Architecture

Application Description

The application streams tweets from the Twitter streaming API, parses tweets into words, counts each of the words, and writes the current count in the stream into a postgres database.

1. **Startup Script** – The ex2_setup.sh file is a shell script that installs the python libraries psychopg2 and tweepy. The script then starts the postgres sever. It also creates a project called wordcount in Streamparse and runs a python script to create a database called postgres wih a table called Tweetwordcount. The final step removes unnecessary files and moves needed files to the correct directories. Once this is complete the script will run the streaming application. The script should be run in root, but it will create directories/databases under the w205 user.
2. **Create database** – The create_postgres_db.py file creates a database named postgres and create a table called Tweetwordcount in the database. The file is run by the ex2_setup script.
3. **Spout** – The tweets.py file uses the Tweepy library to read a live stream of tweets from twitter's streaming API. The API utilizes OAuth which is used by websites to authorize applications to access information. The access is granted by passing unique credentials in the form of consumer_key, consumer_secret, access_token, and access_token_secret.
4. **Bolts** – The parse.py file takes a tweet and parses it into words and emits the words to the next bolt. The wordcount.py bolt counts the number of each of the words parsed by the previous bolt and updates the count in the Tweetwordcount table in the postgres database. For updating the Tweetwordcount table the code checks to see if the word exists in the database, if it does not it will insert a row with the new word and a count of 1. If the word already exists the count is incremented by 1. An alternative would be to maintain the current count of words in the stream and use this to update the count of existing words however by incrementing by 1 we do not have to maintain a counter.
5. **Topography** – The tweetwordcount.clj file defines the topography. Defines a spout called tweet spout that emits data to the parse-tweet-bolt, which emits data to the count-bolt. The parse-tweet-bolt and the count-bolt both have two instances.
6. **Serving Scripts** – The finalresults.py file gives the final count of the words entered by the user. If no words are entered the file returns an ordered list of all words in the Tweetwordcount table. The histogram.py file takes two integers and returns the words that fall within the range of the integers. The script has error handling for when the user does not enter the correct number of integers or if the user provides a non numeric value.

Files, Descriptions, and Locations

File Name	Description	Location
ex2_setup.sh	Shell script that setups up the application	/home/w205/exercise_2
Create_postgres_db.py	Creates the postgres database and Tweetwordcount table.	/home/w205/exercise_2
Tweets.py	Tweet -spout	/home/w205/wordcount/src/spouts
Parse.py	Parse-tweet-bolt	/home/w205/wordcount/src/bolts
wordcount.py	Count-bolt. Updates Tweetwordcount table with count.	/home/w205/wordcount/src/bolts
Tweetwordcount.clj	Topology for the program	/home/w205/wordcount/topologies
finalresults.py	Shows counts for words in Tweetwordcount table	/home/w205/exercise_2
histogram.py	Shows words within a range of counts	/home/w205/exercise_2

Files Dependencies

File Name	Dependencies
ex2_setup.sh	postgres_create_db.py
Tweetwordcount.clj	Tweets.py, Parse.py, wordcount.py
Parse.py	Tweetwordcount.clj, Tweets.py
wordcount.py	Tweetwordcount.clj, Tweets.py, Parse.py, psycopg2 Library
Tweets.py	Tweepy Library
postgres_create_db.py	psycopg2 Library
finalresults.py	psycopg2 Library
histogram.py	psycopg2 Library