Exercise 2: Architecture

The purpose of this exercise was to analyze live Twitter data. The data path is as follows:

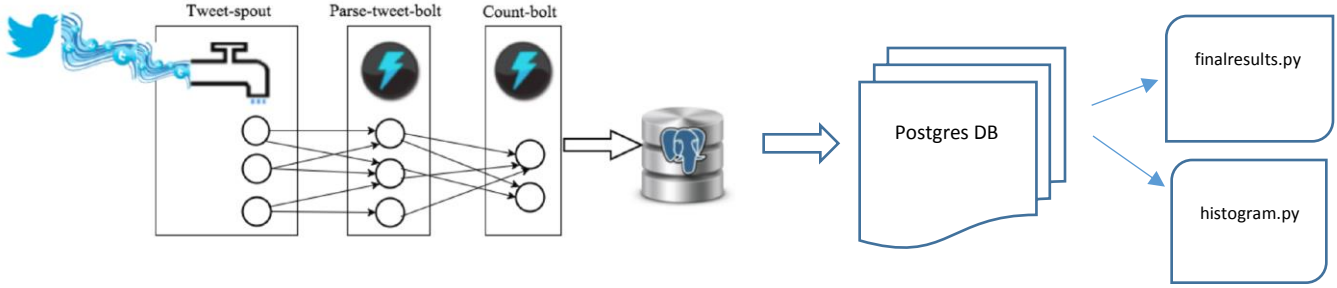
-A Tweet stream, which is captured in tweet.py (spout).

-A Parse-tweet bolt parses words coming from the tweet spout

-A count bolt, which counts the words and stores them in postgres (table tcount).

-finalresults.py, an app that returns the total count of each word from the stream

-histogram.py, which returns all the words with a total number of occurrences greater than or equal to a first number, and less than or equal to another number.



The exercise structure is under extweetwordcount folder:

extweetwordcount
- finalresults.py
- histogram.py
-topologies
- tweetwordcount.clj
-src
- bolts
- wordcount.py
- parse.py

- spouts
- tweets.py

The topology, tweetwordcount.clj, is the following:

```
(ns extweetwordcount
 (:use    [streamparse.specs])
 (:gen-class))


(defn extweetwordcount [options]
  [
   ;; spout configuration
   {"tweet-spout" (python-spout-spec
        options
        "spouts.tweets.Tweets"
        ["tweet"]
        :p 1
        )
   }
   ;; bolt configuration
   {"parse-tweet-bolt" (python-bolt-spec
        options
        {"tweet-spout" :shuffle}
        "bolts.parse.ParseTweet"
        ["word"]
        :p 2
        )
    "count-bolt" (python-bolt-spec
        options
        {"parse-tweet-bolt" ["word"]}
        "bolts.wordcount.WordCounter"
        ["word" "count"]
        :p 2
        )
   }
 ]
)
```

For running twitter stream sparse application:

1.- Inside extweetwordcount folder, run the storm streamparse:

```
$sparse run
```

2.- Run finalresults.py to return the total count of every word in the stream, or a given word.

```
$python finalresults.py
```

3.- Run histogram.py to find the total number of occurrences greater than or equal to a first number, and less than or equal to another number.

```
$python histogram.py
```