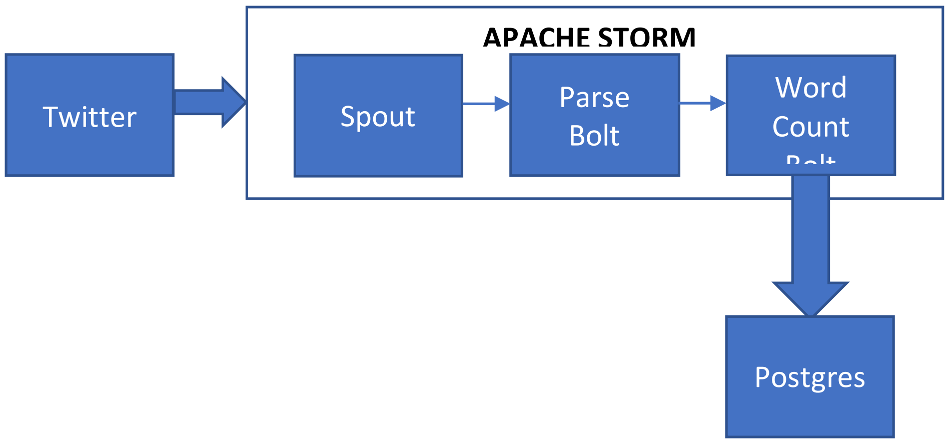
Exercise 2- W205 Section 3 Spring 2017
Sanjay Dorairaj

This document describes the architecture of the W205 Exercise 2Twitter application. This project demonstrates the integration of Apache Storm, Postgres and Twitter. The tweepy library is used to access twitter. Postgres access via python is made possible via the psycopg2 library. The program is written using Python and the server configuration is Centos 6.

Description of the Architecture

The diagram below captures the high level architecture for this project.



Tweets are accessed via APIs exposed via the tweepy library. Authentication keys to access tweets are retrieved by registration on the Twitter developer website. A storm spout retrieves tweets and extracts the text portion of the tweet from the encapsulating JSON payload. The text is then forwarded to the parse bolt where each word is split. The word and the count are then forwarded to the word count tweet where word counts are updated and written to the postgres database.

Two scripts finalresults.py and histogram.py allow the user to fetch the counts of tweets from the database.

Directory Structure

The below section captures the directory structure for this application.

```
├── Exercise-2.md
├── Exercise-2.pdf
├── extweetwordcount
│   ├── _build
│   │       └── leiningen.core.classpath.extract-native-dependencies
│   ├── config.json
│   ├── fabfile.py
│   ├── logs
│   ├── project.clj
│   ├── README.md
│   ├── _resources
│   │   └── resources
│   │       ├── bolts
│   │       │   ├── __init__.py
│   │       │   ├── parse.py
│   │       │   └── wordcount.py
│   │       └── spouts
│   │           ├── __init__.py
│   │           └── tweets.py
│   ├── src
│   │   ├── bolts
│   │   │   ├── __init__.py
│   │   │   ├── parse.py
│   │   │   └── wordcount.py
│   │   └── spouts
│   │       ├── __init__.py
│   │       └── tweets.py
│   ├── tasks.py
│   ├── topologies
│   │   └── extweetwordcount.clj
│   └── virtualenvs
│       └── wordcount.txt
├── finalresults.py
├── Grading_Rubric.md
├── histogram.py
├── images
│   ├── 1_topology.png
│   ├── 2_create_new_app.png
│   ├── 3_create_application.png
│   ├── 4_create_token.png
│   └── 5_access_token.png
├── initdb.py
├── README.md
├── Twittercredentials.py

19 directories, 33 files
```

Dependencies

AMI Configuration

This project uses the UCB-AMI-FULL AMI that is available in the AWS Community AMIs. The following steps were taken in order to resolve all the dependencies for the project.

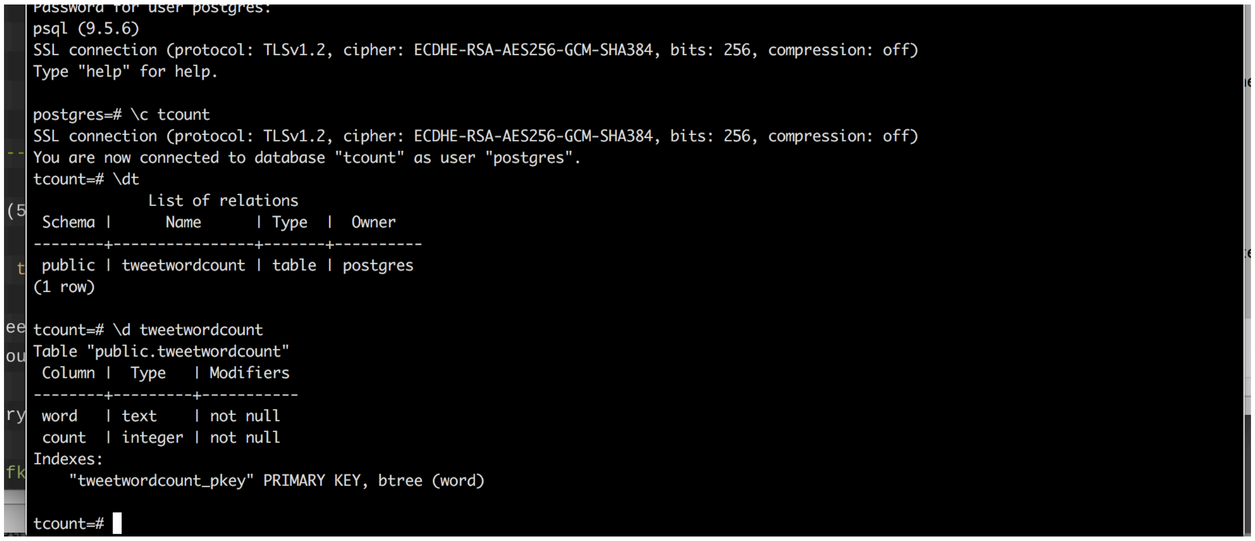
1. Create an EBS volume and attach to the instance.
2. Re-run the postgres/Hadoop initialization and startup scripts in Lab2.
3. Install psycopg2
4. Install tweepy
5. Fetch exercise_2 from the ucb_w205 git repository at https://github.com/UC-Berkeley-I-School/w205-spring-17-labs-exercises/tree/master/exercise_2
6. Register at Twitter developer site and obtain API authentication keys.

Database/Table structure

The database has the below tables and associated schema for the purposes of this project.

- Database name – tweet
- Table name – tweetwordcount

Shown below is a screenshot of the database, table and schema for the tweetwordcount table.

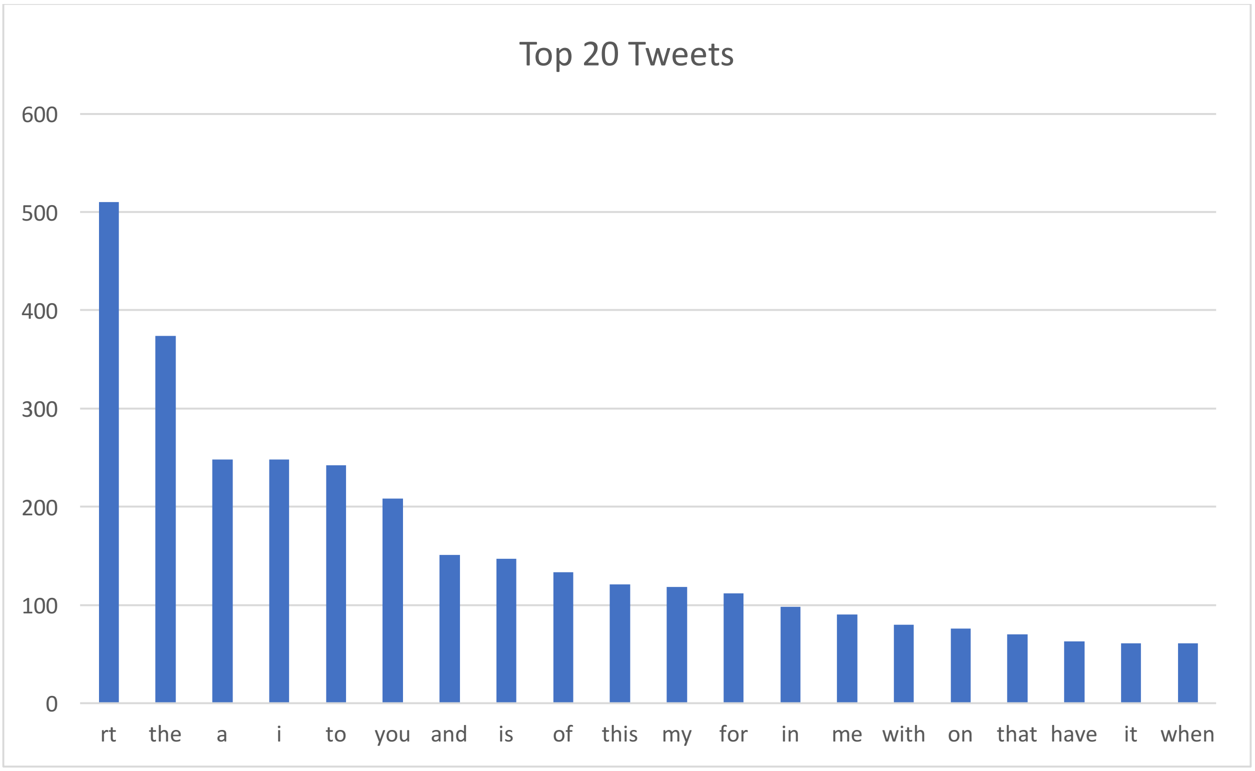


```
Password for user postgres:
psql (9.5.6)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=# \c tcount
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
You are now connected to database "tcount" as user "postgres".
tcount=# \dt
             List of relations
 Schema |      Name      | Type  |  Owner
--------+----------------+-------+----------
 public | tweetwordcount | table | postgres
(1 row)

tcount=# \d tweetwordcount
Table "public.tweetwordcount"
 Column |  Type   | Modifiers
--------+---------+-----------
 word   | text    | not null
 count  | integer | not null
Indexes:
    "tweetwordcount_pkey" PRIMARY KEY, btree (word)

tcount=#
```

Code Changes

1. Move tweetwordcount to extweetwordcount and make the following changes

   a. Modify the tweety.py spout under src/spout to add code to connect to Twitter and fetch tweets.
   b. Modify wordcount.py bolt under src/bolts to write words and their corresponding counts to the postgres database.
2. Create a python script called finalresults.py
   a. Add command line argument support to allow user to enter a word at the command line.
   b. If user enters a parameter, look up the corresponding parameter and count from the database.
   c. If user does not enter any command line parameter, retrieve all words and their corresponding counts in ascending order and display it on the console.
3. Generating top 20 words
   a. All tweet words are extracted from the database.
   b. The below bash command is executed on the result in order to get the top 20 tweets.
      i.
      ```
      python finalresults.py | cut -d' ' -f4,9 | sort -
      k2,2n | tr ' ' ','
      ```


Top 20 Tweets
rt
the
a
i
to
you
and
is
of
this
my
for
in
me
with
on
that
have
it
when