**Exercise 2: Tweet Word Count with Apache Storm**

**Keri Wheatley**

**UCB MIDS w205 Spring 2017**

**Application Idea**

This tweet word count application creates a real-time data stream of Twitter tweets using the Tweepy package and stores the counts of individual words to a Postgres database which can be read by two Python programs. The tweet word count application consists of a three-layered architecture: the Processing Layer, the Storage Layer, and the Application Layer. This architecture allows for a separation of responsibilities for processing, storing, and analyzing data.

1. Processing Layer

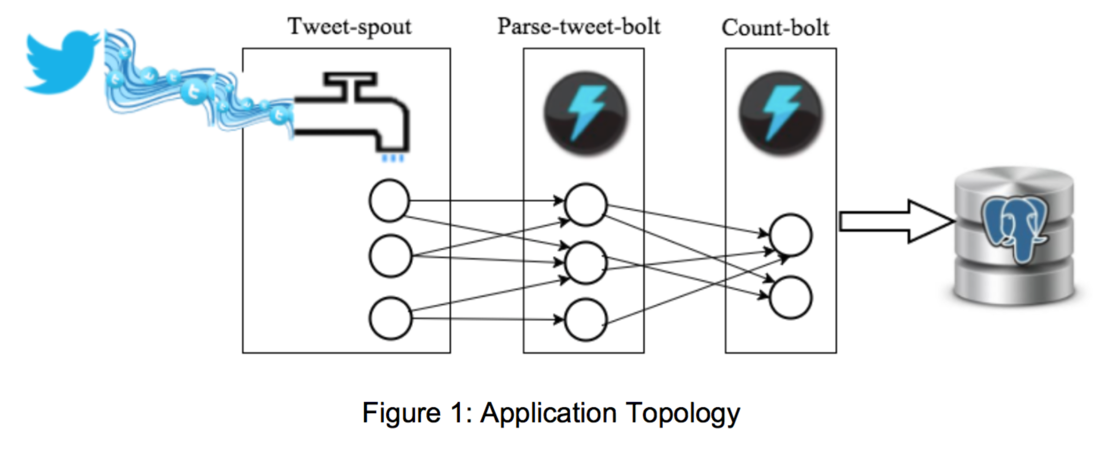
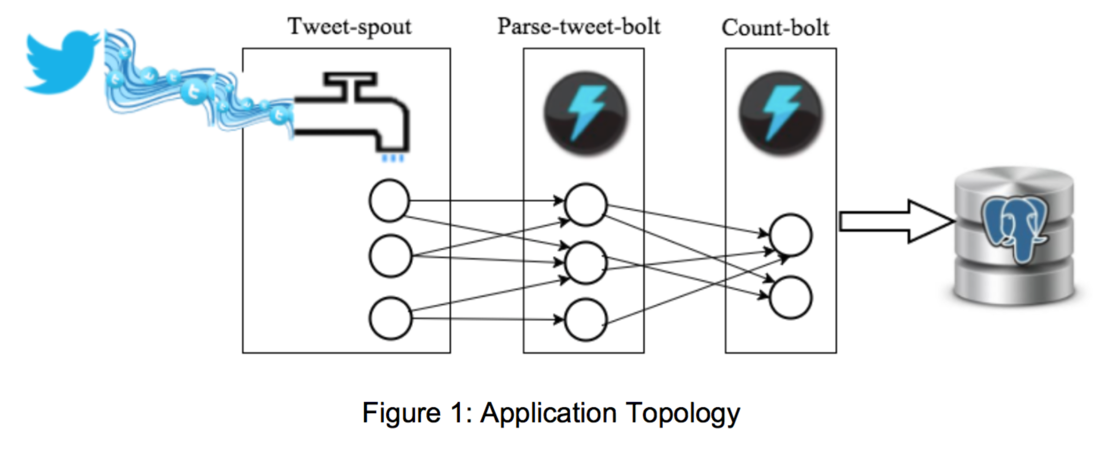
This layer consists of an Apache Storm topography, which includes one spout and two bolts as displayed in the architecture diagram below. The spout “tweet-spout” uses the Tweepy package and Twitter API to create a live tweet data stream. The tweets in this stream go through the bolt “parse-tweet-bolt” which uses the Streamparse package to split tweets into the individual words and then filters out hash tags, user mentions, retweet tags, urls, and punctuation. The resulting words get processed by the “count-bolt” which uses PsycoPG2 to connect to a Postgres database to write word counts to a table.

1. Storage Layer

The purpose of this layer is to store words and counts emitted by the processing layer. This layer consists of a Postgres database called Tcount. Words and their counts are written to the Tweetwordcount table within this database. New words are recorded and existing words are incremented as they flow from the processing layer into the storage layer.

1. Application Layer

The responsibility of this layer is to provide tools to query the storage layer and retrieve relevant information for the user. Two Python programs are provided as a part of this application. The finalresults.py program provides the user with a list counts for one or all words in the Tweetwordcount table. The histogram.py program provides the user with a list of words and their corresponding counts for a range of counts specified by the user.

 **Architecture Diagram**

**Application Layer**

**Storage Layer**

**Processing Layer**

**histogram.py**

**finalresults.py**

**Postgres database “Tcount"**

**wordcount.py**

**“count-bolt”**

**parse.py**

**“parse-tweet-bolt”**

**tweets.py**

**“tweet-spout”**

**Directory File Structure**

|  |  |  |
| --- | --- | --- |
| **File Name** | **Location** | **Description** |
| extweetwordcount.clj | exercise\_2/extweetwordcount/topologies/ | Topology for the Apache Storm application |
| tweets.py | exercise\_2/extweetwordcount/src/spouts/ | Creates Twitter feed; tweet-spout in topology |
| parse.py | exercise\_2/extweetwordcount/src/bolts/ | Parses and cleans Tweets from tweet-spout; parse-tweet-bolt in topology |
| wordcount.py | exercise\_2/extweetwordcount/src/bolts/ | Counts and stores words from parse-tweet-bolt into Postgres database; count-bolt in topology |
| screenshot-twitterStream.png | exercise\_2/screenshots/ | Screenshot of activated Apache Storm application |
| finalresults.py | exercise\_2/scripts/ | Returns word count for one or all words; optional input <word> |
| screenshot-finalResults.png | exercise\_2/screenshots/ | Screenshot of running finalresults.py program |
| histogram.py | exercise\_2/scripts/ | Returns the word counts for words within a specified count range; requires inputs <min\_count>, <max\_count> |
| screenshot-histogram.png | exercise\_2/screenshots/ | Screenshot of running histogram.py program |
| readme.txt | exercise\_2/ | Contains instructions on setup and running the application |
| Plot.png | exercise\_2/ | Bar plot of 20 most common words |

**File Dependencies**

tweets.py Stores Twitter Credentials

**Database Structure**

Database tcount  
User postgres

Password pass  
Table tweetwordcount (word TEXT, count INT)

**GitHub Repository**

git@github.com:keriwheatley/w205-spring-17-labs-exercises.git

**Required packages:**

Apache Storm, Amazon EC2, Python, Twitter API, Streamparse, Postgres, PsycoPG, Tweepy

**How to Run Application:**

1. Run Storm program:

cd w205-spring-17-labs-exercises/exercise\_2/extweetwordcount

sparse run

2. Exit Storm program:

ctrl + c

3. Run program to return counts for all words:

cd ~ && cd w205-spring-17-labs-exercises/exercise\_2/scripts

python finalresults.py

4. Run program to return count for one word:

cd ~ && cd w205-spring-17-labs-exercises/exercise\_2/scripts

python finalresults.py *<word>*

5. Run program to return word counts in a specified range:

cd ~ && cd w205-spring-17-labs-exercises/exercise\_2/scripts

python histogram.py *<min\_count> , <max\_count>*