MIDS - W205
Exercise 2
Sally Hong

Architecture

Directory and File Structure:

- Tweetwordcount
    - logs
    - src
        - spouts
            - `tweets.py` : modified from original with twitter credentials
        - bolts
            - `parse.py` : default from provided file
            - `wordcount.py` : created code to locally count words passed by parse.py and to store in postgres database
    - topologies
        - `tweetwordcount.clj` : default from provided file
    - virtualenvs
        - `wordcount.txt` : default from sparse quickstart command
    - `config.json` : default from sparse quickstart command
    - `drop_create_table.py` : drops and creates table tweetwordcount
    - `fabfile.py` : default from sparse quickstart command
    - `finalresults.py` : serving script to print words and their counts for all or specific words
    - `histogram.py` : serving script to print words and their counts for MIN,MAX specified
    - `project.clj` : default from sparse quickstart command
    - `top20hist.py` : generates PNG file in directory of top twenty words and their counts

Application Idea:

- This program will use Storm via streamparse to
    - live stream tweets in English
    - parse the tweets to remove any #, @, RT, etc.
    - pass to a counter
    - print the word and local count
    - update a postgres database for future queries.
- Then, serving scripts will be used to
    - print the words and their counts captured during the duration of when the streamparse program was run along with an option to search the count for any specific word queried
    - list the words and their counts, listed alphabetically in ascending order, whose counts are between the min and max values specified
    - create a bar plot with the top twenty (20) words and their counts.

Description of the Architecture:

This program will use streamparse which uses the Storm architecture. There is one spout and two bolts. For the spout (tweets), the Twitter API via tweepy is used to live stream tweets in the English language. The tweets are then passed to the bolt (parse). This bolt extracts the tweet, splits the tweet, cleans out #, @, rt, urls, leading and lagging punctuations, checks if the word is in ASCII, then passes it to the next bolt (wordcount). This bolt takes the word passed from the parse bolt and (1) checks if the word is in the local counter (2) sets up a local counter or increments the counter if the word already exists (3) adds the word to the postgres database tcount in table tweetwordcount or increments the count of the word if it already exists.

File Dependencies

`ez_setup` (to install pip)
`pip` (to install remaining dependencies)
`lein`
`tweepy`
`streampaarse`
`virtualenv`
`matplotlib`
`python 2.7`

Necessary Information to Run the Application:

Details of how to run are set out in the readme.txt. Overall, Python 2.7 and postgres will be required along with the dependencies set up above. In addition, a database in postgres will be created before running streamparse.