

Twitter Parser User Guide

Welcome to Twitter Parser, an open source version of a tool I developed and used extensively for my research during grad school. The purpose of this application is to allow someone with little or no programming background to convert and filter data from the Twitter API into something that can be further worked on in Excel, R or your favorite SQL database. This user guide will walk you through collecting data from Twitter, filtering and generating statistics from that data, using this tool to run batch processes on Twitter data, and developing your own add ons for Twitter Parser.

Collecting Twitter Data

Twitter Parser is built to be fully compatible with Tweet Collector (<https://github.com/ToddBodnar/Tweet-Collector>), an open source tool that mimics the basic functionality of the collector used for our research. Download the most recent release of the tool and unzip it at a convenient location. You will then need to obtain access keys to the Twitter API through Twitter's developer page. Modify settings.properties by adding the keys and tokens to the relevant fields.

Tweet Collector can collect data from the Twitter stream using three selection methods: geographical querying, keyword querying and user querying. “geo” querying takes two points of longitude/latitude to define a bounding box of an area that you want to get tweets from. “keyword” collects Tweets if they contain at least one user defined keyword of interest. “follow” collects any tweets a set of users Tweet. Run the Tweet Collector with the “help” argument to get more details.

When you have configured and tested your tweet collector, run the *install.sh* script. This will install cron jobs to make sure the data collector is still running and to split/compress/move the collected data into hourly segments. If you are running Windows, it is suggested that you first install Cygwin. Your machine will now continue to collect Tweets when the machine is on until you removed the two cron jobs and kill the collector.

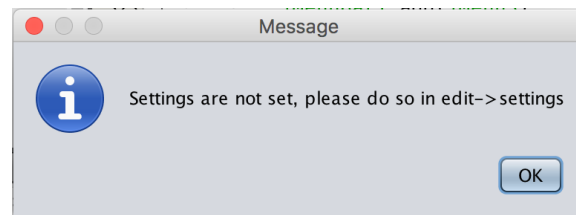
Just be aware that based on your collection time and search query, you may have to deal with a large amount of data to be stored.

Setting up a basic task

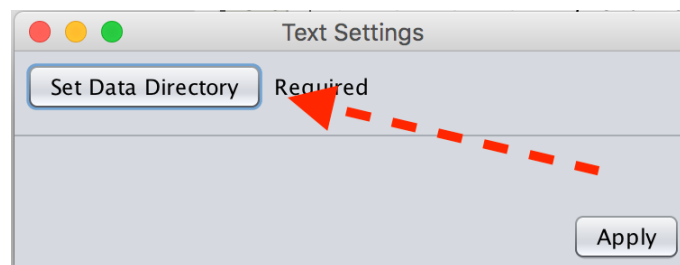
Launch Twitter Parser by double clicking on the jar (not recommended), available for download under github's release tab or by running it through the terminal (recommended) by executing:

```
java -Xmx800m -jar TwitterParser.jar
```

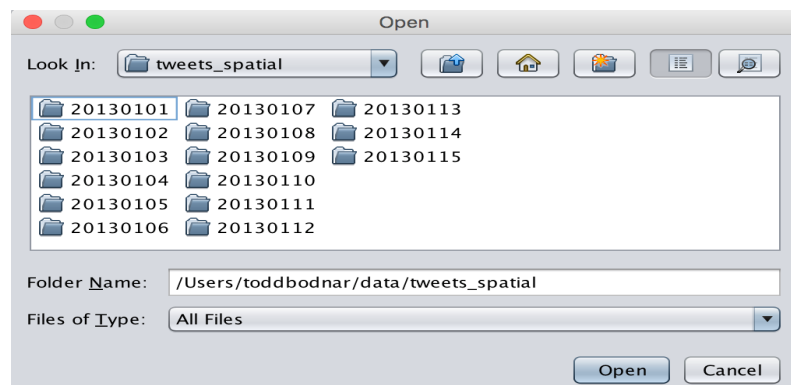
The first time you execute the program, you will have to set the location of the dataset you want to use.



Click the “Set Data Directory” Button.

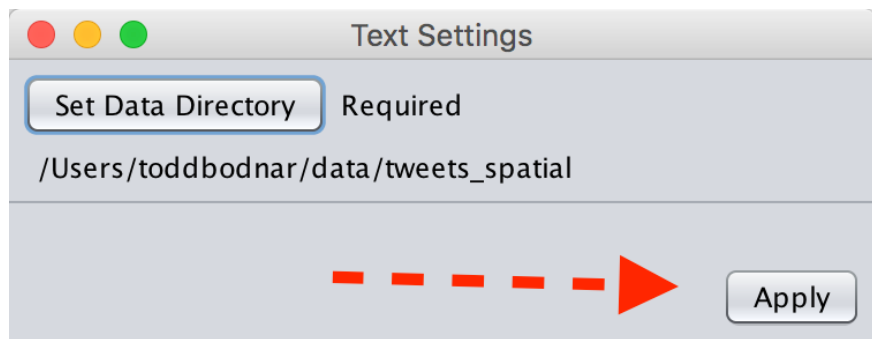


If you used Tweet-Collector, the data should be stored in something like this



You may need to click on a specific date to select all files in the folder.

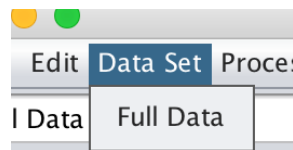
Click the “Apply” button once the data directory is set.



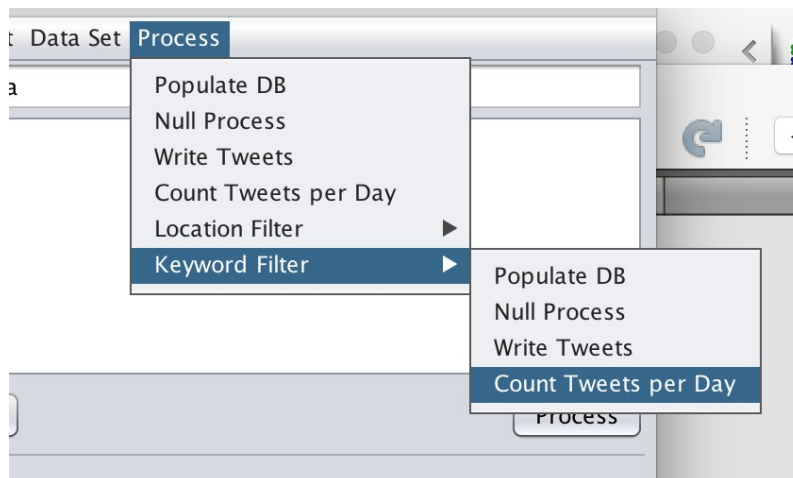
You can change the dataset at any time using the Edit->Settings button.

Now we're going to make a simple job that counts the number of tweets containing the word “snow” in each day.

First select the “Full Data” option under the Data Set menu.



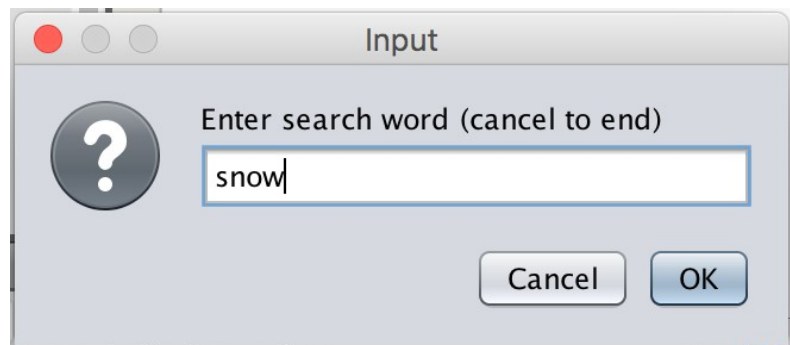
Next, bring up the process menu. Since we want to first filter the data by keyword, select the “Keyword Filter.” Then select “Count Tweets per Day.” For more information on each option, hover over the process or filter.



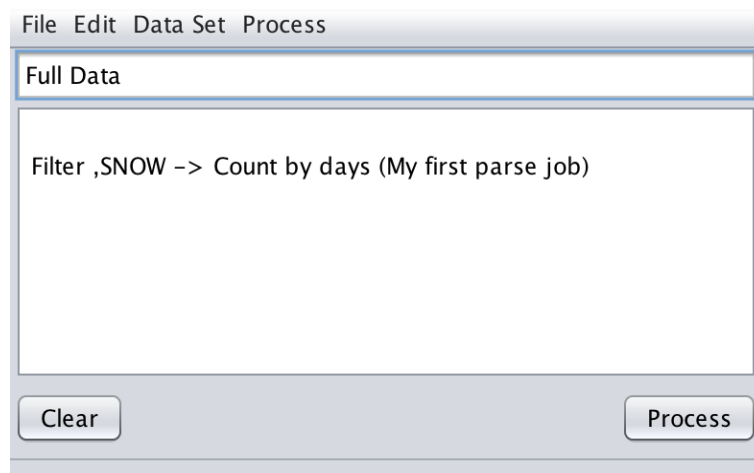
Next, choose a name for the task. Since this is your first job, we'll call this “My first parse job”



Next enter a (not case sensitive) word to search for and hit ok.



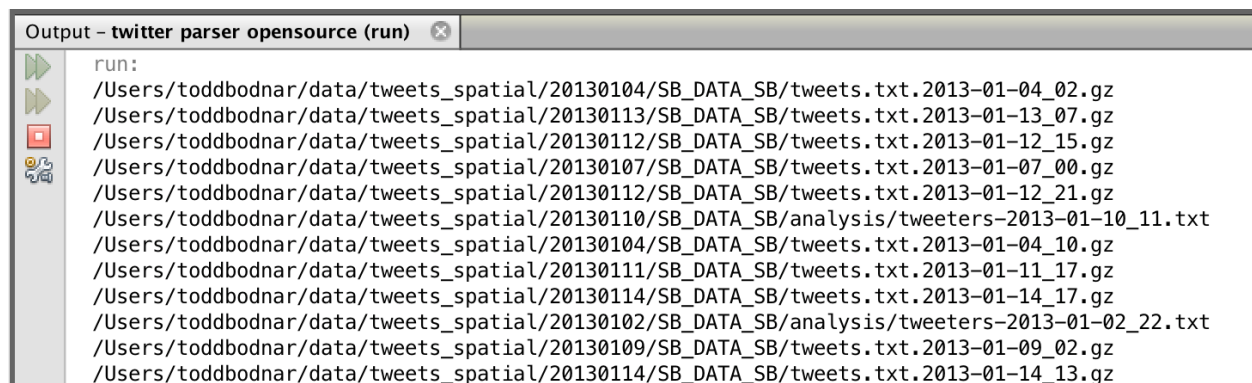
Hit cancel afterwards since we only care about this word. The task is now listed in the main menu.



If you want to do more tasks, you can add them now. It will be faster to run multiple tasks in one process than independent of each other. Click “Process” to begin the job. If everything was set up correctly, you should see something like this:

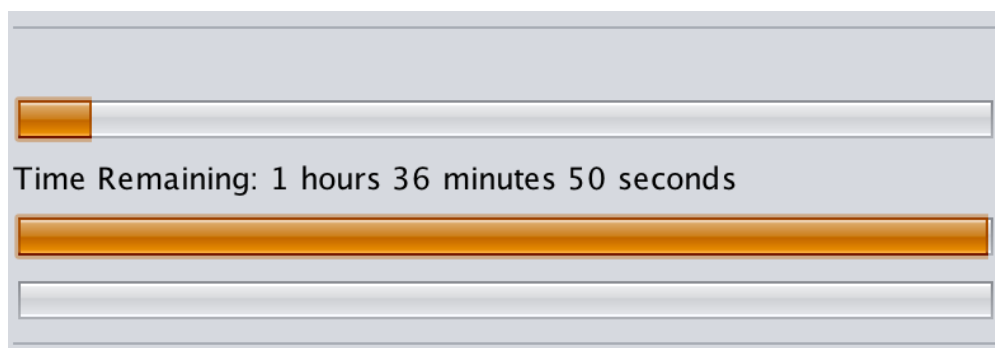


If you launched the program from the terminal, the terminal will list all files to be processed. If some files in the directory are not tweet files, they will be skipped.

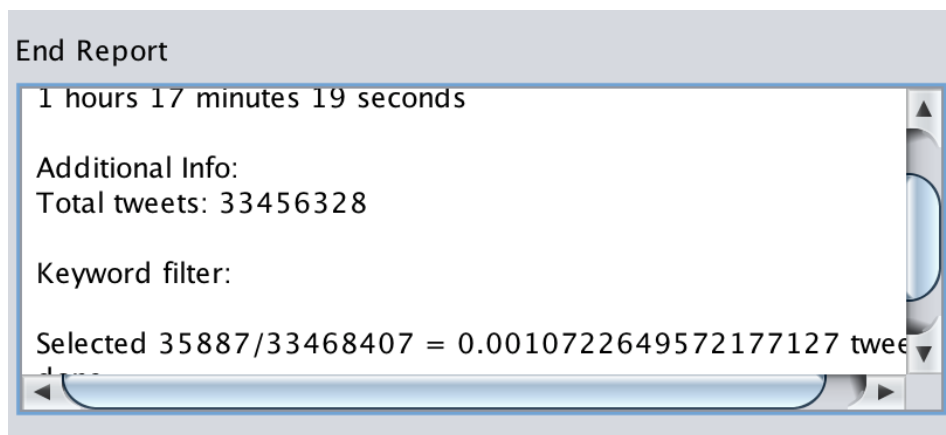
A terminal window titled "Output - twitter parser opensource (run)" with a close button. It displays a list of file paths for processing, each on a new line. The paths are located in the directory /Users/toddbodnar/data/tweets_spatial/ and include various subdirectories and file names with timestamps.

```
run:
/Users/toddbodnar/data/tweets_spatial/20130104/SB_DATA_SB/tweets.txt.2013-01-04_02.gz
/Users/toddbodnar/data/tweets_spatial/20130113/SB_DATA_SB/tweets.txt.2013-01-13_07.gz
/Users/toddbodnar/data/tweets_spatial/20130112/SB_DATA_SB/tweets.txt.2013-01-12_15.gz
/Users/toddbodnar/data/tweets_spatial/20130107/SB_DATA_SB/tweets.txt.2013-01-07_00.gz
/Users/toddbodnar/data/tweets_spatial/20130112/SB_DATA_SB/tweets.txt.2013-01-12_21.gz
/Users/toddbodnar/data/tweets_spatial/20130110/SB_DATA_SB/analysis/tweeters-2013-01-10_11.txt
/Users/toddbodnar/data/tweets_spatial/20130104/SB_DATA_SB/tweets.txt.2013-01-04_10.gz
/Users/toddbodnar/data/tweets_spatial/20130111/SB_DATA_SB/tweets.txt.2013-01-11_17.gz
/Users/toddbodnar/data/tweets_spatial/20130114/SB_DATA_SB/tweets.txt.2013-01-14_17.gz
/Users/toddbodnar/data/tweets_spatial/20130102/SB_DATA_SB/analysis/tweeters-2013-01-02_22.txt
/Users/toddbodnar/data/tweets_spatial/20130109/SB_DATA_SB/tweets.txt.2013-01-09_02.gz
/Users/toddbodnar/data/tweets_spatial/20130114/SB_DATA_SB/tweets.txt.2013-01-14_13.gz
```

While the job is running, you can create more jobs and they will queue up. Meanwhile, the top progress bar will show you how far the current process is.



When the task is completed, the End Report will display information about the process.



And daily counts will be outputted to standard out.

```
Year,Month,Day,My first parse job
null,0
2012,12,31,270
2013,1,5,3092
2013,1,6,2657
2013,1,3,3578
2013,1,4,2573
2013,1,1,2900
2013,1,2,2687
2013,1,9,1941
2013,1,11,2781
2013,1,7,1818
2013,1,8,1541
2013,1,10,2783
2013,1,13,2100
2013,1,12,2419
2013,1,14,2747
Total tweets: 33456328
```

Keyword filter:

Selected 35887/33468407 = 0.0010722649572177127 tweets
done

1 hours 17 minutes 19 seconds

Additional Info:

Total tweets: 33456328

Keyword filter:

|

Selected 35887/33468407 = 0.0010722649572177127 tweets
done

Headless Execution

Twitter Parser can execute scripts from the command line without having to spawn a GUI. This is useful if you want to regularly execute some analysis without user interaction.

First, set up a process (see above) but do not process it.



Second, save the script through File->Save. The script is in a human readable form. One task per line. If you want to chain filters, they are separated by semicolons. (The first line is a date setting, which doesn't work but is maintained for compatibility with older scripts. Fill it with a comment if you want.) For example, this code:

```
20110000,20200000
Populate DB;org.git.mm.mysql.Driver;jdbc:mysql://localhost:8889/tweets;
Location Filter;/Users/toddbodnar/a_shape_file.shp;output.csv
Null Process;
```

- populates a local db with tweets
- Makes a csv of tweet data filtered by a_shape_file.shp
- Does nothing with tweets

To execute the script, simply run

```
java -Xmx800m -jar TwitterParser.jar your_script_name
```

Extending Twitter Parser

Adding additional pipes and sinks to Twitter Parser is easy. There are two types of plugins you can build: *twitterProcess*-es and *tweetFilter*-s. A *twitterProcess* is defined by an interface at *TwitterParser.TweetParser.processors.twitterProcess*. The process is created through either a call to *clone()* by the GUI or *load(String)* by the headless processor. These functions should get any configuration for the process through either GUI prompts or from the passed String, respectively. When the tweets have been parsed, they will be fed into the *consume(tweet)* function in a single thread. Once all tweets have been parsed and fed into all of the processes, *end()* will be called. It is expected that *end()* will return a summary string of the completed process. *tweetFilter* works in a similar way, except that it is given another *twitterProcess* or *tweetFilter* that it should feed tweets into after any processing/filtering has been done.

Once you have developed your process or filter, you should add it to the *processlist* or *filters* arrays, respectively, in *TwitterParser.helpers.JobFileIO*. This will then add it to the menus in the GUI and add it to the script loader. Feel free to submit a merge request on GitHub to add your code and improve Twitter Parser.

Execution Path

Main Thread

1. Begin with a job built either by the GUI or from reading a script.
2. Recursively build a list of files in the data directory.
3. Shuffle the list of files
4. Pass the list of the files to the tweet_consumer thread.
5. Wait for tweet_consumer thread to finish

tweet_consumer Thread

1. Start the tweet_parse_driver Thread
2. Start the tweet_process_driver Thread
3. For each (gzip) file from the main thread:
 1. Make a decompression stream from the file
 2. For each line in the file
 1. Add the line to the line_buffer
4. Signal tweet_parse_driver Thread that input is complete
5. Wait for the tweet_process_driver to complete
6. Output the result from the tweet_process_driver to std out and the GUI, if available.

tweet_parse_driver Thread:

1. For each line in the line_buffer
 1. Parse the json from the line into a Tweet object
 2. Add the Tweet object to the tweet_buffer
2. If the tweet_consumer hasn't signaled completion, wait a bit and then goto 1.
3. Signal to tweet_process_driver that input is complete

tweet_process_driver Thread:

1. For each tweet in the tweet_buffer
 1. Pass a copy of the tweet to each twitterProcess in the job
2. Goto 1 if the tweet_parse_driver hasn't signaled completion, otherwise goto 3
3. Call end() on each of the twitterProcesses.
4. Combine the outputs of each of the end() calls and pass it back to the tweet_consumer thread

GUI Description

The GUI window has a title bar with three colored buttons (red, yellow, green) and a menu bar with 'File', 'Edit', 'Data Set', and 'Process'.

Select Data Set **Name of Dataset**

List of processes in the current job

Clear **Process**

Submits the above list to the job queue (indicated by a dashed arrow from the 'Process' button to the 'Job Queue' section)

Job Queue

Jobs to be executed

Current job's completion

Waiting **ETA** **Line Buffer**

Tweet Buffer

End Report

Results from twitterProcess.end()