Directory and File Structure

The file system contains 3 main directories:

- EXTweetwordcount: This is the Apache Storm project which runs the realtime tweet word counts and puts each word into the Postgres database.
- Serving_scripts: This directory contains the finalresults.py and histogram.py files as requested in the assignment. It also contains the plot.py file which creates the requested bar chart.
- Screenshots: This directory contains end to end screenshots of the functioning of the application.

Application Architecture

The architecture of the application works in the following way (these are high-level bullets – detailed instructions can be found in the README file):

- The Initial_create.sql script creates a database called Tcount, and a table within this database called tweetwordcount, which will be written to by the Apache Storm application. The table schema contains two columns:
    - Word (string): Word parsed from tweet
    - Count (int): Number of occurences of the word while the storm application was run
- When the EXTweetwordcount Apache Storm Project is run:
    - It streams tweets in real time – filtering out @mentions, hashtags, retweets.
    - The application then writes to the Postgres database tcount, table tweetwordcount – and writes the count for every word, incrementing the count, or adding a new word as necessary. Note that if the application is stopped, and then restarted, the count picks up where it left off, and does not need to start from zero again.
- The Serving script contains three scripts:
    - Finalresults.py: If run with an argument, returns the argument, and the number of occurences of the argument. If run without an argument, returns all words sorted by count in ascending order, each count separated alphabetically.
    - Histogram.py: Takes two numeric arguments, and returns all words with count between the first argument (lower limit) and second argument (upper limit), sorted by descending order of count. If the script is not run with the correct arguments, it prompts the user to enter correct arguments.
    - Plot.py: Although a bar chart of 20 most occurring words is already provided, the script which creates this on the user's instance is also provided. Running this file creates a file called Plot.png in the same directory.

Overall, the application architecture resembles a lambda architecture, in which the storm application functions as a real-time layer, while the serving scripts have the ability to function as batch scripts. In order to complete the lambda architecture, we would have to keep the storm application running in the background, and then we could make periodic requests to the serving scripts, and watch the application's results evolve in real time.