W205 Exercise 1 – Summer 2017
Frederick “Michael” Frazzini

Tweetwordcount

Application Overview and Idea

Tweetwordcount is an application that parses and analyzes streamed “tweets” from Twitter, the popular social network site that allows any user to “tweet” 140 character messages in a global, near real-time, open chat like platform. The application requires a Twitter user account with a Twitter application setup that provides for usage of the Twitter API. Tweetwordcount uses the Twitter API to read tweets as they are published and streams each tweet to an Apache Storm streaming application that parses and analyzes each tweet. Tweetwordcount creates and produces word level statistics for the tweets it receives while it is running and it could be the basis for more advanced streaming applications that use and integrate with Twitter.

Overview Architecture

Tweetwordcount requires a Twitter account and application that provides for API access and usage keys. Tweepy must also be installed which enables Twitter API integration. Apache Storm is utilized with a Tweet spout, and parse and wordcount bolts based in Python, configured in a multi-threaded topology. Overview diagram presented below:

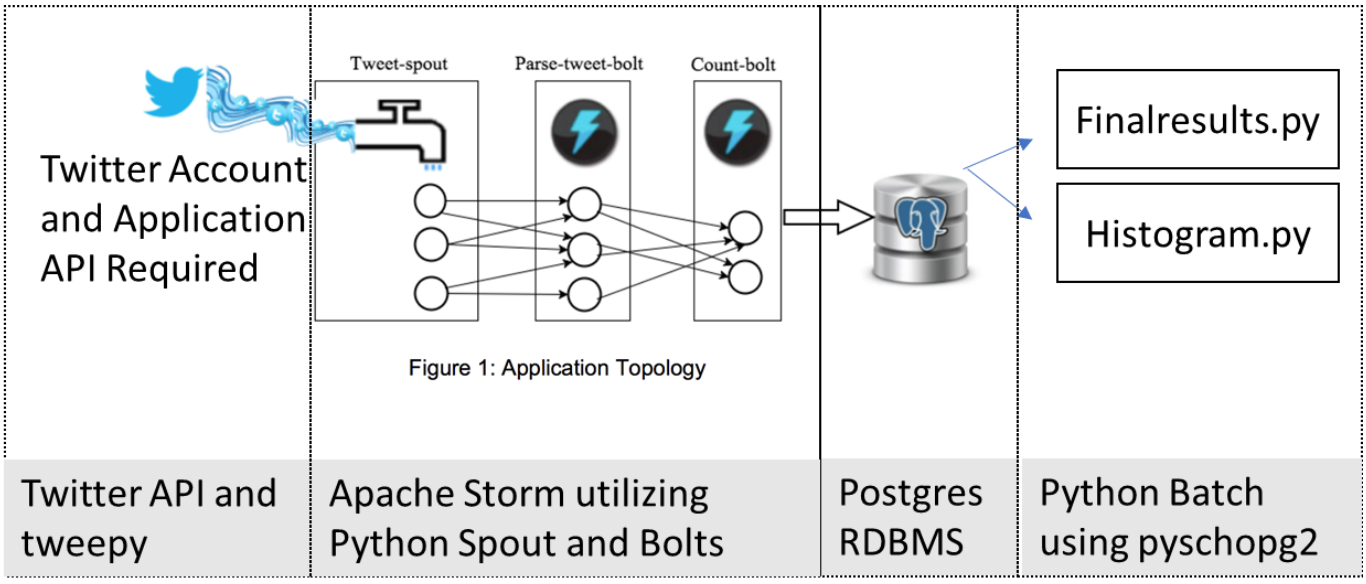

Figure 1: Application Topology

File Structure and Dependencies

The following directories and files are required for Tweetwordcount, described below:

/extweetwordcount (Streamparse generated quickstart project directory)

Note: various directories and files are created as part of the Streamparse project that are not discussed here – only directories and files that are customized for the Tweetwordcount application are presented.

/extweetwordcount/finalresults.py

Program to count Tweetwordcount total word occurrences of a specific word that is passed as a runtime parameter.

/extweetwordcount/histogram.py

Program to count Tweetwordcount total word occurrences where counts are >= and <= an integer range passed as runtime parameters.

/extweetwordcount/topologies/tweetwordcount.clj

Apache Storm topology written in Clojure. This topology defines a Tweet spout, and parse and wordcount bolts based in Python, in a multi-threaded configuration.

/extweetwordcount/src/spouts/tweets.py

This Python based spout utilizes tweepy and requires Twitter API credentials to be defined as specific environment variables:

TWITTER_CONSUMER_KEY = Twitter Consumer Key
TWITTER_CONSUMER_SECRET = Twitter Consumer Secret
TWITTER_ACCESS_TOKEN = Twitter Access Token
TWITTER_ACCESS_SECRET = Twitter Access Token Secret

The tweets.py spout listens for tweets and emits each incoming tweet to the parse.py bolt.

/ extweetwordcount/src/bolts/parse.py

This Python based bolt receives tweets from the tweets.py spout, parses each word, and removes non-ASCII characters and invalid words.

/ extweetwordcount/src/bolts/wordcount.py

This Python based bolt receives parsed words from the parse.py bolt, connects and integrates with a Postgres SQL RDBMS, and inserts and updates a table containing words and word counts.

Additional Dependencies:

The following additional dependencies exist for the Tweetwordcount application:

1.) Twitter Application Setup with Twitter API credentials to be defined as specific environment variables:

TWITTER_CONSUMER_KEY = Twitter Consumer Key
TWITTER_CONSUMER_SECRET = Twitter Consumer Secret

TWITTER_ACCESS_TOKEN = Twitter Access Token
TWITTER_ACCESS_SECRET = Twitter Access Token Secret

2.) Installation of tweepy (see http://www.tweepy.org/)
3.) Postgres SQL RDBMS install, with a database created (named "tcount"), and a table named "tweetwordcount" with the following DDL:

```
CREATE TABLE tweetwordcount
(word VARCHAR(140) PRIMARY KEY     NOT NULL,
 count INT     NOT NULL);
```

Operating Instructions:

1.) Source the following environment variables using the credentials gained with a Twitter account and Twitter application setup (https://apps.twitter.com) and ensure they are setup in the environment used to run Tweetwordcount before the application is run:

TWITTER_CONSUMER_KEY = Twitter Consumer Key
TWITTER_CONSUMER_SECRET = Twitter Consumer Secret
TWITTER_ACCESS_TOKEN = Twitter Access Token
TWITTER_ACCESS_SECRET = Twitter Access Token Secret

2.) Ensure that a Postgre SQL RDBMS installation exists and the server is running. It is expected that the tcount database is setup with the tweetwordcount table exists and is empty. IMPORTANT: You can ensure the table is empty each run with the following command:
truncate tweetwordcount;

3.) Go to root directory of the application:
$ CD /extweetwordcount

4.) Run the application using Streamparse:
$ sparse run

5.) From another terminal window (to keep the Tweetwordcount running in the initial window), run the finalresults.py program to count Tweetwordcount total word occurrences of a specific word that is passed as a runtime parameter:
$ python finalresults.py <word to get total counts for>

6.) From another terminal window, run the histogram.py program to count Tweetwordcount total word occurrences where counts are >= and <= an integer range passed as runtime parameters:
$ python histogram.py <lower range integer,upper range integer example: 3,8>