Exercise 2: W205 Fall 2016 John Blakkan: Twitter/streamparse

Directory structure

```
exercise_2                                  # Main project directory
        Extwotweetwordcount                 # Streamparse project subdirectory
                src                         # Source directory for streamparse bolts and spouts
                        bolts
                                __init__.py     # Streamparse initialization file.
                                parse.py        # Bolt to filter and split tweets into words
                                wordcount.py    # Bolt to increment word counts in a postgres table
                        spouts
                                __init__.py     # Streamparse initialization file.
                                tweets.py       # Spout to get twitter data and convert to stream

                topologies                  # Streamparse standard directory
                        tweetwordcount.clj  # Description of spout/stream/bolt connection (in clojure)
                <others>                    # Standard/generated streamparse directories and files
        screenshots                         # directory for screenshots of program execution
                screenshot-twitterstream-running-stopping-top20.png
                screenshot-extract-results.png
                screenshot-both-screens-local-linux-with-graph.png
        finalresults.py                     # script to give word occurrence count (some or all)
        histogram.py                        # script to list words in an occurrence count range
        create-database.py                  # purges and initializes the database
        top20tocsv.py                       # "helper" script to create .csv used to create Plot.png
        plot.txt                            # Description of how I produced Plot.png
        makeplot.r                          # R script to create Plot.png offline- not run in instance
        Plot.png                            # exercise deliverable- bar graph of 20 most frequent words
        Readme.txt                          # Detailed instructions and comments on running
        Architecture.pdf                    # This file
        Twittercredentials.py               # Copy of credential just for test (they're in tweets.py)

# Note there are many other development files and trial versions of files in the github directory
```

Application: concept and use case

The overall idea is to capture data from twitter and use it to count the frequency of occurrence of words in the tweets. The analysis is made available by command line programs (finalresults.py and histogram.py), as well as a bar graph.

This information could be used for many purposes (e.g. designing specially optimized compression algorithms for tweet data, tracking twitter users interests and trends, etc.) By leaving the word occurrence data in a postgres database, other, more advanced analytic programs could be developed in the future.

Architecture

tweepy is used to get the twitter data. Twitter requires users to authenticate in order to access the stream. Authentication tokens are generated (and registered with Twitter) in Twitter's standard manner (as detailed in the assignment). It uses the Oauth third-party authentication mechanism for this, although in this case the "third-party" is Twitter itself.

streamparse is the core framework, providing a python interface to Apache Storm. (Note that the toplogy file is clojure, not python.) A three stage pipeline is used. The tweet-spout uses the tweepy library (authenticating with Oauth) to access twitter data and convert it int a "tweet stream." There are three instances of Tweet-spout tasks running in parallel. Next, comprising the second stage, are three parallel tasks of parse-tweet-bolt. These perform some data clean up, and split the tweets into individual words (including words containing apostrophes, which adds some complexity to subsequent processing). The final pipleline stage consists of two instances of the count-bolt. These take the streams (actually, each instance gets a subset of the tuples in the streams, with the instances handling a disjoint set of words) and use them to update the word counts stored in a postgres table.

psycopg2 is used to access the postgres table. Note that currently every write is committed. For a full production system, it might be desirable to relax this (or build some caching of writes into the counter bolt). We are using postgres version 8.4, so we don't have direct support of the "upsert" operation (i.e. increment a word count, or set the word count to 1 if this is it's first occurance). We simulate this "upsert" operation with a small python function in the "count-bolt." Moving to the postgres 9-series will eventually eliminate the need for this.

finalresults.py will give the number of occurrences of a specified word (specified as a command line argument), or a table of the occurrence count of all words, sorted alphabetically. Note that by intent, fairly lax filtering is applied in the main application, so there are low occurrences of some unusual "words" (i.e. special symbols). This script is runnable while the program is operating (typically from a separate ssh session)..

historgram.py will give a list of words with occurrence counts in a specified range (range is given as two integers on the command line). This script is runnable while the program is operating. (typically from a separate ssh session).

Plot.png is a plot of word occurrences in a representative run. See the file plot.txt for a description of how this was produced.

top20tocsv.py will give (to stdout) a .csv formatted list of the top 20 most frequent words, and their respective occurrence count. It was originally developed to support the production of the Plot.png file, but is very useful for monitoring the ongoing running of the application. It is runnable while the program is operating (typically from a separate ssh session).

File dependencies

The main dependency is on the Twitter authentication token information. For the streamparse application, it's not a separate file, but is copied into the top of the tweet-spout.

One external dependency is the use of "R," which, for this project, I ran on a separate, standalone computer, rather than load up R on the AMI.

How to run the application

The Readme.txt file in the project main directory provides the details.