December 3, 2017

Nathaniel Velarde
W205-2, Fall 2017
Exercise 2

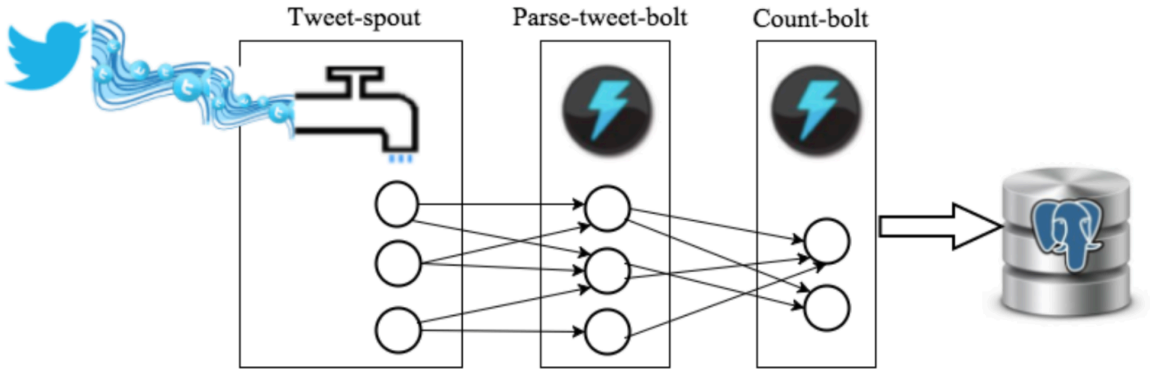
Twitter Application Architecture

Executive Summary

We have implemented an end-to-end streaming application that reads the stream of tweets from the Twitter streaming API using Apache Storm/streamparse, parses the tweets into words, keeps track of the number of the occurrences each word appears in the stream (word counts) and then aggregates the words and word counts into a Postgres database. The application includes two python scripts – finalresults.py and histogram.py – to perform basic analytics on the data collected and stored in Postgres.

Architecture

The figure below shows the overall architecture of our application as well as the Storm topology implemented.



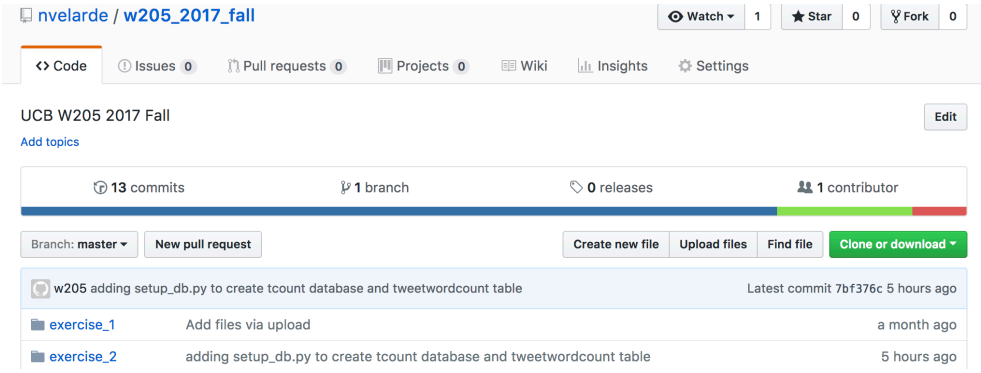
The application uses the Tweepy library (a python module used to interact with the Twitter API) to read the live stream of tweets from Twitter in the "Tweet-spout" component. Then, the "Parse-tweet-bolt" parses the tweets (cleaning them up by filtering out hash tags, user mentions, urls as well as leading and leading and lagging punctuations), extracts individual words from each parsed tweet and emits the words to the "Count-bolt" which counts the number of each word and updates the counts associated with each word in the "tweetwordcount" table in the "tcount" Postgres database. The "tcount" database and "tweetwordcount" table are created in advance of executing the streamparse code.

We implemented the topology as shown in the above figure where each circle represents an instance. Our topology has 3 instances of “Tweet-spout”, 3 instances of “Parse-tweet-bolt” and 2 instances of “Count-bolt.”

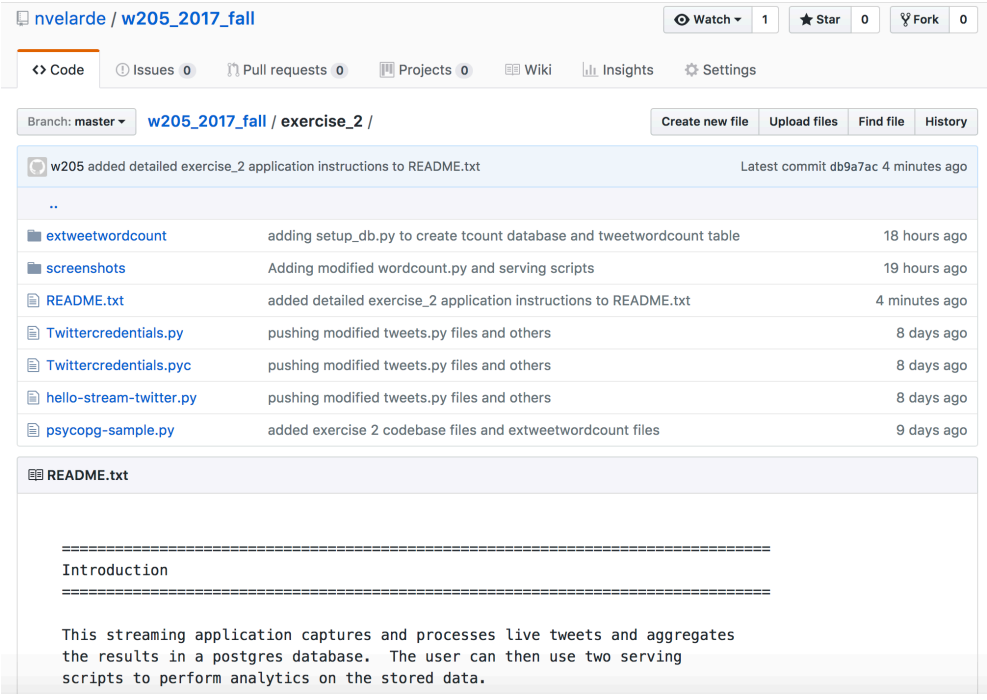
Directory and File Structure

The application files can be accessed through my **w205_fall_2017** GitHub repository under the **exercise_2/** directory.

https://github.com/nvelarde/w205_2017_fall

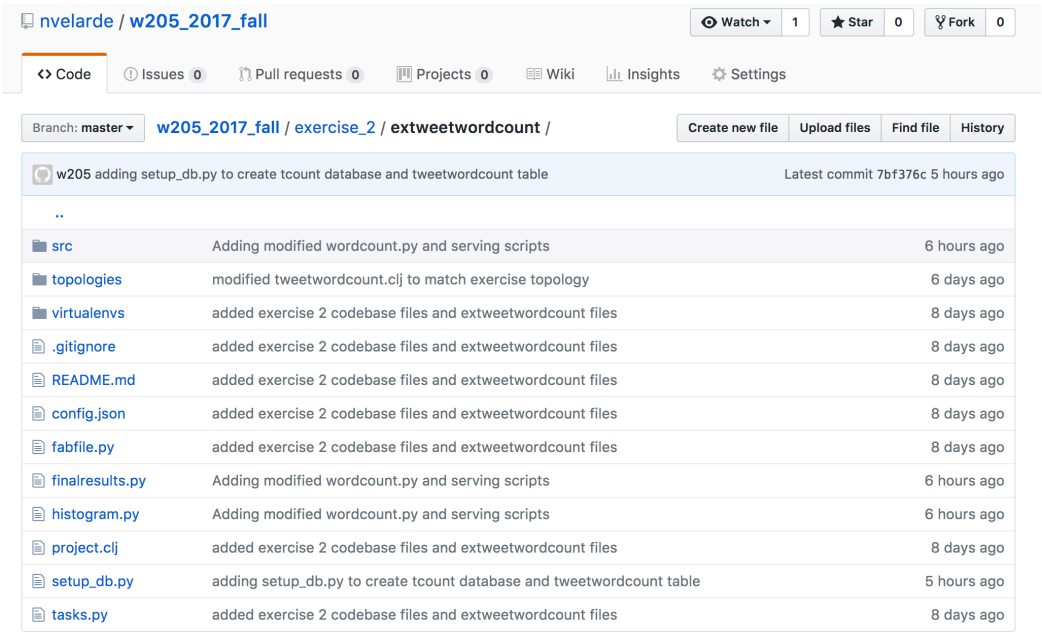


Within the **exercise_2** directory, all of the files necessary to run the application are in the **extweetwordcount** directory. The **exercise_2** directory **README.txt** file has detailed instructions on how to run the application.



Key **extweetwordcount** subdirectories and associated files:

- **src** (directory) – python source code for topology spouts and bolts
    - spouts (directory)
        - tweets.py
    - bolts (directory)
        - parse.py
        - wordcount.py
- **topologies** (directory)
        - tweetwordcount.clj



The **extweetwordcount** directory also contains three python serving scripts:

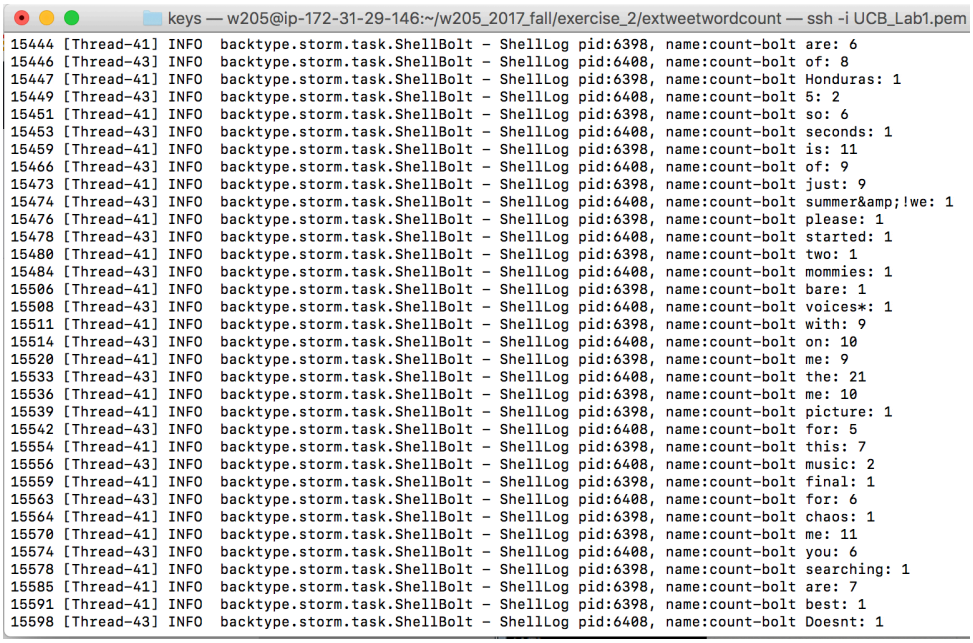
- **setup_db.py** – creates **tcount** database and **tweetwordcount** table – this script has to be run first as the rest of the application code is dependent on the existence of the **tcount** database and **tweetwordcount** table
- **finalresults.py** – when passed a single word as an argument, finalresults.py returns the total number of occurrences of the word in the captured Twitter stream. The user can run the program without passing an argument which will return all of the words in the stream and their total count of occurrences, sorted alphabetically.
- **histogram.py** – takes two arguments, k1 and k2, which must be positive integers separated by spaces and returns all the words with a total number of occurrences greater than or equal to k1 and less than or equal to k2.

Application Execution Instructions

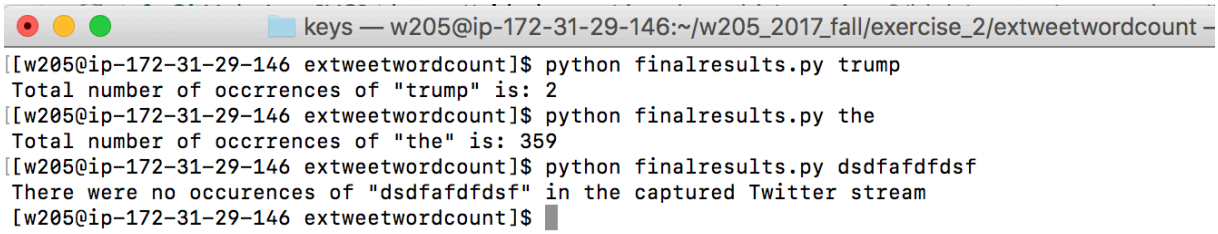
As mentioned on page 2 (see lower screenshot), detailed instructions and information required to setup and run the application can be found in the **README.txt** in the **exercise_2** directory.

Screenshots of Running Application

Continuous log of incoming parsed tweets – words and word counts



Selected output from finalresults.py (case where argument is passed)



Selected output from histogram.py