Exercise 2 – Processing Twitter data

Diana Rodenberger – w205-1

Purpose

Gets insights from my home timeline tweets. The application uses the Twitter streaming API (tweepy) to access the tweets from my home timeline and identifies trending topics by calculating the most frequent used words.

Instructions

1. Before running the application, verify that you have streamparse, python 2.7.6 and postgress installed.
2. Start postgres from directory /data

```
$ ./start_postgres.sh
```

3. Run all applications under user 'root'
4. Clone repository https://github.com/DianaRodenberger/w205-exercise_2.git

```
$ git clone https://github.com/DianaRodenberger/w205-exercise_2.git
```

5. Create database in postgres using script in directory w205-exercise_2

```
$python createdb.py
```

6. Go to directory Tweetwordcount

```
$cd Tweetwordcount
```

7. Run streaming application

```
$ sparse run
```

8. Stop execution after 5 minutes or longer (5 minutes recommended)

```
hit keys Ctrl+C
```

9. Go to directory w205-exercise_2

```
$cd w205-exercise_2
```

10. Run serving scripts
    a. Run script to get the number of occurrences for a given word

```
$python finalresults.py Trump
```

    b. Or run script to get the number of occurrences for each word collected from the stream of tweets

```
$python finalresults.py
```

c. Run script to get words that have occurences within the input range

```
$python histogram.py 150 200
```

d. Run script to get the top 20 words in a csv file. The csv file can be used to create a plot.

```
$python top20words_csv.py
```

Database

- Database name: Tcount
- Table name: Tweetwordcount
- Schema: Tweetwordcount(word, count)

Streaming Application

Topology

The topology of the streaming application is composed of 3 Tweet spouts, 3 Parse-tweet bolts and 2 Count bolts (3,3,2)

Tweet spouts

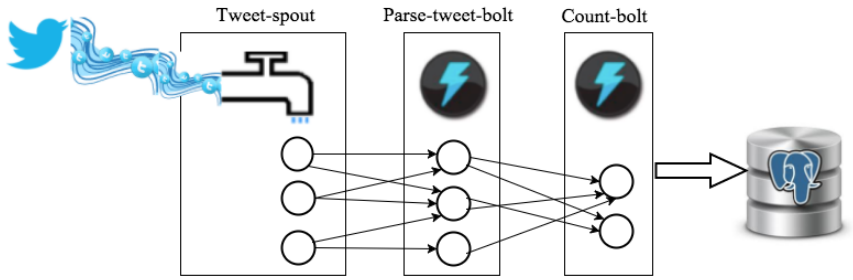
The spout establishes a connection to my twitter home timeline through the Twitter streaming API and establishes a pipeline to listen to tweets posted on my timeline. Emits tweets

Parse-tweet bolt

Process tweets and filters hash tags, RT and URLs.

Count bolt

This count bolt counts the words emitted by the Parse-tweet bolt and stores this information in a Postgres database 'Tcount', table 'Tweetwordcount'. The Count bolt attempts to insert the tuple to the table Tweetwordcount; if the tuple already exists then it updates the tuple with the new count.



Serving Layer

There are three scripts created to analyze results of the streaming twitter application.

Script **finalresults.py** returns the number of occurrences for the word provided. If no word is provided, it returns the number of occurrences for all words stored in the database.

Script **histogram.py** returns the words with number of occurrences that fall between the range provided.

Script **top20words_csv.py** creates a csv file in the same directory where the script is located with the top 20 words in my twitter timeline.

Directories and Files

File	Description	Location
Createdb.py	Creates or recreates a database to store tweet word counts	w205-exercise_2\
Tweetwordcount.py	Topology of the program	w205-exercise_2\tweetwordcount\topologies\
tweets.py	Spout	w205-exercise_2\tweetwordcount\topologies\
parse.py	bolt	w205-exercise_2\tweetwordcount\src\bolts
wordcount.py	bolt	w205-exercise_2\tweetwordcount\src\bolts
Finalresults.py	Gets the number of occurrences for a given word or all words	w205-exercise_2\
Histogram.py	Extracts words with number of occurrences within the range provided	w205-exercise_2\
Top20words_csv.py	Creates csv file with top 20 words in timeline	w205-exercise_2\
Tweetwords.csv	Top 20 words in my timeline	w205-exercise_2\

Additional notes

- All scripts should run under the user ‘root’
- Port 5432 in the EC2 instance must be open
- User needs to create a twitter account and a twitter application prior to running the streaming application.
- User needs to create a sparse project ‘Tweeterwordcount’