Tweet Word Count Documentation

Daniel Volk

W205 Data Storage and Retrieval – Section 6

December 9th, 2017

Directory and File Structure

This project folder contains the files necessary to stream data from twitter, count each word in each tweet as they come into the stream and write the output to a database. It also includes several scripts to output the results of the twitter stream. Everything that runs the streaming application is contained in the extweetwordcount directory. The serving scripts are contained in the results directory. Screenshots of the process running and the resulting output is contained in the screenshot directory.

- Exercise_2
    - extweetwordcount
        - src
            - bolts
                - __init__.py
                - parse.py
                - wordcount.py
            - spouts
                - __init__.py
                - tweets.py
        - topologies
            - tweetwordcount.clj
        - virtualenvs
            - wordcount.txt
        - .gitignore
        - config.json
        - fabfile.py
        - project.clj
        - README.md
        - tasks.py
    - results
        - finalresults.py
        - histogram.py
        - Plot.png
    - screenshots
        - screenshot_postgresOutput.png
        - screenshot_ServingScriptOutputs.png
        - screenshot_twitterStream.png

Application Idea

Tracking and analyzing live data feeds from Twitter can give businesses a better understanding of current events and trending topics. News agencies can monitor Twitter trends both globally and locally to identify breaking news and curate the news of the day to those stories which matter most to their target audience. This application shows off the basic architecture necessary for streaming live data directly from Twitter, processing it, and writing the results directly to a database. Additional processing could allow tweets to be analyzed in real time to identify particular events or trends of interest to the business.

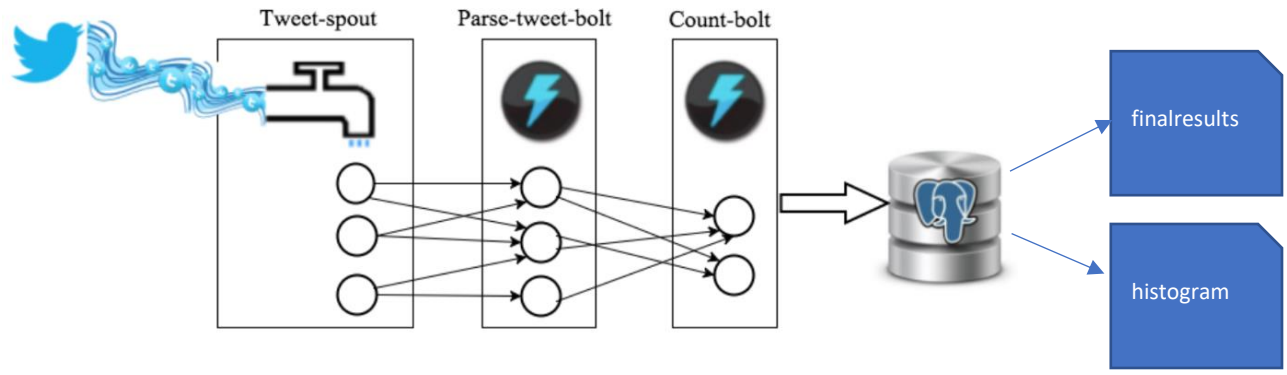
Description of the Architecture

The application architecture has several steps. The first step is a storm layer which is used to stream data from twitter, process it, and write it to a postgres database. The second layer is the postgres storage layer where the results from the streaming are updated as more words stream int. The final layer is two python serving scripts which access the data on postgres and output predefined results.

The storm layer contains 3 main components. There are 3 tweet-spouts which read tweets in from twitter, extract the full text of each tweet from the incoming json, and send each tweet to the next bolt. The next layer contains 3 parse-tweet-bolts which are used to split each tweet into individual words and clean the words before passing them on to the next bolt. The final bolt is a count bolt which will update the word count by one if the word currently exists in the database, otherwise it will create a new row for the word with count 1.

The postgres database contains one table with a primary key of word and a count value associated with that word. It is written to dynamically by the storm architecture.

The two serving layers are available to view the data in the database without having to enter and query postgres. The finalresults script will output the count for a specific word if one is provided. If no word is provided, it will output the full contents of the database. The histogram script takes 2 numeric arguments and will return all the words with wordcounts between the two provided values. If the values are invalid the script will output an error.



File Dependencies

The following programs come with the “UCB MIDS W205 EX2-FULL” community AMI available on AWS.

- Storm
- Postgres
- Python
    - psycopg2
    - tweepy

Running the Application

To run the application, one must already have storm installed as well as psycopg and tweepy as indicated above. Ensure that postgres is running on the machine or there will be errors. In the UC Berkeley community AMI this is done by running the command “/data/start_postgres.sh”. Next, you should be sure that there is a “tcount” database on postgres with a tweetwordcount table that can accept the results of the stream.Finally, you should navigate to the *extweetcount* folder and run the command “sparse run” which will begin the storm job. This job can run as long as you choose and data will be written to the database until the stream is interrupted.

To interact with the parsed data, you can either query it directly in postgres or use the serving scripts to extract specific information. Running the command “python finalresults.py” will return the full contents of the database. The command “python finalresults.py hello” will return the count for how many times the word “hello” appeared in the stream. The command “python histogram.py 10,12” will return all the words in the database with a count between 10 and 12.

Scaling out the Application

The application is built in storm and postgres and will be able to scale out by adding more spouts and bolts in Storm and allocating enough space in postgres. At the moment, the application takes advantage of Twitter’s free stream, but that stream only provides a small portion of the actual twitter traffic. If one were to use the enterprise level stream to get all the data from Twitter, then more bolts would need to be allocated in order to process and write all of the incoming data.