About this application.
Processing and analyzing data in real time allows getting insights into “what is trending now?”. Simple word count may not be as powerful tool as natural language processing, but it gives an application user some hints of what the people on social media are talking about, which in turn serves as a starting point and search process fine tune could be done with other tools.
The purpose of the application is

1. to capture live Twitter streaming data,
2. parse the tweets/extract words from tweets
3. implement the word count
4. integrate all the components above (1-3)
5. input the word count results into Postgres database
6. store data in Postgres database
7. retrieve data from database for further analysis

Implementation tools

Technology used:
Twitter API
Tweepy (authentication, connection, creating and destroying the session, reading incoming messages, and partially routing messages.
Apache Storm (processing streams of data)
Python
Postgres
Streamparse (integrates Python with Apache Storm)
PsycoPG (provides connection to Postgres, using Python programming language)

AWS: Application is set up on AWS EC2(virtual machine) using **UCB MIDS W205 EX2-FULL**
AMI ID: ami-d4dd4ec3

Access to files: **Natallia112202/w205_2017_spring/**exercise_2/extweetwordcount (directory cloned from EC2)
Figure below depict basic **components** of the application. **Topology file:**
Natallia112202/**w205_2017_spring/exercise_2/**extweetwordcount/topologies/wordcount.clj

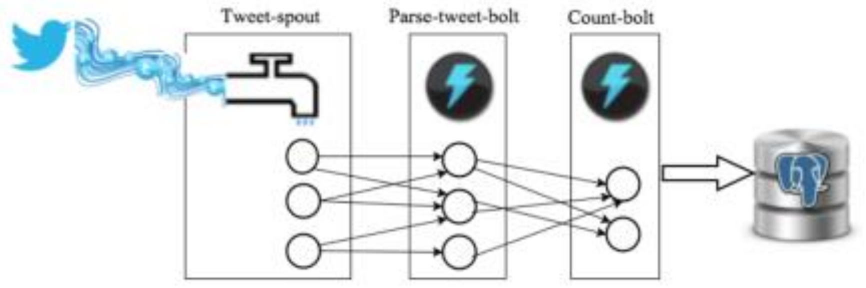

Figure 1: Application Topology

File system structure and application summary.

task	Files/directories	Topology component	Files created or modified from original
-capturing live Tweets	~/exercise_2/extweetwordcount/ src/spouts/tweets.py ~/exercise_2/extweetwordcount/ Twittercredentials.py	**Tweet-spout**	tweets.py and Twittercredentials.py modified by entering Twitter API keys: consumer key/consumer secret key & access token/access token secret from Twitter API
-parsing the tweets/extract words from tweets	~/exercise_2/extweetwordcount/ src/bolts/parse.py	**Parse-tweet-bolt**	Not modified
-implementing the word count and inputting the word count results into Postgres database	~/exercise_2/extweetwordcount/ src/bolts/wordcount.py	**Count-tweet-bolt**	wordcount.py file modified to provide connection to Postgres (adding additional code lines) to open connection to Postgres, insert the word/word count into Postgres database, update word/word count, save changes and close connection.
-Integrating topology components	~/exercise_2/extweetwordcount/ topologies/wordcount.clj	**Topology**	Topology file wordcount.clj modified to reflect topology structure Figure 1
retrieve data from database for further analysis(1)	~/exercise_2/finalresults.py	**Serving script 1**	created file. When passed a single word as an argument, finalresults.py returns the total number of word occurrences in the stream. Running finalresults.py without an argument returns all the words in the stream, and their total count of occurrences, sorted alphabetically, one word per line
retrieve data from database for further analysis(2)	~/exercise_2/histogram.py	**Serving script 2**	created file. histogram.py script gets two integers int1,int2 and returns all the words with a total number of occurrences greater than or equal to int1, and less than or equal to int2; if no words have count between [int1 and int2] returns “no words with count between int1 and int2”