TweetCount

Live Tweet Count Application

# Application Idea

TweetCount is a streaming application that:

1. Reads the stream of English tweets from the Twitter streaming API
2. Parses them
3. Counts the number of each word in the stream of tweets
4. Writes the final results back to a Postgres database

You can use TweetCount to find the frequency of any given English word within the time frame that you choose to run the application. TweetCount also allows you to view all words that occur within a given range of frequencies.

# Directory and File Structure

*w205-haroon*

* finalresults.py
* histogram.py
* table-create.py
* db-create.py
* *tweetwordcount*
  + config.json
  + fabfile.py
  + project.clj
  + tasks.py
  + *src*
    - *bolts*
      * parse.py
      * wordcount.py
      * \_\_init\_\_.py
    - *spouts*
      * tweets.py
      * \_\_init\_\_.py
  + *topologies*
    - tweetwordcount.clj
  + *virtualenvs*
    - tweetwordcount.txt

# Description of Architecture

Using the Tweepy library, TweetCount pulls a live stream of tweets that pass through an English filter through the *Tweet Spout*.

The tweets then enter the *Parse Bolt*, which then parses each tweet by individual words, filtering out strings that may correspond to usernames, Retweets, hashtags, websites, etc. The *Parse Bolt* then transforms each word into lower-case, ensuring that each word is counted only once, regardless of capitalization inconsistencies.

The individual words then enter the *WordCounter Bolt*, where the count of each word is recorded in a Postgres table named **tweetwordcount** inside of the **tcount** database.

# File Dependencies

Prior to running TweetCount, you must run the following files in their exact order:

1. **db-create.py** file to create a Postgres database.
2. **table-create.py** file to create the table in which our tweet data will be stored

Once we are done executing these files, we are then able to run the TweetCount application to collect our tweet word count data.

Upon completion of our TweetCount application, we can then query the database to get specific results using the **finalresults.py** and **histogram.py** files.

There are several other file dependences within the Apache Storm application contained within the **tweetwordcount** directory. Within the **topologies** folder is a file that outlines the topology of our application. As outlined in the topology file, the dependences are follows:

The final step of the application, which occurs in the **src/bolts/wordcount.py** file, consists of recording the count of a given word in a Postgres table. This step depends on the words sent from the **src/bolts/parse.py** file, which parses each English tweet into separate words, eliminating certain words that are discerned to be hashtags, URLs, usernames, etc., and then transforms them into lowercase letters so that capitalization inconsistencies are neutralized from the final count. The **parse.py** file depends on the tuple sent from the **src/spouts/tweets.py** file, which uses tweepy to get real-time tweet data, which it then passes to **parse.py** in the form of a tuple.

# Before Running the Application

Prior to running the TweetCount application, you will need to make sure to:

* Run the **UCB MIDS W205 EX2-FULL** instance
* Attach the EBS volume from Lab 2 (which contains the Postgres server)
* Start the Postgres server
* Install the **psycopg2** and **tweepy** packages
* Get the application files by cloning the appropriate Github repo
* Run the **db-create.py** and **table-create.py** script files to create a Postgres DB and table

The complete step-by-step process to running getting the application running is contained within the **readme.txt** document.