W205 Exercise 2: Tweet Word Count

Erika Lawrence

Introduction

The goal of this exercise is to create an end to end streaming application which takes twitter data in through a storm spout, parses the words, and stores the count of each word occurrence in a postgres table. Two python scripts can be used to query the results: one for the count of a word supplied in a parameter, and another for the words with counts between numbers supplied in parameters.

Directory and File Structure

Files under the root/extweetwordcount directory are stored as follows:

- Storm Files (all required)
    - topologies:
        - tweetwordcount.clj file directing spouts and bolts processing
    - src/spouts:
        - “tweets.py’: pulls tweets in a dynamic stream
    - src/bolts:
        - parse.py: tokenizes all tweet words
        - wordcount.py: keeps a running count of tokens and stores results in the postgres table tweetwordcount
- Result Scripts
    - finalresults.py: displays the count of a parameter token word, or all word counts if no parameter is provided
    - histogram.py: displays words with counts between parameters given

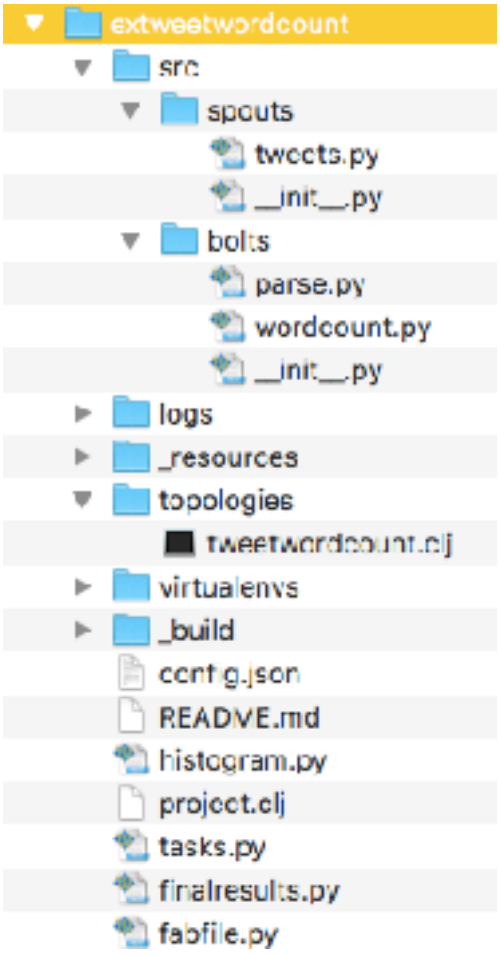

Fig 1: extweetwordcount file structure

Application Overview

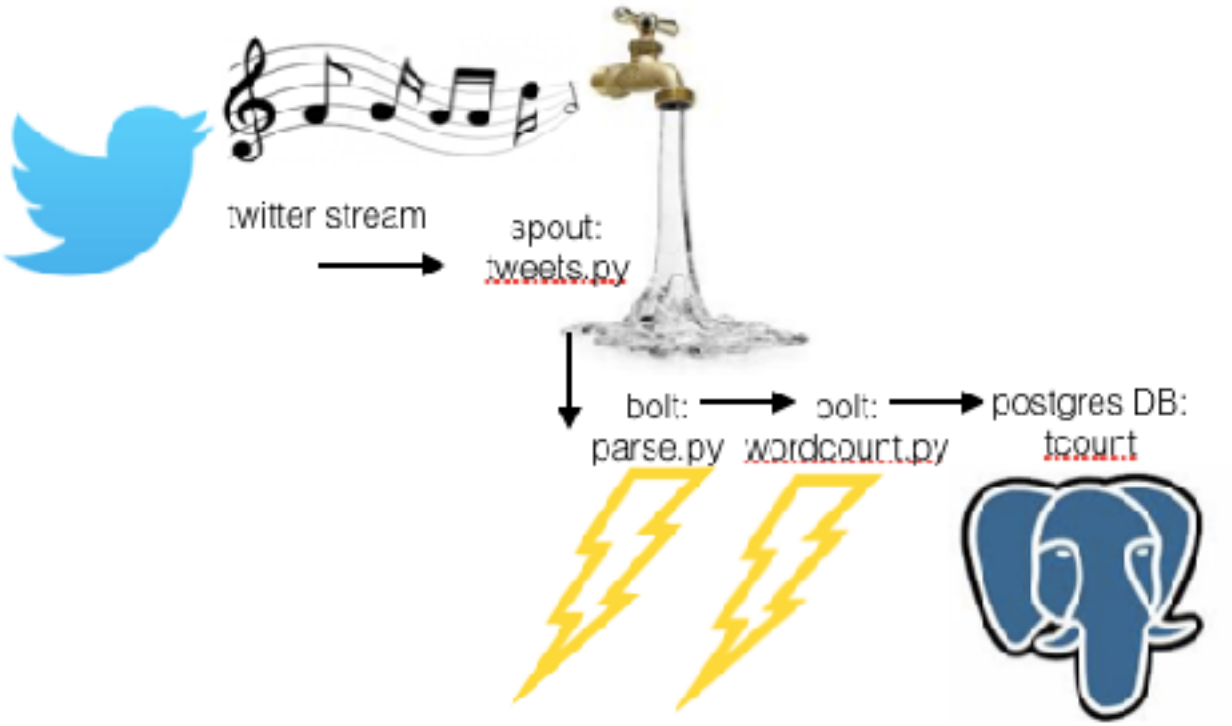

Fig 2: Application Architecture

The Tweet Word Count application consists of

1. a storm component which streams tweets through a spout into a word parser and counter bolt:

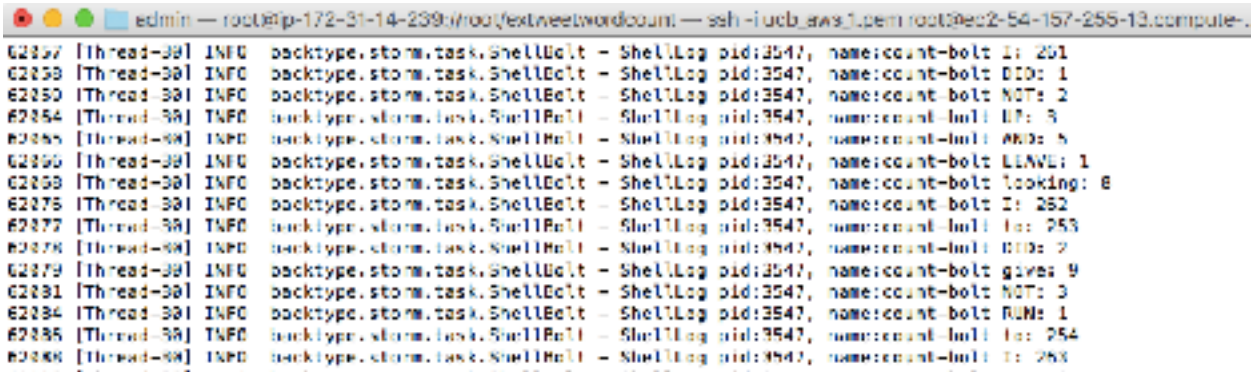

Fig 3: extweetwordcount streaming

2. a postgres database easily queried for result sets

```
[root@ip-172-31-14-239 extweetwordcount]# python finalresults.py Lorax
Total number of occurences of 'Lorax':1
[root@ip-172-31-14-239 extweetwordcount]# python finalresults.py puppies
Total number of occurences of 'puppies':1
[root@ip-172-31-14-239 extweetwordcount]# python finalresults.py was
Total number of occurences of 'was':11
[root@ip-172-31-14-239 extweetwordcount]#
```

Fig 4: final results.py script results with parameter

```
[root@ip-172-31-14-239 extweetwordcount]# python finalresults.py
('!', 2)
('!!', 1)
('#CleanPowerPlan', 1)
('$0.00', 1)
('%^*!', 1)
('&amp', 18)
('*coughs*', 2)
('-', 7)
('--', 1)
('-Wife', 1)
('/', 1)
('//forgets', 1)
('1', 1)
('10', 2)
('1000', 1)
('12', 1)
```

Fig 5: final results.py script results without parameter

```
[root@ip-172-31-14-239 extweetwordcount]# python histogram.py 8 9
("don't", 9)
('When', 9)
('get', 9)
('right', 9)
('who', 9)
('think', 9)
('what', 8)
('know', 8)
('as', 9)
('some', 8)
('or', 9)
[root@ip-172-31-14-239 extweetwordcount]#
```

Fig 6: histogram.py script results

Run the Application

1. Copy the extweetwordcount directory to an AWS instance using UCBMIDSW205EX2-FULL
2. Replace the existing twitter application credentials with a new set.
3. Attach a disk to the instance that includes postgres.
4. Create a postgres database and table:

```
psql -U postgres
create database tcount;
\c tcount

CREATE TABLE tweetwordcount
      (word TEXT PRIMARY KEY     NOT NULL,
      count INT     NOT NULL);

tcount-# \dt
            List of relations
 Schema |      Name      | Type  |  Owner
--------+----------------+-------+----------
 public | tweetwordcount | table | postgres
(1 row)

\q
```

5. From the extweetwordcount directory, type ‘sparse run’ and hit enter. Use Ctrl-C to break when ready
6. Use finalResults.py and histogram.py to review the results.

Results

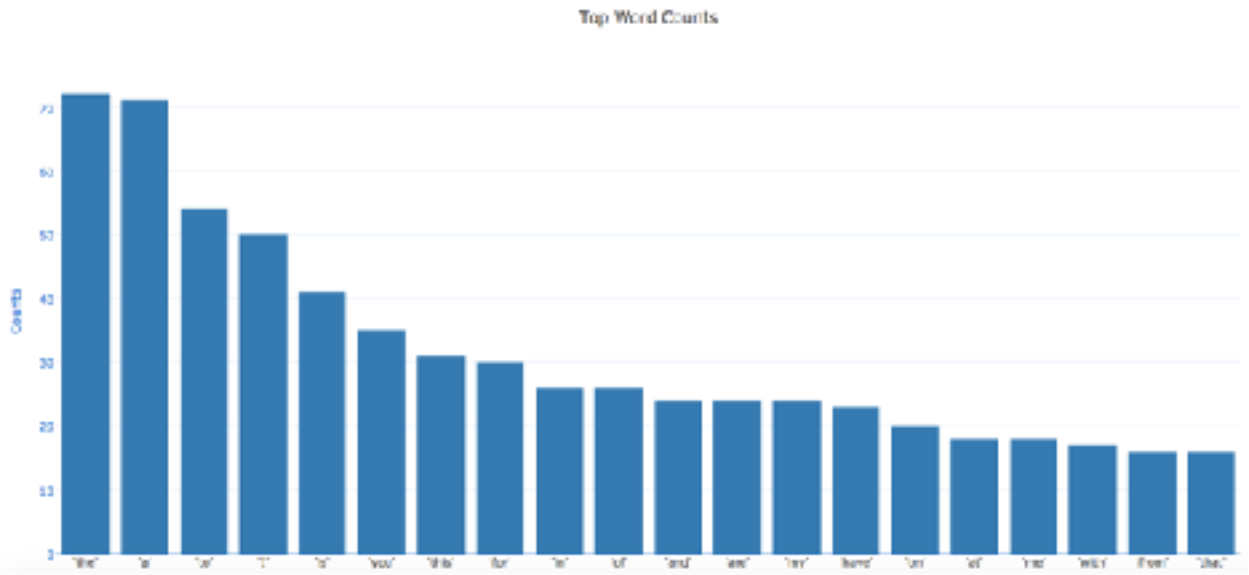

Fig 7: Top Word Counts