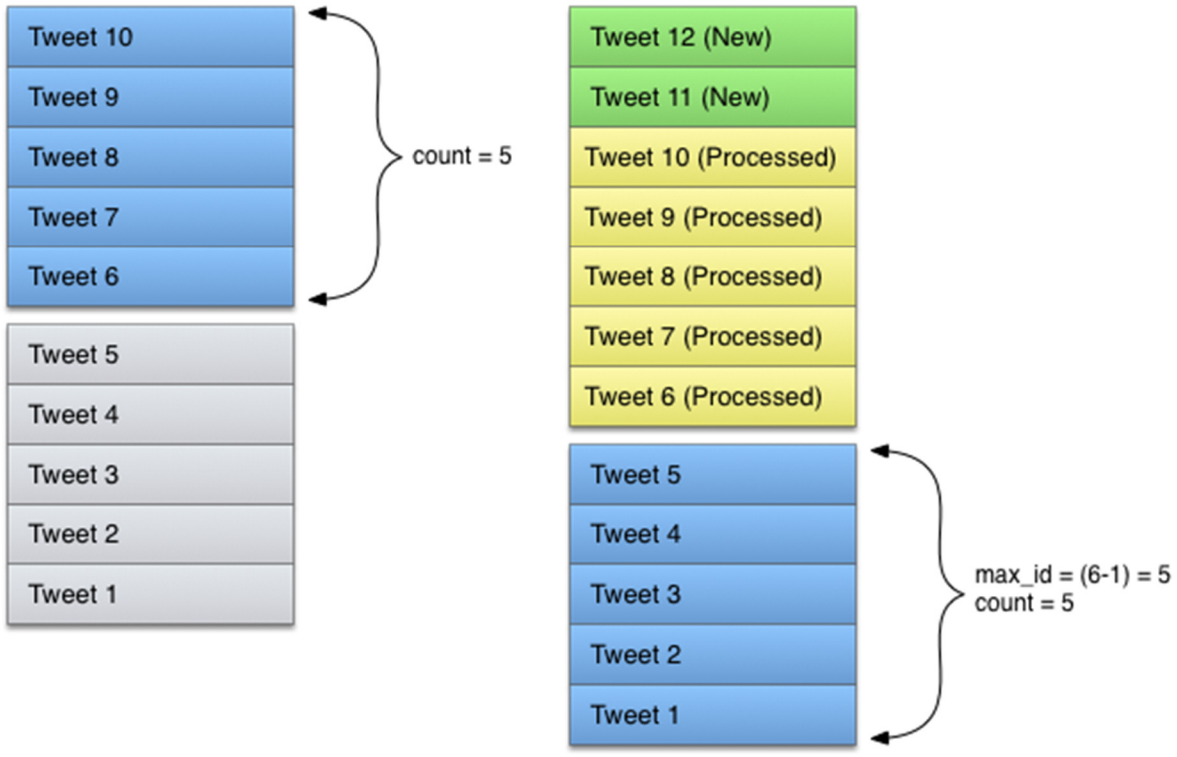

Figure 7: Max_ID Logic [1]

Once there are no older tweets available, the tweets happened after the first gather should be queried. In the case of the diagram, it is any tweets that happened after Tweet10. This is where the since_id parameter is used. When the algorithm gathers for the first time, the Latest Tweet ID is recorded as since_id. In the case of this diagram it is Tweet10 = 10. Once all the older tweets that can be gathered are collected, algorithm comes back to collect tweets with since_id = 10.

Figure 8 shows how the since_id logic works in tandem with max_id logic. It often happens that there are a lot of tweets that would have been created after the stored since_id in Meta. So, it will gather the latest N number of tweets after the tweet of since_id. In the diagram, it can be observed that the Tweet18 to Tweet14 is gathered. This created a gap between the gathered tweets. This is accommodated in the metafile as it can hold two id-ranges. Next time the algorithm gathers the data, it uses since-id as 10 and max_id as 13, so that it gathers data in the gap. This helps in filling the gap. This is continued until there are no more tweets available

in the gap. Once the gap is filled, both id ranges are merged and the since_id is updated to the latest tweeted gathered so far. This approach helps to gather the maximum number of continuous tweets in a timeline.

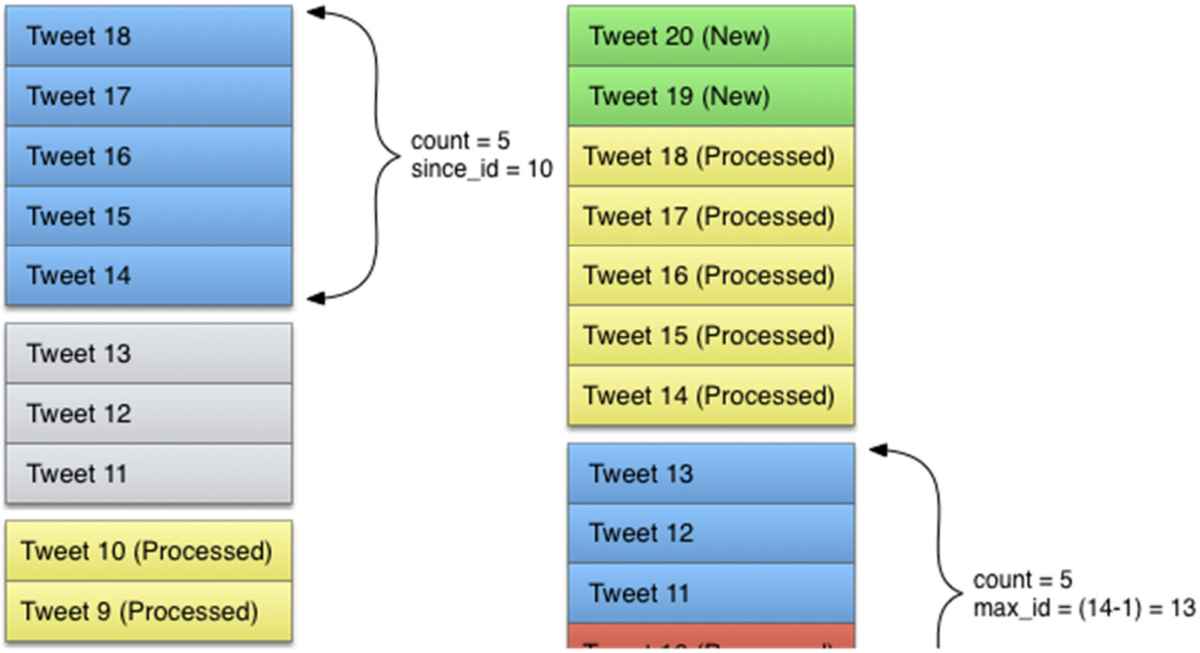

Figure 8: Since_ID [1]

Figure 9 shows the algorithm of submodule that makes the twitter API calls. These calls are made in a recurring manner while handling the 15-minute rate limit internally.

Batchcount used in the application is the maximum allowed limit by twitter which is 100 tweets. If retrieved tweet count is equal to batch count, it indicates that there is more tweet to be gathered. The max_id and since_id explained in the above section is used to gather the tweets in the most efficient manner.

Rate limiting is used by twitter to reduce load on the Twitter API. This means that there is a limit of queries that are allowed within a rate limit window, which is set as 15 minutes by Twitter. Twitter allows 450 twitter search queries within a rate limit window. Application handles the rate limiting by automatically sleeping for 30 second intervals whenever the rate limit window is encountered. This keeps the program running but reduces the resource utilization during the rate limit wait time.

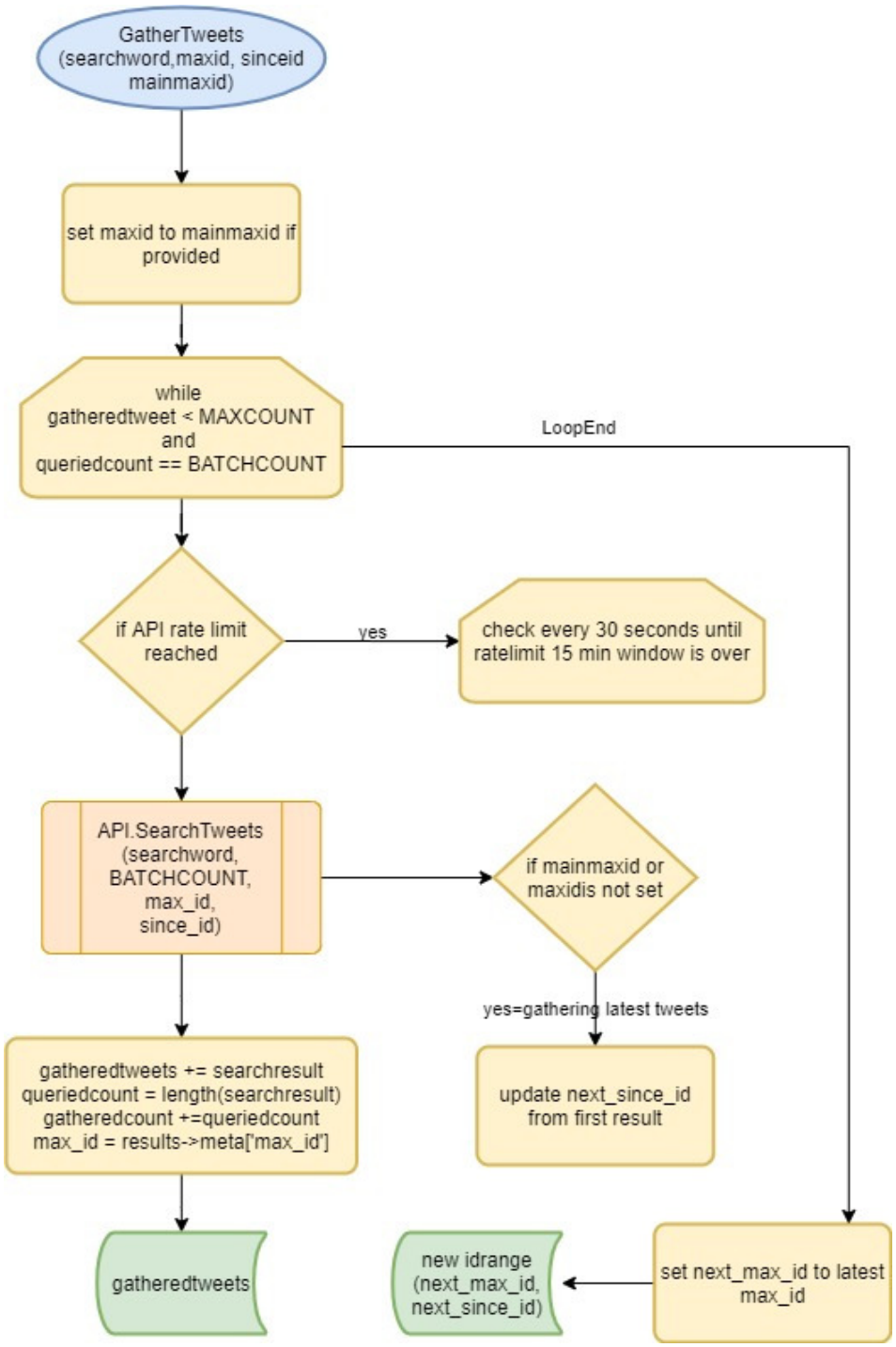

Figure 9: Gather Tweet Module

2.2.2. Prepare

Figure 10 demonstrates the prepare algorithm which removes the duplicates of the data and creates two datasets, **major** (with a continuous timeline) and **full** (with all retrieved data).

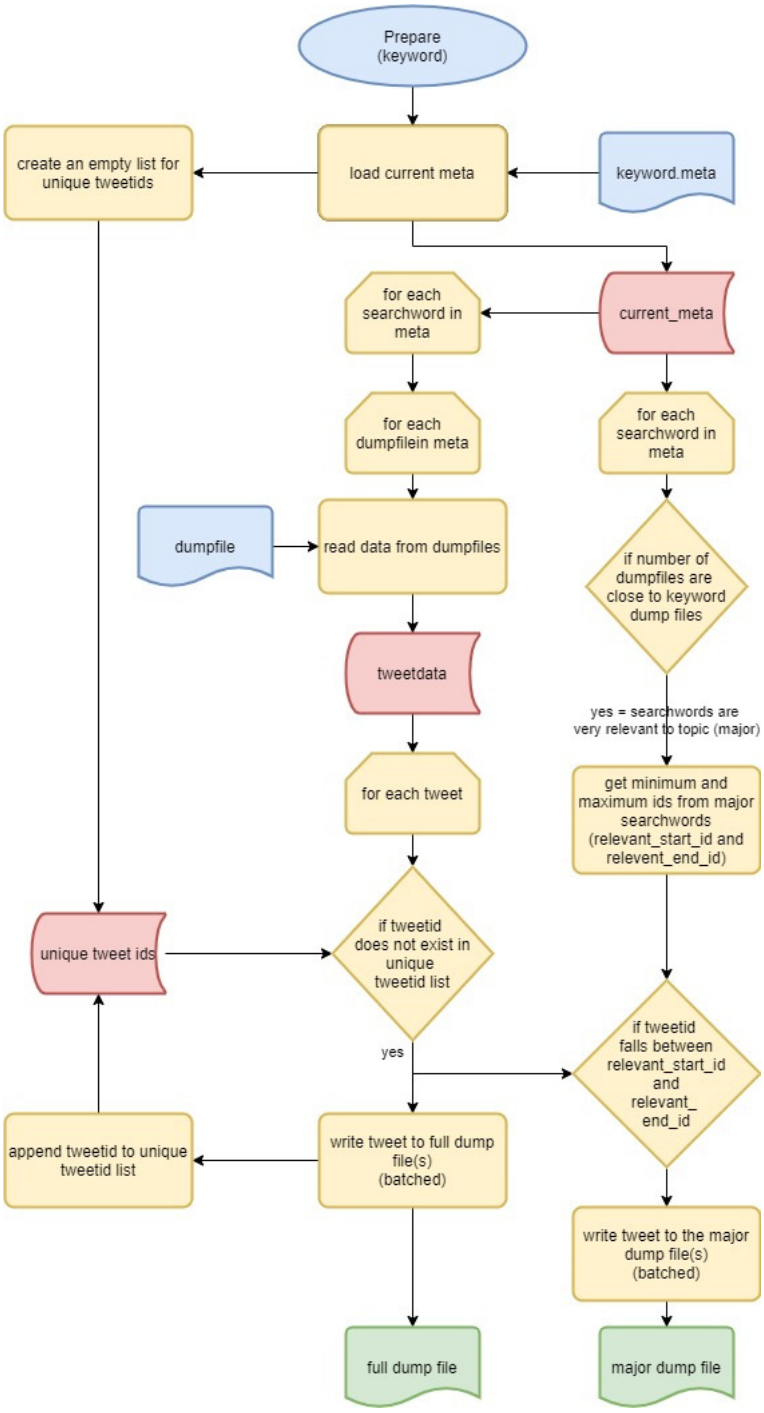

Figure 10: Prepare Algorithm

2.2.3. Analyze

Figure 11 shows the analyse algorithm which loads the prepared data in memory and creates counter objects for each dimension. Sample plots are created for each of this counter.

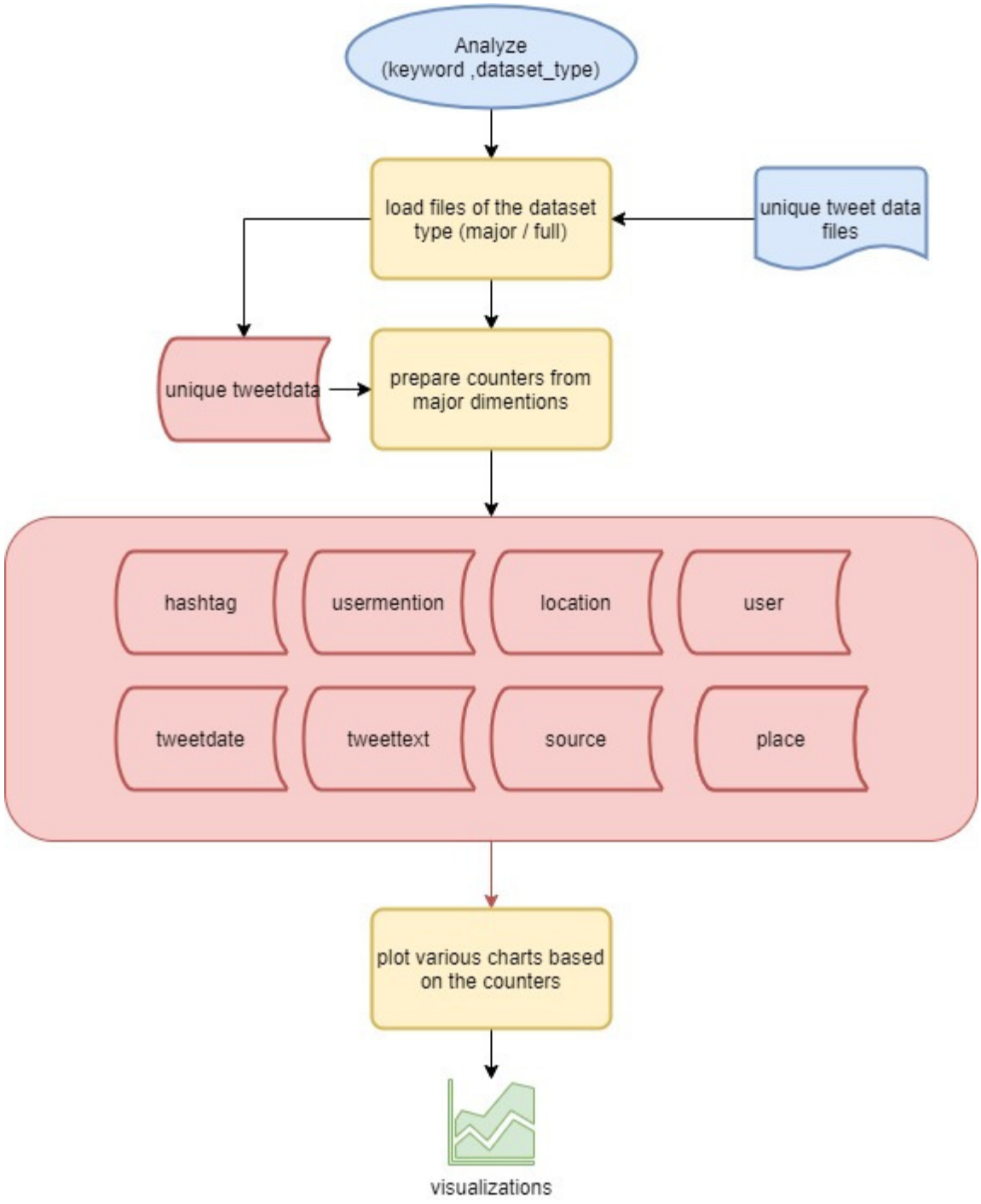

Figure 11: Analyse Algorithm

3. Environment Set-Up

Twitter Analysis Tool is built on Python and uses twitter api modules to interact with API. The user should have a twitter developer account in order to use the tool. Following are the environment set up activities that need to be completed for usage or further development of the tool. All the below instructions are tested with Linux operating system.

3.1. Twitter Developer Account and Application

For tweet searching, the user should have a twitter developer account. Follow the below instructions to create one.

1. Create a twitter account if you don't have one.
2. Log in at https://developer.twitter.com/ with your Twitter username and password.
3. Click Apply
4. Fill in the details about how you are planning to use the twitter data
5. After the review by twitter (1-2 days), you will receive notification on approval of developer account.

Once the developer account is set, proceed with the following for creating and application which will help you create credentials to use data.

1. Login on the twitter development account
2. Click on Create App
3. Fill in application details and Save
4. This will generate consumer key and secret which will be used for application-only authentication in the Tool.

3.2. Python Dependencies

Create a python environment with python 3.8. For the development virtualenvwrapper was used to isolate python version (For Windows, use virtualenvwrapper-win). Following statement can be used to create and switch to a virtual environment.

```
mkvirtualenv -p python3.8 twitteranalysis
```

Python dependencies are mentioned in the requirements.txt file. Following statement can be executed to install the python dependencies

```
pip install -r requirements.txt
```

For ease of execution, the Twitter credentials (created in the 3.1) can be set in the environment variable using following command (applicable to Linux environment).

```
source setup.sh
```

For Windows, environment variable would need to be set using "`set`" command or through System Properties.

4. Execution and Test Results

Once the dependencies are installed, the application can be executed as follows

```
python twitterapp.py <command> <keyword> <optional parameters>
```

Following sections provide sample executions of the different commands.

4.1. Gather

Gather process is initiated with the following command line execution as shown in Figure 12.

```
python twitterapp.py gather <keyword>
```

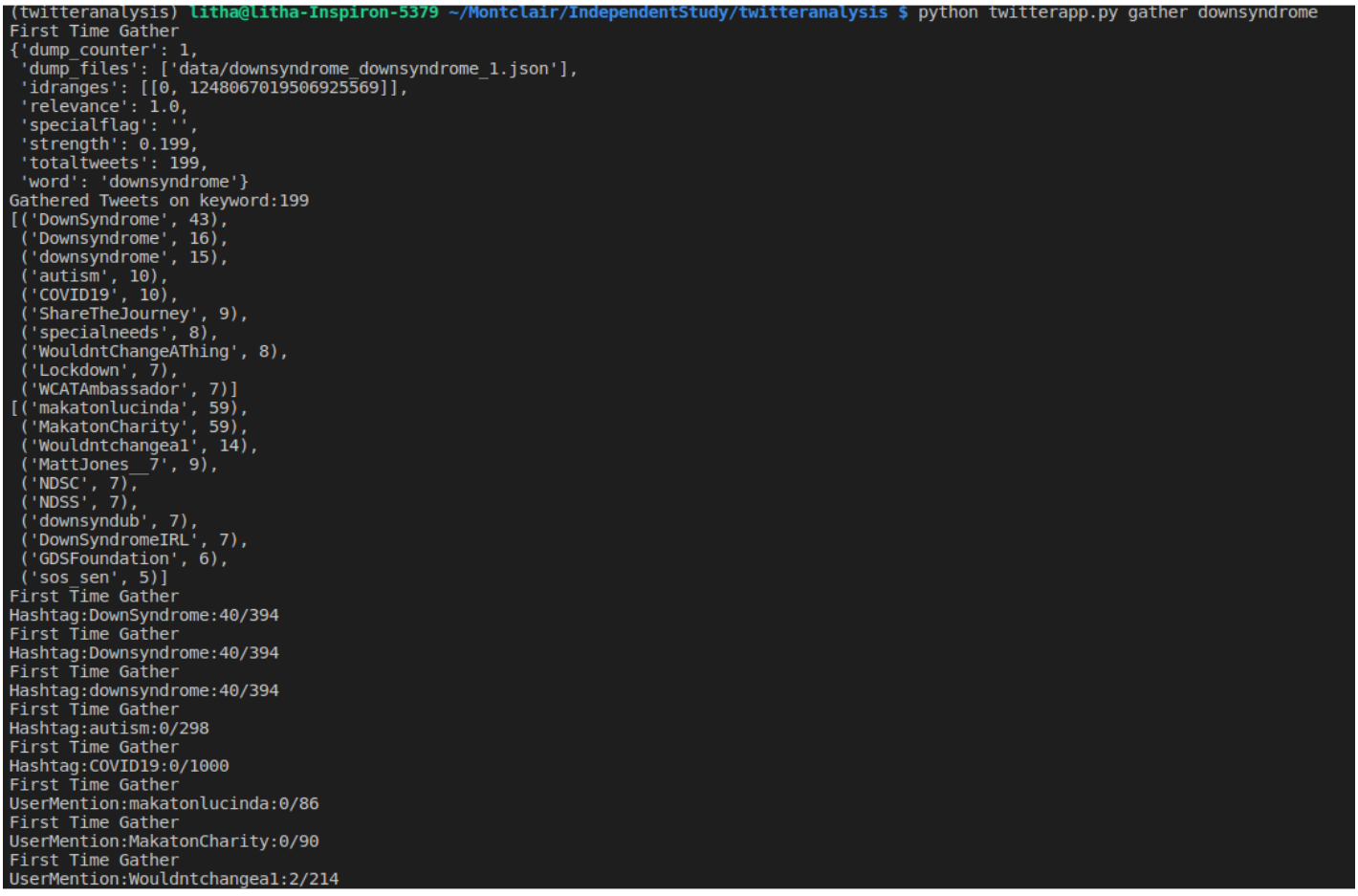

Figure 12: Gather Execution

When the Rate Limit is exceeded, it waits 30 seconds intervals until the rate limit window opens up again as shown in Figure 13.

```
(twitteranalysis) litha@litha-Inspiron-5379 ~/Montclair/IndependentStudy/twitteranalysis $ python twitterapp.py gather coronavirus
Bridging the gap
Rate Limit Exceeded.Try Again Later!!
EndpointRateLimit(limit=450, remaining=0, reset=1586402014)
Rate Limit Exceeded.Trying again after 30 sec
Rate Limit Exceeded.Trying again after 30 sec
Rate Limit Exceeded.Trying again after 30 sec
```

Figure 13: Rate Limit waiting

Gather process creates a meta file under the meta folder. This folder keeps track of all the keyword searches done by the user as shown in Figure 14.

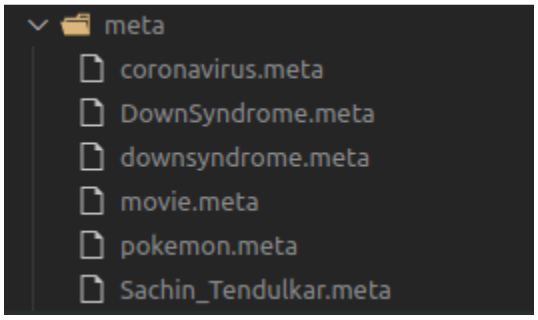

Figure 14: Meta folder content

Meta File contains the last gathered time and search word list. It keeps track of gathered twitter id ranges and the data files as shown in Figure 15.

meta > downsyndrome.meta

```
{
    "metagmttime": "2020-04-09 02:00:44",
    "metakeyword": "downsyndrome",
    "relevant_searchwords": [
        {
            "specialflag": "",
            "word": "downsyndrome",
            "relevance": 1.0,
            "strength": 0.199,
            "totaltweets": 199,
            "idranges": [
                [
                    0,
                    1248067019506925569
                ]
            ],
            "dump_counter": 1,
            "dump_files": [
                "data/downsyndrome_downsyndrome_1.json"
            ]
        },
        {
            "specialflag": "#",
            "word": "DownSyndrome",
            "relevance": 0.10152284263959391,
            "strength": 0.394,
            "totaltweets": 394,
            "idranges": [
                [
                    0,
                    1248067019506925569
                ]
            ],
            "dump_counter": 1,
            "dump_files": [
                "data/downsyndrome_#DownSyndrome_1.json"
            ]
        },
```

Figure 15: Meta file Content

Datafiles are stored under the data folder as shown in Figure 16. Files names are in the following format.

`<keyword>_<searchword>_<counter>.json`

```
(twitteranalysis) litha@litha-Inspiron-5379 ~/Montclair/IndependentStudy/twitteranalysis $ ls -ltr ./data/coronavirus*
-rw-rw-r-- 1 litha litha 5806434 Mar  6 20:42 ./data/coronavirus_coronavirus_1.json
-rw-rw-r-- 1 litha litha 6349869 Mar  6 20:42 ./data/coronavirus_#coronavirus_1.json
-rw-rw-r-- 1 litha litha 6390746 Mar  6 20:42 ./data/coronavirus_#Coronavirus_1.json
-rw-rw-r-- 1 litha litha 6795913 Mar  6 20:42 ./data/coronavirus_#COVID19_1.json
-rw-rw-r-- 1 litha litha 4666449 Mar  6 20:42 ./data/coronavirus_@elonmusk_1.json
-rw-rw-r-- 1 litha litha 6428964 Mar  6 20:43 ./data/coronavirus_@jbouie_1.json
-rw-rw-r-- 1 litha litha 5403367 Mar  6 20:43 ./data/coronavirus_@CNN_1.json
-rw-rw-r-- 1 litha litha 5494893 Mar  6 20:43 ./data/coronavirus_@funder_1.json
-rw-rw-r-- 1 litha litha 6481067 Mar  6 20:46 ./data/coronavirus_@whatchidid_1.json
-rw-rw-r-- 1 litha litha 5860097 Mar  6 20:50 ./data/coronavirus_coronavirus_2.json
-rw-rw-r-- 1 litha litha 6511337 Mar  6 20:51 ./data/coronavirus_#coronavirus_2.json
-rw-rw-r-- 1 litha litha 6506060 Mar  6 20:51 ./data/coronavirus_#Coronavirus_2.json
-rw-rw-r-- 1 litha litha 6696941 Mar  6 20:51 ./data/coronavirus_#COVID19_2.json
-rw-rw-r-- 1 litha litha 4664129 Mar  6 20:51 ./data/coronavirus_@elonmusk_2.json
-rw-rw-r-- 1 litha litha 5698575 Mar  6 20:51 ./data/coronavirus_@jbouie_2.json
-rw-rw-r-- 1 litha litha 5442110 Mar  6 20:51 ./data/coronavirus_@funder_2.json
-rw-rw-r-- 1 litha litha 6456455 Mar  6 20:51 ./data/coronavirus_@whatchidid_2.json
-rw-rw-r-- 1 litha litha 5085011 Mar  6 20:51 ./data/coronavirus_@kumailn_1.json
-rw-rw-r-- 1 litha litha 6001272 Mar  6 21:00 ./data/coronavirus_coronavirus_3.json
-rw-rw-r-- 1 litha litha 6444109 Mar  6 21:00 ./data/coronavirus_#coronavirus_3.json
```

Figure 16: Data folder content

Data File contains tweets stored as a JSON array as shown in Figure 17. These are unaltered data coming from the twitter api. Since the data files are kept as pure json for each tweet, any application that works with tweet data can start directly interacting with these files.

data > {} DownSyndrome_#Makaton_1.json > ...


```
[
    {
        "created_at": "Wed Mar 04 03:37:27 +0000 2020",
        "id": 1235046672499118080,
        "id_str": "1235046672499118080",
        "full_text": "RT @makatonlucinda: Makaton Sign of the Week - 'Story' \ud83d\udcda\n\nA sign ready for #WorldBookDay !\n\n#Makaton #WeTalkMakaton #Story
        #DownSyndrome #W\u2026",
        "truncated": false,
        "display_text_range": [
            0,
            140
        ],
        "entities": {
            "hashtags": [
                {
                    "text": "WorldBookDay",
                    "indices": [
                        75,
                        88
                    ]
                },
                {
                    "text": "Makaton",
                    "indices": [
                        92,
                        100
                    ]
                },
                {
                    "text": "WeTalkMakaton",
                    "indices": [
                        101,
                        115
                    ]
                },
                {
                    "text": "Story",
                    "indices": [
                        116,
                        122
                    ]
                },
                {
                    "text": "DownSyndrome",
                    "indices": [
                        123,
                        136
                    ]
                }
```


Figure 17: Data file content

4.2. Prepare

Prepare process is initiated with the following command line execution as shown in Figure 18.

```
python twitterapp.py prepare <keyword>
```

```
(twitteranalysis) litha@litha-Inspiron-5379 ~/Montclair/IndependentStudy/twitteranalysis $ python twitterapp.py prepare downsyndrome
[{'dump_counter': 1,
  'dump_files': ['data/downsyndrome_downsyndrome_1.json'],
  'idranges': [[0, 1248067019506925569]],
  'relevance': 1.0,
  'specialflag': '',
  'strength': 0.199,
  'totaltweets': 199,
  'word': 'downsyndrome'},
 {'dump_counter': 1,
  'dump_files': ['data/downsyndrome_#DownSyndrome_1.json'],
  'idranges': [[0, 1248067019506925569]],
  'relevance': 0.10152284263959391,
  'specialflag': '#',
  'strength': 0.394,
  'totaltweets': 394,
  'word': 'DownSyndrome'},
 {'dump_counter': 1,
  'dump_files': ['data/downsyndrome_#Downsyndrome_1.json'],
  'idranges': [[0, 1248067019506925569]],
  'relevance': 0.10152284263959391,
  'specialflag': '#',
  'strength': 0.394,
  'totaltweets': 394,
  'word': 'Downsyndrome'},
 {'dump_counter': 1,
  'dump_files': ['data/downsyndrome_#downsyndrome_1.json'],
  'idranges': [[0, 1248067019506925569]],
  'relevance': 0.10152284263959391,
  'specialflag': '#',
  'strength': 0.394,
  'totaltweets': 394,
  'word': 'downsyndrome'}]
0
1248067019506925569
Working withdata/downsyndrome_downsyndrome_1.json
Working withdata/downsyndrome_#DownSyndrome_1.json
Working withdata/downsyndrome_#Downsyndrome_1.json
Working withdata/downsyndrome_#downsyndrome_1.json
Dumping prepped/downsyndrome_full_0.041.json
Dumping prepped/downsyndrome_major_0.041.json
(twitteranalysis) litha@litha-Inspiron-5379 ~/Montclair/IndependentStudy/twitteranalysis $
```

Figure 18: Prepare Execution

Prepare create two sets of files.

- Full – Deduplicated set of all the tweets gathered by the Gather process for the keyword.
- Major – Deduplicated set of all the tweets between the timeframe where all major searchwords data is present in the whole set of gathered tweets.

If the Prepare process is rerun, the existing set of prepped files are removed before recreating them as shown in Figure 19.

```
removingprepped/coronavirus_full_6.json
removingprepped/coronavirus_full_24.json
removingprepped/coronavirus_full_31.json
removingprepped/coronavirus_major_6.json
removingprepped/coronavirus_major_13.5318.json
removingprepped/coronavirus_full_4.json
Working withdata/coronavirus_coronavirus_1.json
Working withdata/coronavirus_coronavirus_2.json
Working withdata/coronavirus_coronavirus_3.json
Working withdata/coronavirus_coronavirus_4.json
Working withdata/coronavirus_coronavirus_5.json
Working withdata/coronavirus_coronavirus_6.json
Working withdata/coronavirus_coronavirus_7.json
Working withdata/coronavirus_coronavirus_8.json
Working withdata/coronavirus_coronavirus_9.json
Working withdata/coronavirus_coronavirus_10.json
Dumping prepped/coronavirus_full_1.json
Dumping prepped/coronavirus_major_1.json
Working withdata/coronavirus_coronavirus_11.json
Working withdata/coronavirus_coronavirus_12.json
Working withdata/coronavirus_coronavirus_13.json
Working withdata/coronavirus_coronavirus_14.json
Working withdata/coronavirus_coronavirus_15.json
Working withdata/coronavirus_coronavirus_16.json
Working withdata/coronavirus_coronavirus_17.json
```

Figure 19 : Prepare re-execution

4.3. Analyze

Analyze process is initiated with the following command line execution as shown in Figure 20.

```
python twitterapp.py analyze <keyword> <optional parameters>
```

```
(twitteranalysis) litha@litha-Inspiron-5379 ~/Montclair/IndependentStudy/twitteranalysis $ python twitterapp.py analyze DownSyndrome --analyze_type full
Loadingprepped/DownSyndrome_full_1.json
185966
```

Figure 20: Analyze execution

First graph provided by the module is Top Hashtags. Most popular 50 Hashtag about the keyword is displayed as a bar graph as shown in Figure 21.

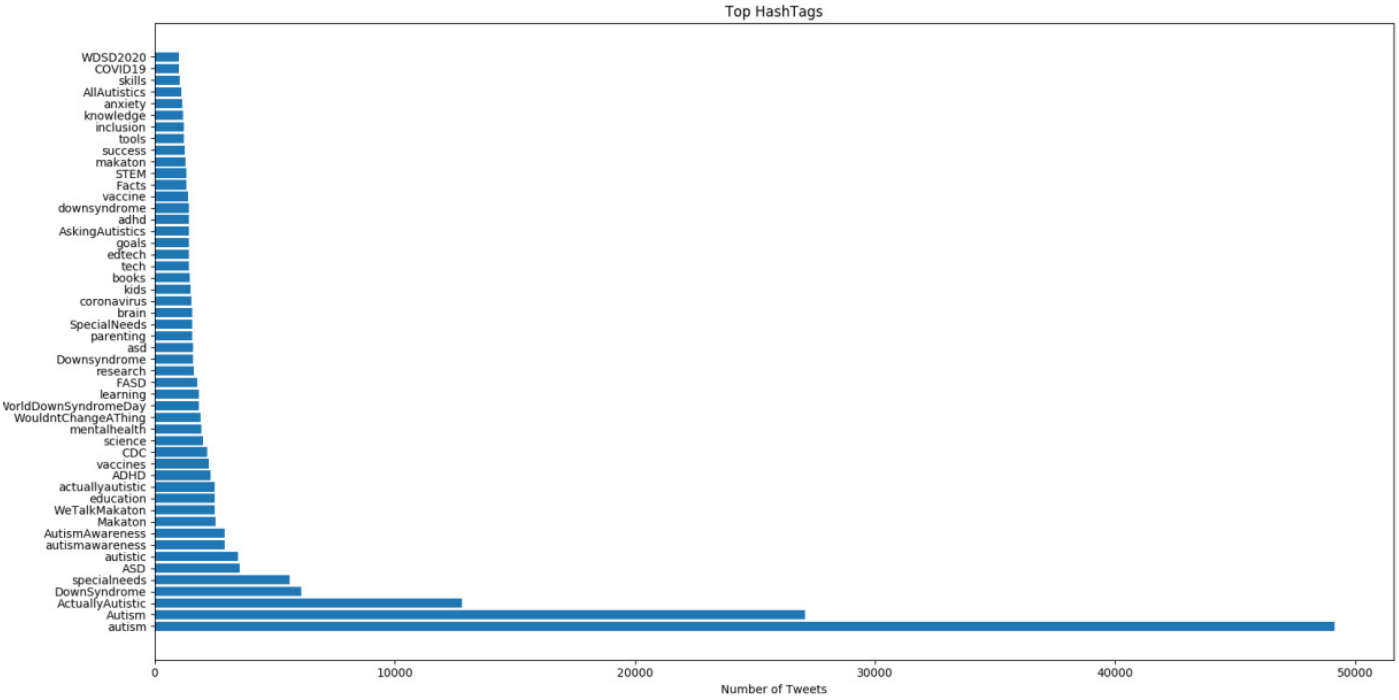

Figure 21: Top Hashtags graph

Next graph shown is the top tagged locations among the tweets as shown in Figure 22. Only a very few tweets are geo-tagged by the tweeters, so large datasets are required to make use of this graph.

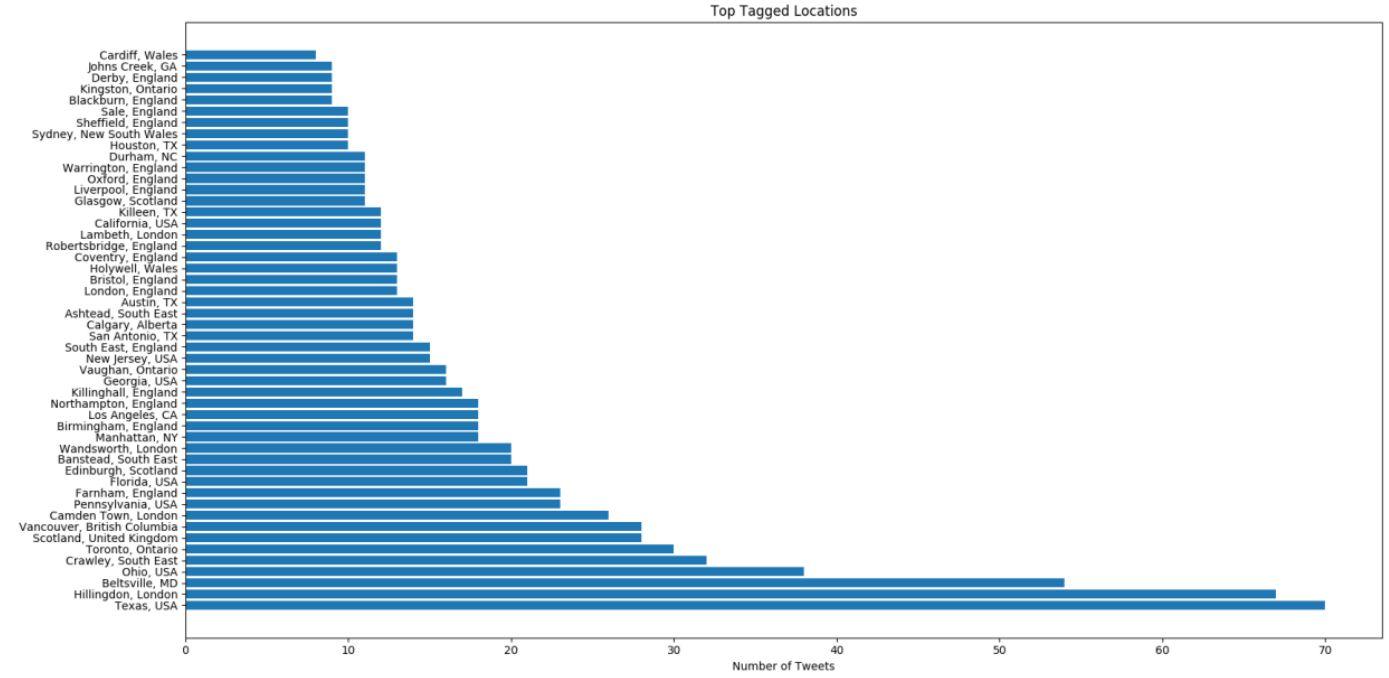

Figure 22: Top Tagged Locations graph

Next graph shown is the top user profile locations among the tweeters as shown in Figure 23.

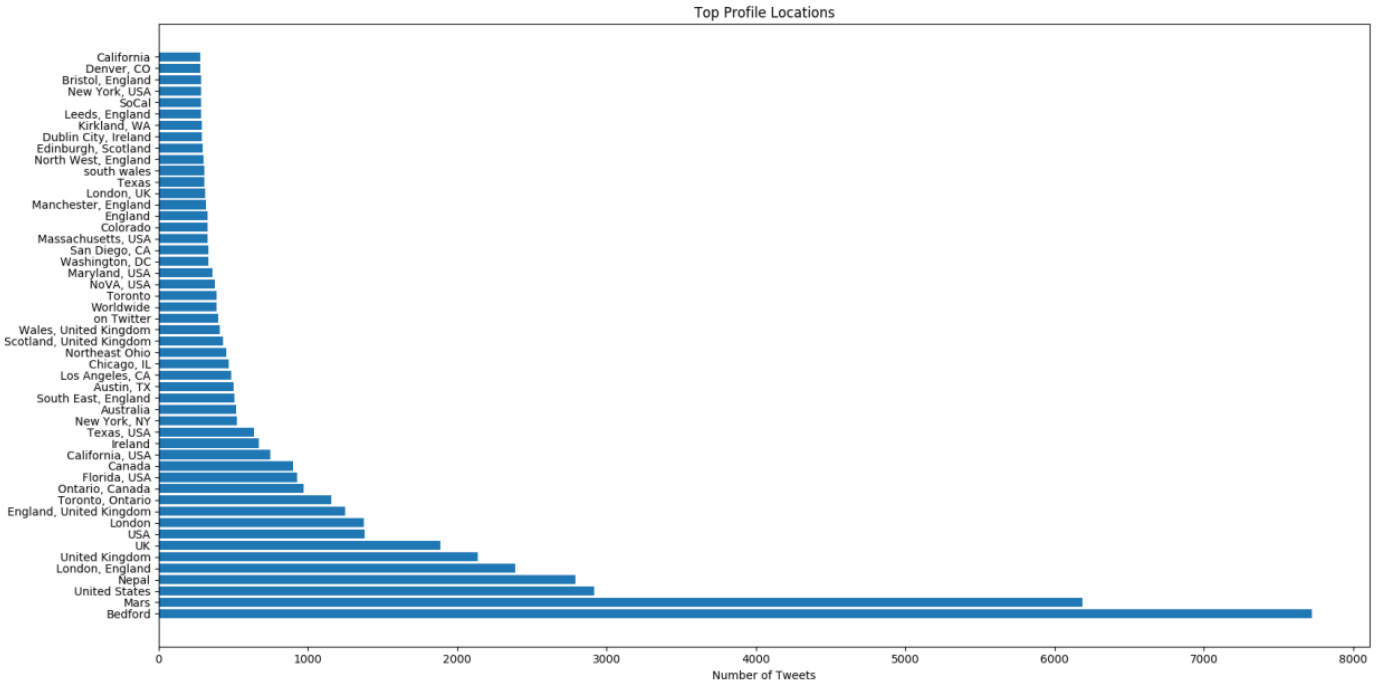

Figure 23: Top Profile Locations graph

Next graph shown the twitter user tags that provided the most tweets in the collected dataset as shown in Figure 24.

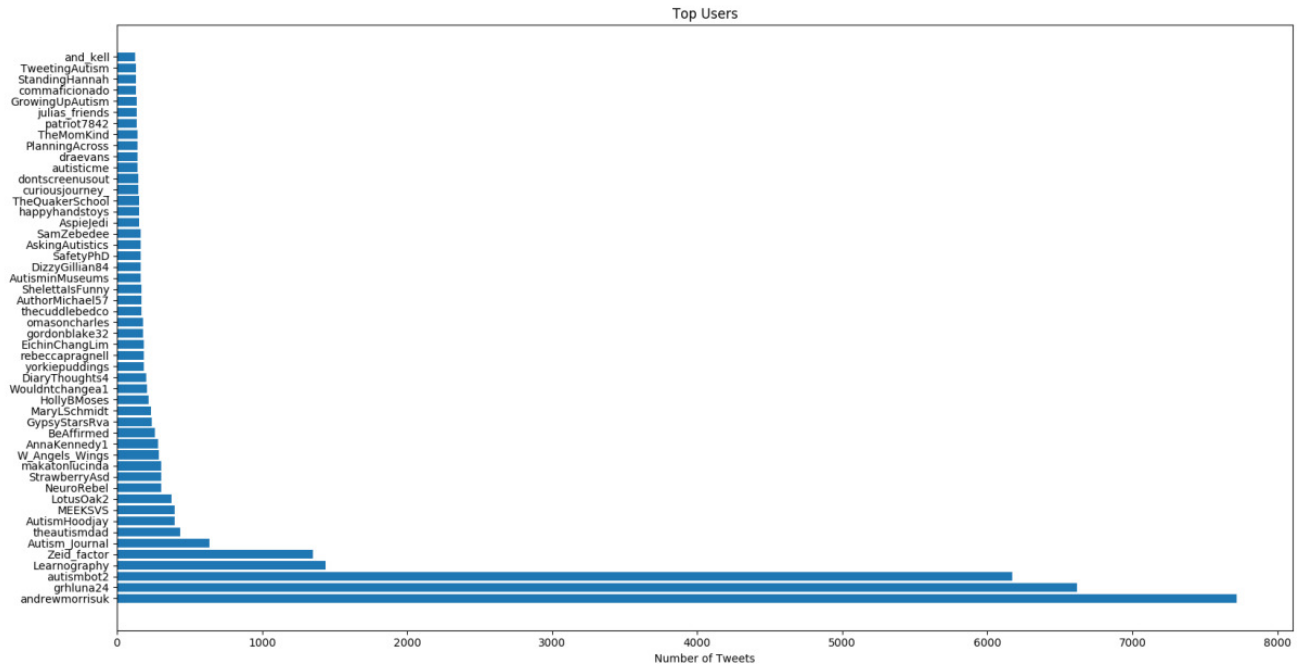

Figure 24: Top Users graph

Next graph provides top 10 sources of the dataset as shown in Figure 25.

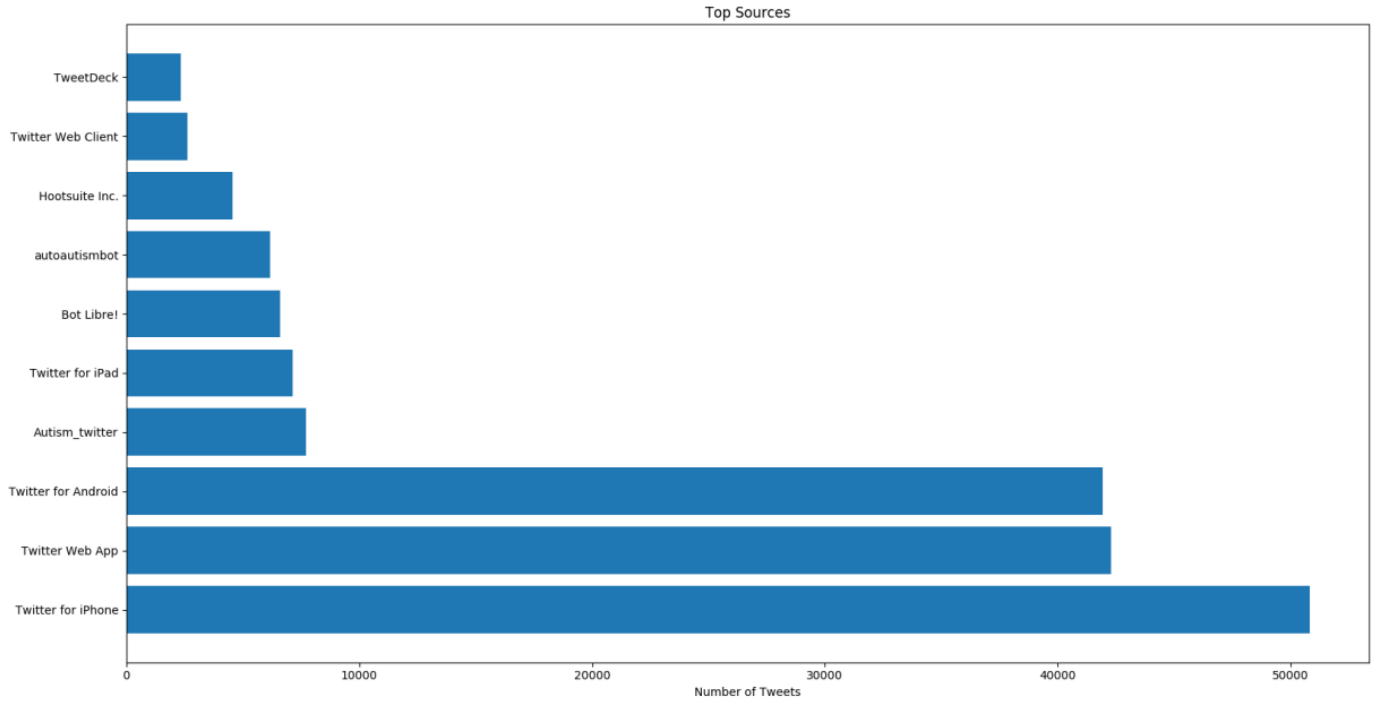

Figure 25: Top Sources graph

Next graph shows the tweet counts among each date based on the collected set as shown in Figure 26.

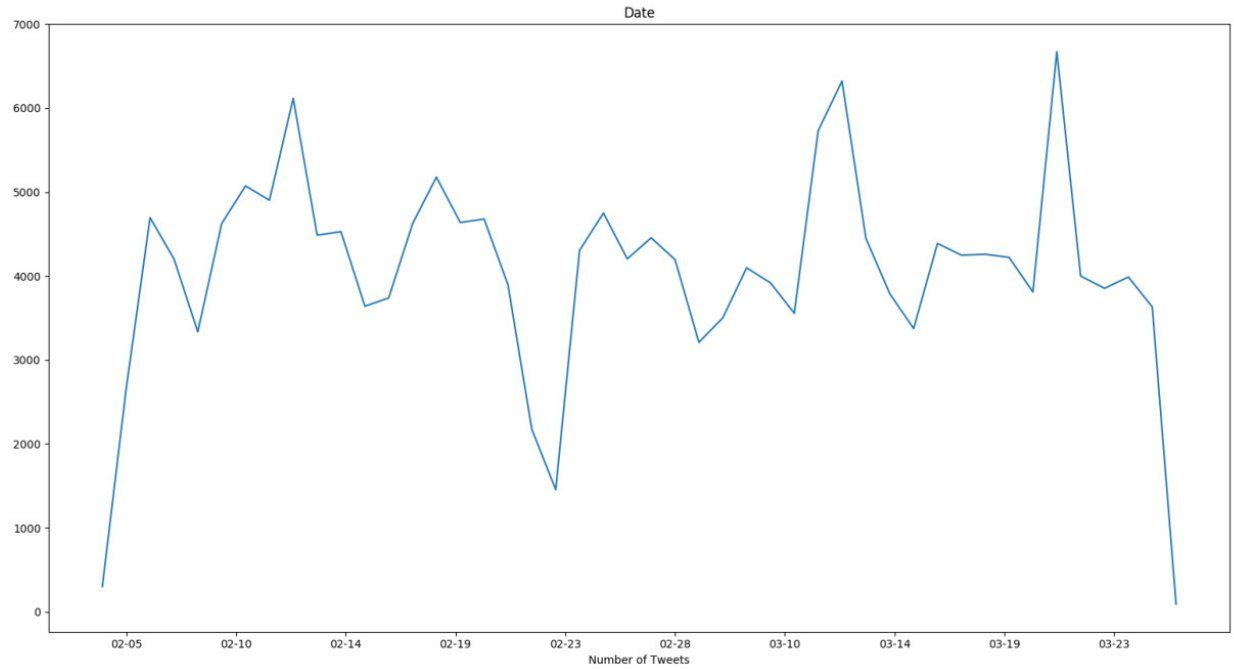

Figure 26: Date-wise Tweet Counts

Next set of graphs are WordClouds for Top Hashtags, Top User Mentions, and Top Phrases in Tweets as shown in Figure 27 , Figure 28 and Figure 29 respectively.

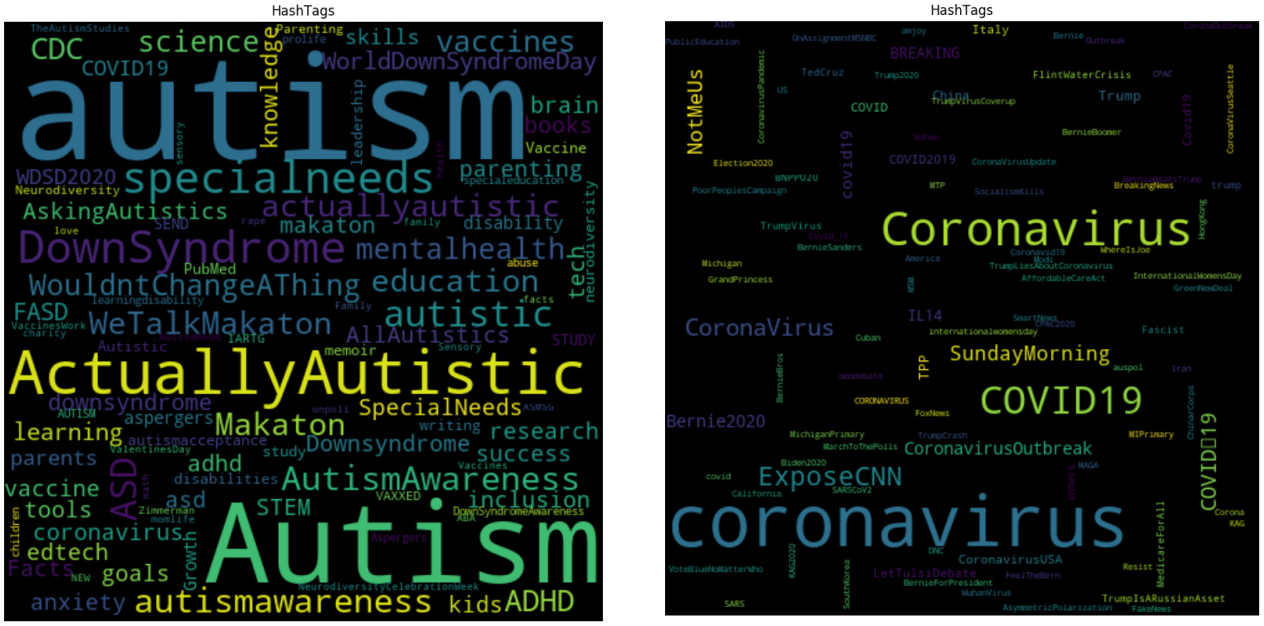

Figure 27: Hash Tag Word Clouds for Down Syndrome and Coronavirus

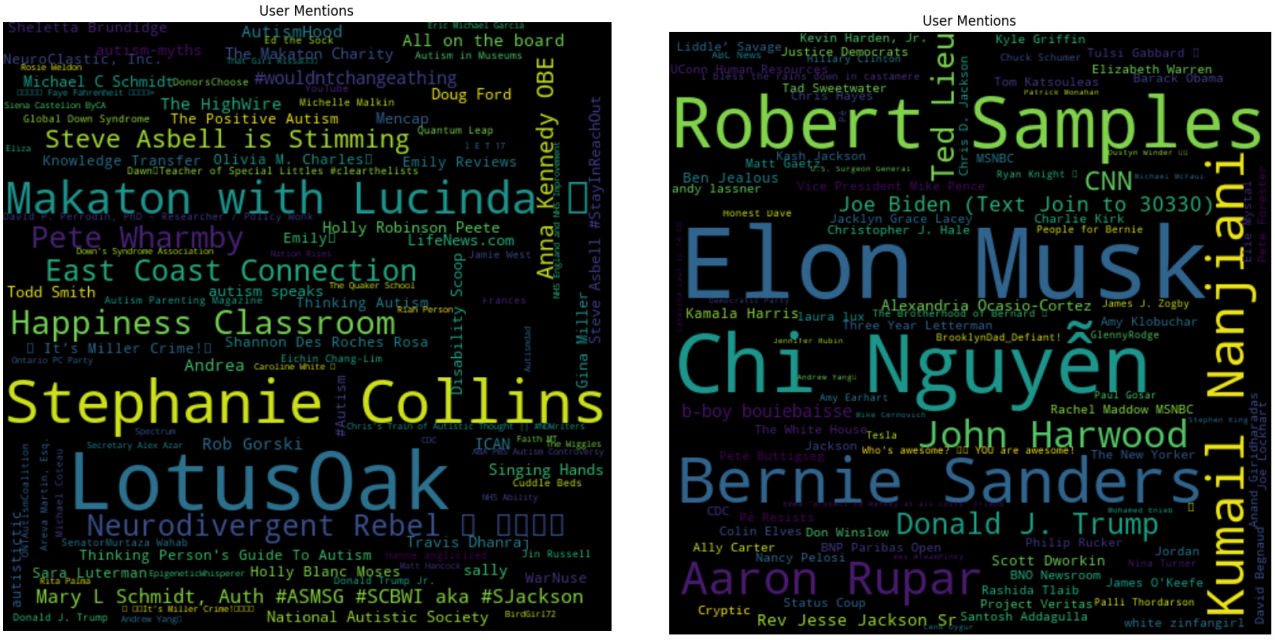

Figure 28: User Mention Word Clouds for Down Syndrome and Coronavirus

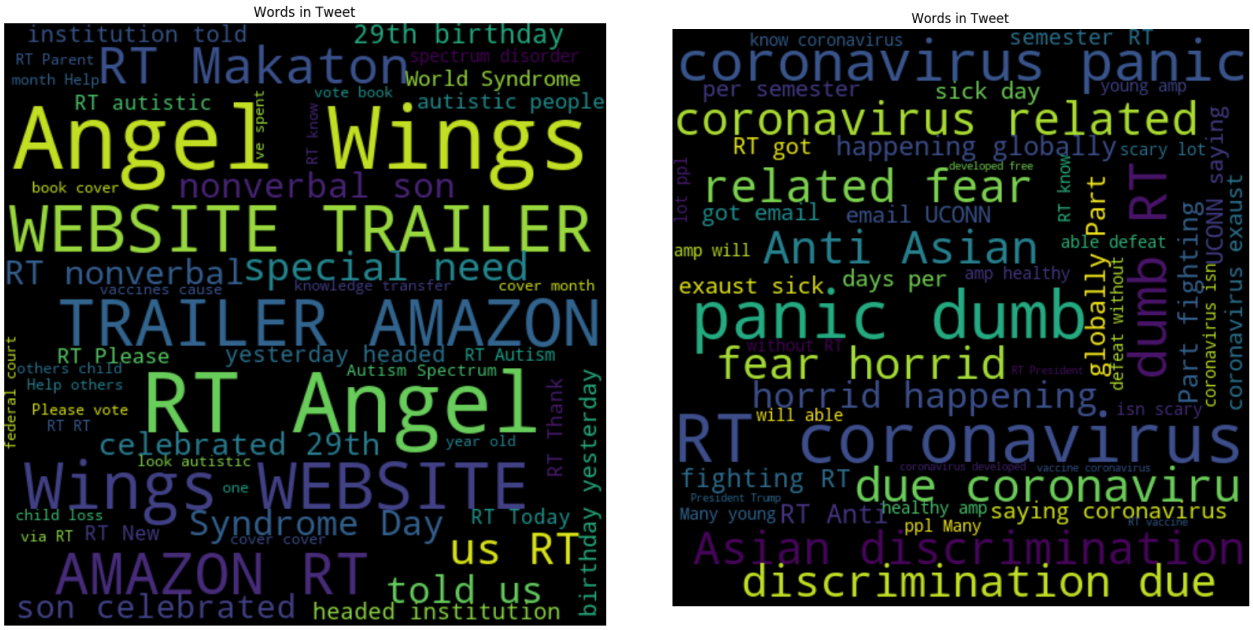

Figure 29: "Words in Tweet" Word Cloud for Down Syndrome and Coronavirus

5. Future Enhancements

Following are the enhancements that can be made to this application if further development effort is possible.

- Enhancement to the prepare logic to handle retweets.
- Sentiment Analysis Module to further enhance Analyze module.
- Scheduler Module set the application as a service which gathers data in the background.
- Integration with cloud blob storage such as Amazon S3.
- Integration of NLP modules for algorithm to improve the context analysis.
- Database storage support which can further enhance the reporting capabilities.

6. Repository

The code of this project can be accessed from below GitHub link.

https://github.com/lithathampan/twitteranalysis

References

[1] Twitter, [Online]. Available: https://developer.twitter.com/en/docs/tweets/timelines/guides/working-with-timelines. [Accessed 4 April 2020].