Application Description

The application filters the twitter firehose API for some English stop words in order to produce a steady stream of English tweets which are available in real-time using serving scripts.

Folder Structure

./EX2Tweetwordcount/src -> Spark bolts and spout
./EX2Tweetwordcount/topologies -> Contains streamparse application topology
./serving -> contains serving scripts for displaying wordcount data
./ddls -> contains bash and SQL for creation of postgres database and word count table
./screenshots -> contains screenshots of running application

Topology & Architecture

The application filters the twitter firehose API for some English stop words in order to produce a steady stream of English tweets that are then parsed by Parse-tweet-bolt for words that should not be included in analysis of overall word frequency. The topology then sends the parsed word list into the Count-bolt which saves the count for each word into a Postgres table called "Tweetwordcount" within the "tcount" database. Serving scripts are provided which allow a user to display the updated word counts for any word or all words in real-time and also to select a list of words using filtering of word count totals.

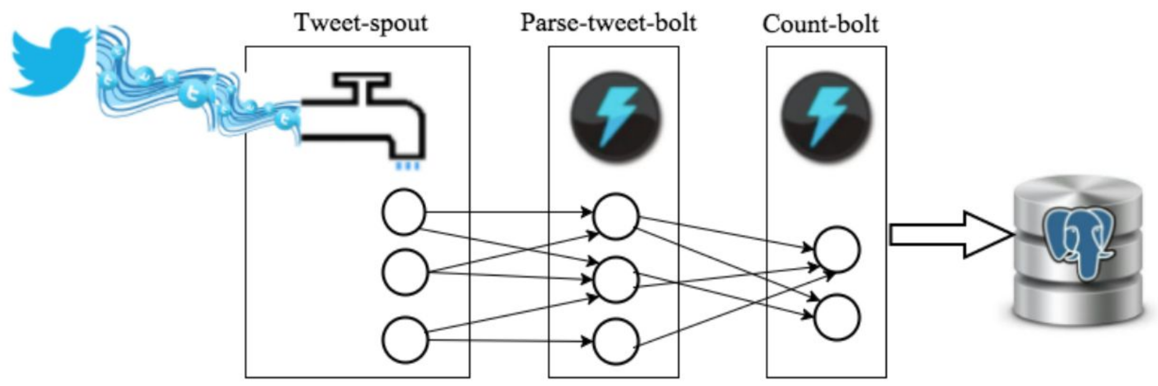

Figure 1: Application Topology

Dependencies

- Streamparse (pip install streamparse)

- Tweepy (pip install tweepy)
- psycopg2 (pip install psycopg2)

Steps to Deploy Application

1. Install pip and then install each of the dependencies listed above
2. Install postgres and ensure server is running
3. Execute ./ddls/run_ddls.sh to create Postgres database and word count table
4. Add the following lines to your ~/.bash_profile:
   export TWITTER_KEY=[key]
   export TWITTER_SECRET=[secret]
   export TWITTER_OAUTH_TOKEN=[oauth token]
   export TWITTER_OAUTH_SECRET=[oauth secret]
5. Within directory EX2Tweetwordcount execute command “sparse run” (to run in background, execute “nohup sparse run”)
6. Execute ./ddls/finalresults.py to see all word count results in real-time
7. Execute ./ddls/finalresults.py [word] to see word count results for [word] in real-time
8. Execute ./ddls/histogram.py [min] [max] to display a sorted list of all words with count at least [min] and less than [max]