W205, Information Storage and Retrieval

Week #: 14
Exercise #: 2
Word Count: 974
Name: Gregory Ceccarelli
Date: 12/10/2015
Document Location & Name:
/root/EX2Tweetwordcount/documentation/Architecture.pdf

Introduction & Idea

Twitter data provides a valuable real time source of the web's (and accordingly those who use it) sentiment on nearly every topic / interest imaginable (current events, politics, religion, movies, etc). Being able to leverage and count words as they are being "tweeted" is a generally acceptable proxy for understanding what is important to people -- and in fact is used to gauge sentiment in a variety of ML applications. Leveraging a number of open source components, a reliable processing framework can be built and run to process real-time streaming tuples in Python. This data can be stored and analyzed both in real time and in batch capacities

The application that I built based off of the framework outlined in the Exercise 2 document does exactly that. All code and instructions necessary to run the application can be found in the gregce/ex2 private github repository.

Outline of Steps Taken

Setup Tasks

Following the general guideline of steps as instructed in the Exercise 2 document, the first step was to create an EC2 instance strictly to use for this exercise -- in this case a new instance was spun up using the ucbw205_complete_plus_postgres_PY2.7 AMI and an appropriate volume of data was attached. Accordingly, subsequent steps included:

- Once the AMI was live, I followed the step by step instructions in TheEasyButtonforYouAWSEnvironment_2.pdf
    - Including downloading and running the sh script from this location: https://s3.amazonaws.com/ucbdatasciencew205/setup_ucb_complete_plus_postgres.sh
    - When the application is being run from a fresh AMI where the [EX2 repo is cloned to /root], this same script is available in the location: /root/EX2Tweetwordcount/ami_setup/setup_ucb_complete_plus_postgres.sh
- First, lein, a dependency listed in Exercise 2 document was installed.

    - Commands leveraged to do this are saved and stored /root/EX2Tweetwordcount/ami_setup/setup_lein.sh
    - Following the installation of lein & streamparse, a project folder -- already referenced above, EX2Tweetwordcount, -- was created with the following command issued from /root: **sparse quickstart EX2Tweetwordcount**
        - This is unnecessary to replicate because the github repository that is cloned is actually the streamparse project with additional folders [ami_setup, serving, screenshots, documentation] saved within the project folder itself
- To avoid having to install a number of python libraries from scratch, I leveraged the Anaconda python distribution. This and the following dependencies were installed from the /root/EX2Tweetwordcount/ami_setup/conda_setup.sh
    - Downloads and installs Anaconda2-2.4.0-Linux-x86_64.sh
    - Installs a number of libraries relevant for this exercise including (tweepy, psycopg2 and streamparse)
- Because postgres is already installed on the AMI, the only thing that needed to be configured was the **tcount** database and the **tweetwordcount** table. This was and can be installed by running the /root/EX2Tweetwordcount/ami_setup/setup_postgres.sh
    - The sh script depends on the db_create.sql file also found in the same ami_setup
- The git@github.com:UC-Berkeley-I-School/w205-labs-exercises.git repo was cloned and the files moved to the appropriate directories within EX2Tweetwordcount (e.g. the Tweets.py "spout" was moved to /root/EX2Tweetwordcount/src/spouts/)
- Finally, environment variables were set and stored in .bashrc to ensure that the application would run correctly:
    - A copy of the contents of my .bashrc file is saved in /root/EX2Tweetwordcount/ami_setup/.bashrc

Code Creation / Modification Tasks

- Once the above setup tasks were completed, the following code was created/modified in order to create an application representative of Figure 1 in the Exercise 2 document. It was decided that /root/EX2Tweetwordcount/topologies/tweetwordcount.clj was unnecessary to modify as it was already reflective of Figure 1. Below is a summary:
    - Modified:
        - /root/EX2Tweetwordcount/src/spouts/tweets.py
            - Twitter Credentials were added to the twitter_credentials dict corresponding to the twitter application I created
        - /root/EX2Tweetwordcount/src/bolts/parse.py
            - Added logic to improve regex stripping of non-ascii characters
            - Ensured that the emitted word was always lowercase
        - /root/EX2Tweetwordcount/src/bolts/wordcount.py
            - Updated the initialize function to define connections to the local postgres db
            - Updated the process function to

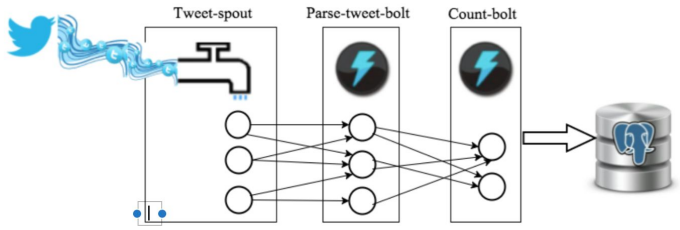
        - Create a dictionary of word counts using the Collections.Counter module
        - Check every time the dictionary grows to exceed 2000 items
        - Create a dataframe that is then written (appended) to the tweetwordcount table
        - Reads the dataset back and “reduces” the dataset from the previous append using the group by functionality
        - Replaces the contents of the table and,
        - Re-Initializes the word counter dictionary
- Created
    - /root/EX2Tweetwordcount/serving/finalresults.py
        - This is a command line application written to replicate the desired results as described in the Exercise 2 document.
            - It accepts either no input or a single word as an input
            - It can be run in a separate terminal as the application is processing bolts to interactively return results from the postgres database
    - /root/EX2Tweetwordcount/serving/histogram.py
        - This is a command line application written to replicate the desired results as described in the Exercise 2 document.
            - It accepts a lower and upper bound value separated by a comma
            - It too can be run in a separate terminal as the application is processing bolts to interactively return results from the postgres database
    - /root/EX2Tweetwordcount/serving/

Application Architecture / Assumptions

Once all of the above steps were completed (or if starting from a fresh AMI, following the readme doc), the assumption to replicate the output of designed architecture (pictured below), was that:

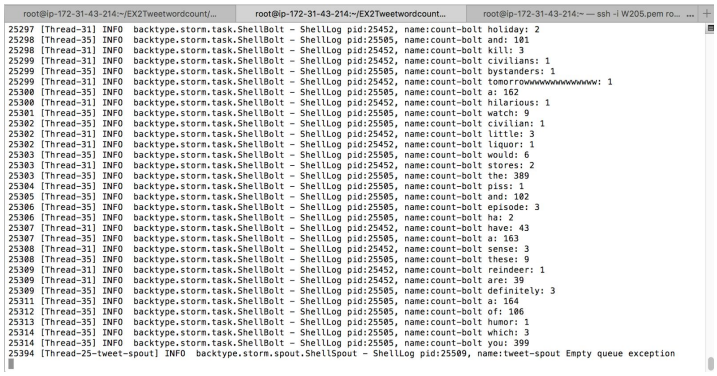
- A terminal prompt would be opened from which to run sparse run.
- A second terminal prompt could be opened and while the Storm topology runs / or ran, the user could execute both the finalresults.py and histogram.py scripts with the appropriate arguments to return results in real time fromt he processed and stored data
- Screenshots depicting these general assumptions can be found both below and in the folder /root/EX2Tweetwordcount/serving/

Architecture



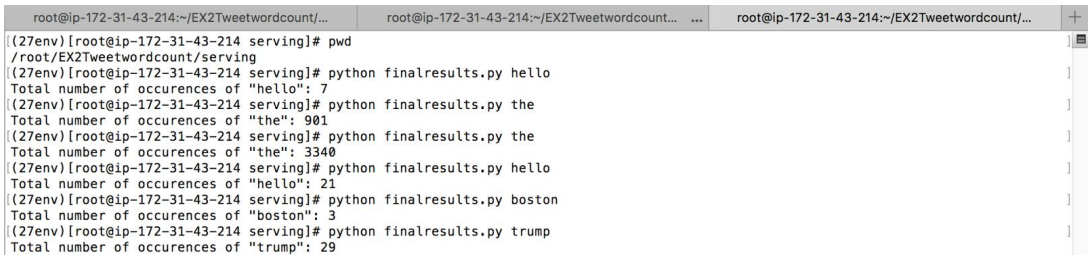
Other Screenshots

Twitter Stream



```
25297 [Thread-31] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25452, name:count-bolt holiday: 2
25298 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25505, name:count-bolt and: 101
25298 [Thread-31] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25452, name:count-bolt kill: 3
25299 [Thread-31] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25452, name:count-bolt civilians: 1
25299 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25505, name:count-bolt bystanders: 1
25299 [Thread-31] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25452, name:count-bolt tomorrowwwwwwwwwwwww: 1
25300 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25505, name:count-bolt a: 162
25300 [Thread-31] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25452, name:count-bolt hilarious: 1
25301 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25505, name:count-bolt watch: 9
25302 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25505, name:count-bolt civilian: 1
25302 [Thread-31] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25452, name:count-bolt little: 3
25302 [Thread-31] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25452, name:count-bolt liquor: 1
25303 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25505, name:count-bolt would: 6
25303 [Thread-31] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25452, name:count-bolt stores: 2
25303 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25505, name:count-bolt the: 389
25304 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25505, name:count-bolt piss: 1
25305 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25505, name:count-bolt and: 102
25306 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25505, name:count-bolt episode: 3
25306 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25505, name:count-bolt ha: 2
25307 [Thread-31] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25452, name:count-bolt have: 43
25307 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25505, name:count-bolt a: 163
25308 [Thread-31] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25452, name:count-bolt sense: 3
25308 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25505, name:count-bolt these: 9
25309 [Thread-31] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25452, name:count-bolt reindeer: 1
25309 [Thread-31] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25452, name:count-bolt are: 39
25309 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25505, name:count-bolt definitely: 3
25311 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25505, name:count-bolt a: 164
25312 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25505, name:count-bolt of: 106
25313 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25505, name:count-bolt humor: 1
25314 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25505, name:count-bolt which: 3
25314 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:25505, name:count-bolt you: 399
25394 [Thread-25-tweet-spout] INFO  backtype.storm.spout.ShellSpout - ShellLog pid:25509, name:tweet-spout Empty queue exception
```

Incremental Final Results Output



```
(27env)[root@ip-172-31-43-214 serving]# pwd
/root/EX2Tweetwordcount/serving
(27env)[root@ip-172-31-43-214 serving]# python finalresults.py hello
Total number of occurences of "hello": 7
(27env)[root@ip-172-31-43-214 serving]# python finalresults.py the
Total number of occurences of "the": 901
(27env)[root@ip-172-31-43-214 serving]# python finalresults.py the
Total number of occurences of "the": 3340
(27env)[root@ip-172-31-43-214 serving]# python finalresults.py hello
Total number of occurences of "hello": 21
(27env)[root@ip-172-31-43-214 serving]# python finalresults.py boston
Total number of occurences of "boston": 3
(27env)[root@ip-172-31-43-214 serving]# python finalresults.py trump
Total number of occurences of "trump": 29
```

Histogram Results Output

```
(27env)[root@ip-172-31-43-214 serving]# pwd
/root/EX2Tweetwordcount/serving
(27env)[root@ip-172-31-43-214 serving]# python histogram.py 5,20
        word  count
      coming     20
       going     20
      having     20
       first     20
      friday     20
     someone     20
          10     20
        them     20
       thats     19
           3     19
         him     19
       right     19
       women     19
         ass     18
        told     18
     amazing     18
         old     18
        ever     18
       thing     18
        such     18
          hi     18
        shit     18
       after     18
      before     18
```