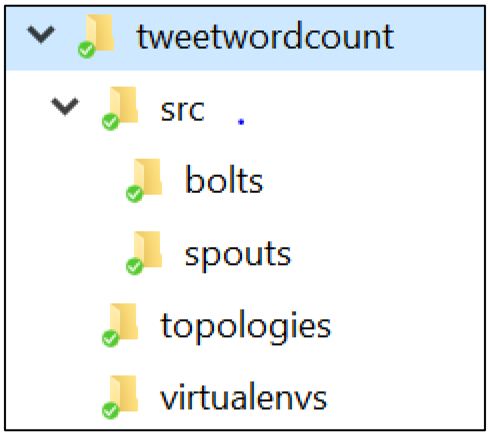
Application Description:

This application is intended to be used for performing simple analyses on twitter live stream data. The application aggregates data of the frequency of words used in real-time tweets provided through twitter API. The data is stored in a postgres table for which the provided python scripts can be used to get the desired data.

Directory and file structure:

The directory that houses the topology file and the bolts and spouts files for the storm application should look like the below folder structure. Within the topology file, the topology should correctly map to the respective bolts and spouts file names and the classes associated with the python bolts and spouts file.



Description of the architecture:

The application retrieves streams of live tweets from the twitter API which passes through the storm spout component. The spout component then passes it to the two bolt components. The first bolt component parses the tweet and splits it into words. That is then passed to the last bolt which takes each of the words passed from the previous bolt and updates/inserts the count of the words into a postgres table.

Two separate python analysis scripts have been prepared to analyze the word counts from the live stream tweets, which are described below.

Miscellaneous information to run the application (also found in Readme):

This application will require a couple of quick setup steps. The environment should already have storm setup with a python version that has tweepy and psycopg2 packages installed, along with Postgres installed. Using the AMI “UCB W205 Spring Ex 2 Image - ami-4cf9f826” will make things much easier. Once the EC2 instance is up and running postgres should be started.

First execute delete_tweet_table.py. This script will ensure that the table tweetwordcount is completely clean. Second execute create_tweet_table.py. This script will create tweetwordcount with the correct schema.

Once the above two scripts are executed change your directory to tweetwordcount and execute sparse run. This will start the streaming application. Once enough information is collected, do a keyboard interrupt to stop the application.

The finalresults.py and histogram.py are two scripts that will query the table for data.

- finalresults.py will fetch the count of a word, i.e. python finalresults.py hello
- finalresults.py executed alone will print out the full table
- histogram.py will fetch the count of a word between two numbers provided, i.e. python histogram.py 2 4