LIVE TWITTER STREAM ANALYSIS

UCB W205 DATA STORAGE AND RETRIEVAL | by TED PHAM | April 9, 2017

APPLICATION SUMMARY

The application captures, processes, and stores live tweets and in "<word>: count" form in a Postgres database. This process runs continuously until user's termination input (Ctrl+c) is received. Analyses of the captured words and their occurrences can be performed by calling the accompanying Python serving scripts which are described in details in the following file structure section.

The codes were developed and tested to be fully functional in a Linux environment from Amazon Web Services EC2 instance of UCB's community AMI UCB MIDS W205 EX2-FULL. The AMI provides the required technologies for the application: 1. **HDFS**; 2. **Storm**; 3. **Postgres**; 4. **PsycoPG**; 5. **Tweepy**; 6. **Python** 2; 7. **Streamparse**.

APPLICATION TOPOLOGY

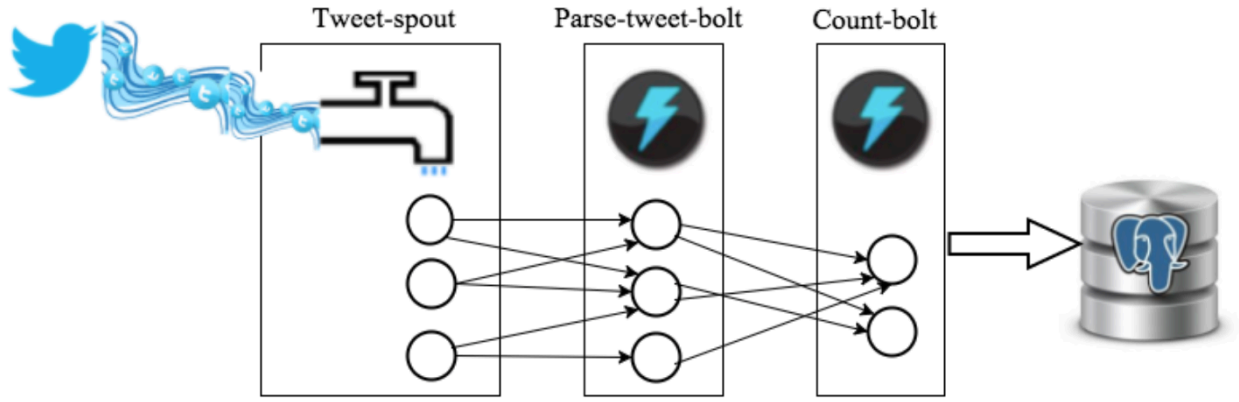

Figure 1: Application Topology

The application topology is the crucial component in capturing and real-time processing of twitter data. A tweet-spout pulls tweets from Twitter streaming API and runs on three threads. Users can obtain their own twitter credentials for the tweet-spout by creating a twitter application at https://apps.twitter.com . For convenience, a set of working credentials is provided. While the tweet-spout retrieves raw tweets, the two bolts parse-tweet-bolt and count-bolt processes tweets to valid words and counts the word respectively. In addition, the count-bolt also pushes and updates the data into a Postgres database. The Postgres database is initiated before each application is ran and contains only the tweets data of each individual run. The database is called tcount with data stored in a table called tweetwordcount.

FILE STRUCTURE

The application files are stored in the following Github repository https://github.com/tedapham/UCB_Tweepy.git , containing exercise_2 folder. The file structure described below is from within this exercise_2 folder.

Name of the program	Location	Description
extweetwordcount.clj	extweetwordcount/topologies/	topology for the application
tweets.py	extweetwordcount/src/spouts/	tweet-spout
parse.py	extweetwordcount/src/bolts/	parse-tweet-bolt
initialize.py	extweetwordcount/	Create a fresh Postgres database Needs to be run before the application
finalresults.py	extweetwordcount/	When passed a single word as an argument, returns the total number of word occurrences in the stream. Without an argument returns all the words in the stream, and their total count of occurrences, sorted alphabetically, one word per line.
histogram.py	extweetwordcount/	Gets two integers k1,k2 and returns all the words with a total number of occurrences between k1,k2
top20.py	extweetwordcount/	returns 20 words with the largest number of occurrences

EXECUTION INSTRUCTIONS

Whether the UCB's community AMI is used, make sure the seven platforms highlighted in the application summary are installed and Postgres server running on the respective Linux platform. Once the repo is cloned, navigate to extweetwordcount subfolder inside UCB_Tweepy/exercise_2.

Run the application as followed:

```
$ python initialize.py
$ sparse run
```

initialize.py must be run to create tcount database and tweetwordcount table in Postgres before sparse can be run.

Once the application is run successfully, a continuous log will be displayed such as this:

```
269933 [Thread-39] INFO  backtype.storm.task.ShellBolt - ShellLog pid:14470, name:count-bolt and: 1197
269934 [Thread-43] INFO  backtype.storm.task.ShellBolt - ShellLog pid:14507, name:count-bolt me: 942
269935 [Thread-39] INFO  backtype.storm.task.ShellBolt - ShellLog pid:14470, name:count-bolt the: 2817
269937 [Thread-43] INFO  backtype.storm.task.ShellBolt - ShellLog pid:14507, name:count-bolt New: 42
269940 [Thread-39] INFO  backtype.storm.task.ShellBolt - ShellLog pid:14470, name:count-bolt the: 2818
269941 [Thread-43] INFO  backtype.storm.task.ShellBolt - ShellLog pid:14507, name:count-bolt Orleans: 5
269942 [Thread-39] INFO  backtype.storm.task.ShellBolt - ShellLog pid:14470, name:count-bolt to: 2009
269943 [Thread-43] INFO  backtype.storm.task.ShellBolt - ShellLog pid:14507, name:count-bolt 8: 30
269944 [Thread-39] INFO  backtype.storm.task.ShellBolt - ShellLog pid:14470, name:count-bolt at: 345
269945 [Thread-43] INFO  backtype.storm.task.ShellBolt - ShellLog pid:14507, name:count-bolt First: 12
269946 [Thread-39] INFO  backtype.storm.task.ShellBolt - ShellLog pid:14470, name:count-bolt pastel: 4
269947 [Thread-43] INFO  backtype.storm.task.ShellBolt - ShellLog pid:14507, name:count-bolt happiest: 5
269948 [Thread-39] INFO  backtype.storm.task.ShellBolt - ShellLog pid:14470, name:count-bolt pastel: 5
269949 [Thread-43] INFO  backtype.storm.task.ShellBolt - ShellLog pid:14507, name:count-bolt Make: 39
269950 [Thread-39] INFO  backtype.storm.task.ShellBolt - ShellLog pid:14470, name:count-bolt pastel: 6
269951 [Thread-43] INFO  backtype.storm.task.ShellBolt - ShellLog pid:14507, name:count-bolt birthdays: 1
269953 [Thread-39] INFO  backtype.storm.task.ShellBolt - ShellLog pid:14470, name:count-bolt obscure: 3
269955 [Thread-39] INFO  backtype.storm.task.ShellBolt - ShellLog pid:14470, name:count-bolt did: 92
```

Stop the process with ctrl+ c. At this point, twitter data have been tabulated in Postgres table twitterwordcount.

Running finalresults.py, histogram.py, and top20.py for analyses. For example:

```
[w205@ip-172-31-0-117 extweetwordcount]$ python finalresults.py love
Total number of occurrences of love is : 474
[w205@ip-172-31-0-117 extweetwordcount]$ python finalresults.py world
Total number of occurrences of world is : 118
[w205@ip-172-31-0-117 extweetwordcount]$ python finalresults.py syria
syria was not found in tweetwordcount
[w205@ip-172-31-0-117 extweetwordcount]$ python finalresults.py trump
Total number of occurrences of trump is : 8
[w205@ip-172-31-0-117 extweetwordcount]$ python finalresults.py trump donald
Please only input only one word
[w205@ip-172-31-0-117 extweetwordcount]$ python finalresults.py fluxcapacitor
fluxcapacitor was not found in tweetwordcount
[w205@ip-172-31-0-117 extweetwordcount]$ python finalresults.py
Words sorted alphabetically and their occurrence count in the Tweeter Stream
#Birds : 2
$$ : 2
$1 : 2
$10 : 2
$100 : 4
$1000 : 2
$10000 : 3
$1000000 : 3
```

```
[w205@ip-172-31-0-117 extweetwordcount]$ python histogram.py 10000,900000
No word has occurence that falls between 10000 and 900000
[w205@ip-172-31-0-117 extweetwordcount]$ python histogram.py 200,200
There is nothing
[w205@ip-172-31-0-117 extweetwordcount]$ python histogram.py 159,159
There is nothing
[w205@ip-172-31-0-117 extweetwordcount]$ python histogram.py 100,100
THE: 100
[w205@ip-172-31-0-117 extweetwordcount]$ python histogram.py 4914 abc
Please input lower and upper bounds for occurences in form of 'integer1, integer2'
[w205@ip-172-31-0-117 extweetwordcount]$ python histogram.py 4914,abc
please input lower and upper bounds for occurrences in form of 'integer1,integer2'
[w205@ip-172-31-0-117 extweetwordcount]$ python histogram.py 100,115
video: 106
Thank: 115
us: 104
come: 115
than: 110
always: 105
better: 112
look: 107
US: 105
happy: 107
into: 107
THE: 100
shit: 114
I'll: 108
person: 105
So: 112
Trump: 115
And: 105
[w205@ip-172-31-0-117 extweetwordcount]$ python histogram.py
Please input lower and upper bounds for occurences in form of 'integer1, integer2'
```