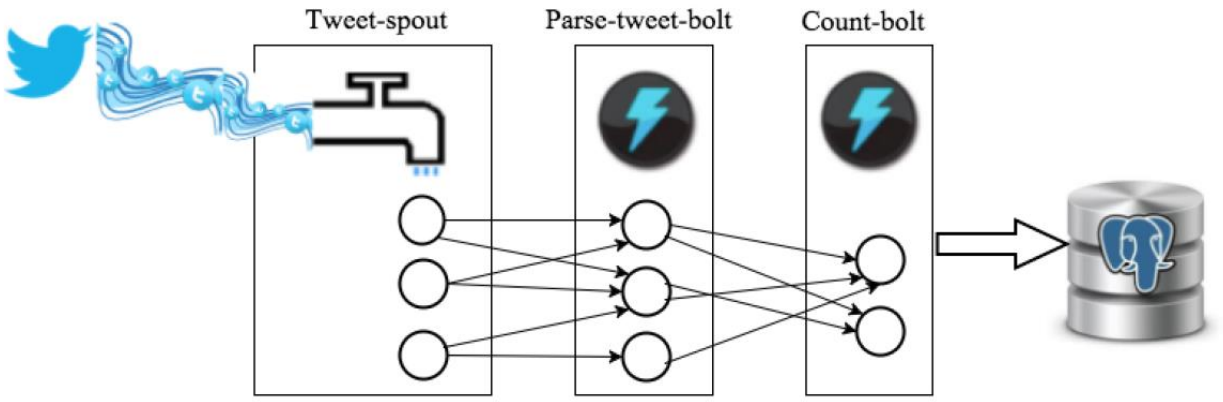

Description:

The application is built as a storm project. The project has one spout and 2 bolts. The spout is configured to be a twitter feed. The first bolt parses the tweet, filters unwanted words, converts the words to lower case and sends it over to a count bolt that counts the number of words and updates the count of words stored in a table within post-gres.

Directory Structure:

Project: The project is named EX2Tweetwordcount and is available as a folder with the same name once the repository from git hub is cloned. /W205_EX2/EX2Tweetwordcount

Topology: The topology file is named `tweetwordcount.clj` and is available at the location /W205_EX2/EX2Tweetwordcount/topologies, where W205_EX2 is the name of the cloned repo.

Spout: The spout is named `tweets.py` and is located in /W205_EX2/EX2Tweetwordcount/src/spouts.

Bolt 1: The parse-tweet bolt is named `parse.py` is located at 205_EX2/EX2Tweetwordcount/src/bolts.

Bolt 2: The count bolt named `wordcount.py` and is located in 05_EX2/EX2Tweetwordcount/src/bolts.

Python scripts:

- `finalresults.py:` : This can be found at /W205_EX2/
- `histogram.py:` This can be found at /W205_EX2/
- `psycopg-v1.py` : This can be found at /W205_EX2/ and the script that needs to be run ahead of running the sparse run on the project. This script creates the Postgres DB called tcount and a table called Tweetwordcount table inside tcount database.

Application Execution:

After cloning the git repository, the user will need to first run `psycopg-v1.py`. This will create the needed db and table structure in Post-gres. Once this is done, the project can be run using the sparse run command.

The spout streams twitter data and passes that along to the first bolt(parse.py) that parses the words, cleanses it, converts it to lower case and passes it along to the second bolt. The word count bolt receives these words from the parse bolt and checks whether the word already exists in the Tweetwordcount PostGres table. If the word does not already exist, it writes it to the db and if it exists, it updates the count on the db with the new count.

The words were converted to lower case for better coherence. So that "To" and "to" for instance wont be counted as different words.

After running the steam for a sufficient period of time, press cntrrl+C to exit. To query the tables use the below scripts:

1) finalresults.py:
   a) If this script is run without any arguments, it will return the list of all words sorted alphabetically and the corresponding word count.
   b) If the script is run with a word passed to it, it will return an output showing the count number for that word.
2) Histogram.py
   Run this script by passing two numerical arguments(k j) separated by a space and it will return words that have a count >=j and <= k.

Code Changes:

1) psycopg-v1.py creates the needed dba and table in postgres and need to be run ahead of running the project stream **.**
2) Spout tweets.py was modified to add twitter credentials.
3) Topology was modified to add the specified number of nodes for the spouts and bolts(3,3,2)
4) Bolt 1(parse.py) was modified to convert all words to lower case before sending it to bolt 2.
5) Bolt2(wordcount.py) was modified to write to the Postgres table as opposed to just displaying the output in the screen.
6) Finalresults.py and histogram.py are new scipt files that have the functionality defined in the above section.