Tweet Counting Architecture

John Kenney

W205

READ-ME.txt:

1. Assumptions:
    a. Postgres is started on instance running
    b. Database tcount has not been created
        i. If it has, please use alternate script
2. Instructions
    a. Clone Repository with:
        i. Git init
        ii. Git clone
    b. Change into directory
        i. Cd w205-Assignments/tweetwordcount
    c. Run scripts
        i. If tcount DB has not been created
            1. Chmod +x ./run.sh
            2. ./run.sh
        ii. If tcount DB has been created
            1. Chmod +x ./run-alternate.sh
            2. ./run-alternate.sh
    d. Tweet stream will automatically commence
        i. Since you will be running as the root, you may have to press “Enter” to override the LEIN warning
        ii. Press Ctrl+z to exit from stream when satisfied

Run.Sh/Run-alt.sh

The Run.Sh script sets up everything we need to run to stream the tweets into our Postgres database. We load psycopg2, which allows us to execute queries directly into our Tweetwordcount table. We also set up our Database and Table.

CreateDatabase.py

This creates our database. Note, if the database already exists, run "Run-alt.sh". We need to set the Isolation Level to Autocommit to ensure that the database is created without error or interruption.

CreateTable.py

This script connects us to the Database, Drops the Tweetwordcount table if it exists, and the creates a fresh table. The reason we drop the table is to start from scratch, since we are only interested in the counts for the tweet listening session, not historical counts. Note, this script should be executed prior to any running of the storm topology to clear the previous cache.

Twitter API (Tweepy)

Hello-stream-twitter.py

This code was created to show that we had successfully established a connection through the Twitter API and could stream tweets. This code does a few things. First, it creates a member of the StreamListener class from the Tweepy library. It then looks in the TwitterCredentials.py for four special hex values, which authenticate both the user (Consumer key and secret) and the application (access token and secret). These values goes through a checksum algorithm to generate the TwitterCredentials.pyc file, which allows the session to begin.

Once the stream is generated, a timer begins. A simple JSON is created for each tweet as it comes in, at which point it is printed to the console. After 60 seconds, the timer ends and the session is terminated. Note, the code was modified so that there is no filter

```
rt @ajwanidaniel: dwight schrute° https://t.co/8yut5gkizj
all of them are shit tbh @lencer_b
carditunes: carditunes: https://t.co/mjczfv6txj #selfiesfornormani #makeafilmawfullybritish #matchgame november 21, 2016 at 09:30am
naeun adalah member pertama yang diperkenalkan oleh cube entertaimen sebagai member a-pink
rt @siska_mobi69ae: https://t.co/gslyb0w6iw
come hangout with me on #bigolive. https://t.co/t0cp7l7n3a https://t.co/pcpdu2fe6d
why do females think being annoying is attractive? it's not
imanfs165 hai kak, untuk smsnya bisa diabaikan aja ya. terindikasi layanan kontent berbayar. thanks ^riri
@misslola328 fn parti patriote et non raciste !!  @le_figaro
rt @adamlevine: everything i need is right here. (beach optional) https://t.co/bsbnfl38or
@inspire_us prove yourself to yourself not others.
@czabe 1) that was funny
2) that i don't understand
3) that was very very stupid
rt @favsofcamila: https://t.co/hwdpp7jyxv
home comforts: hull city set sights on restoring kcom pride - yorkshire post - soccer bets predictions? &gt; https://t.co/eivx5jshqx
rt @shawtydreads20: ain't no half way fuckin wit me
rt @oanderle: it's been a busy weekend.
rt @sojonghyun: what's up jinki https://t.co/smmsmbqj9t
ansa.it: giubileo, il papa chiude la porta santa di san pietro - giubileo. https://t.co/gfezxlbscj
hahahaaa liburan.... https://t.co/ezzreupgxd
@jacyndadooley i'm ugly but i'm funny https://t.co/ll17oxv81v
........ https://t.co/5tyivgybl7
rt @sleepygirl: @beervheart ti amo
rt @borntalead: twitter ain't just twitter when christ is involved.
rt @saucinese: aquarius: played you because you got boring
https://t.co/5brn8278yl
@majesticalexis ugh same
electives include courses in russian https://t.co/nzefkec7ct
rt @justinbashbro: black don't crack lol https://t.co/9fjzmqyorz
rt @4theblunted: so, there's this girl.. and she's pretty cute..
rt @syahirahsaruan: everyone knows my weakness and don't blame me if i become more rebellious and lost my dignity.
rt @meninisttweet: oh hey i found where feminists come from https://t.co/fkaldx1a0h
serious ah tak suka! eeeeiii!!
rt @cgguy44: straight person: if you're gay then just choose not to be gay
```

Fig. 1: Modified Hello-Stream-Twitter Application Running

Storm

TweetWordCount.clj

This file is used to map the topology our Storm Application. We will see three types of processes running. There are three instances of the spout, which receive and pass raw tweets from the Twitter API. There are three parse bolts, which receive shuffled tweets from the spouts, clean the tweets, and send along the individual words. Finally, there are two count bolts, which receive words grouped by field from the parse bolts, count the instances seen for each of the words and then create or update the Postgres table with the word and its counts.

Tweets.py

This is the spout program for the topology. It handles authentication immediately when the spout initializes, passing the four hex values to the twitter API, which allows the listener to be generated.

There are a number of parameters that were modified from the original code to allow the tweets to be processed, since the stream would die after the first 15 seconds, resulting in numerous "Empty Queue" exceptions, which indicated that the system was trying to process a tweet that had not arrived. To overcome this, the max Queue size was increased from 10 to 1,000 to prevent useful tweets from being blocked as well as increasing the Queue.Get timeout from 0.1 seconds to 10 seconds to ensure the task completed as intended, no matter what quantity of memory was available. Finally, the Queue.Put timeout was expanded from 0.01 to 1 to ensure that every Tweet was eventually processed, further reducing the frequency of Empty Queue Exceptions.

The original code filtered the stream to ensure any relevant tweet contained a series of common words. This series was expanded to include the Top 20 most commonly used words in the English language, as indicated by Wikipedia. This was done to expand the boundary to encompass as many tweets coming from the stream as possible.

```
34630 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:6969, name:count-bolt our: 3
34639 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:6969, name:count-bolt American: 1
34667 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:6969, name:count-bolt values: 1
34683 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:6969, name:count-bolt and: 13
34713 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:6969, name:count-bolt work: 1
34729 [Thread-29-tweet-spout] INFO  backtype.storm.spout.ShellSpout - ShellLog pid:7000, name:tweet-spout Empty queue exception
34732 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:6969, name:count-bolt on: 8
34748 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:6969, name:count-bolt behalf: 1
34768 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:6969, name:count-bolt of: 7
34784 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:6969, name:count-bolt all: 2
34802 [Thread-15-tweet-spout] INFO  backtype.storm.spout.ShellSpout - ShellLog pid:6970, name:tweet-spout Empty queue exception
34802 [Thread-35] INFO  backtype.storm.task.ShellBolt - ShellLog pid:6969, name:count-bolt of: 8
34831 [Thread-13-tweet-spout] INFO  backtype.storm.spout.ShellSpout - ShellLog pid:6968, name:tweet-spout Empty queue exception
34931 [Thread-29-tweet-spout] INFO  backtype.storm.spout.ShellSpout - ShellLog pid:7000, name:tweet-spout Empty queue exception
35003 [Thread-15-tweet-spout] INFO  backtype.storm.spout.ShellSpout - ShellLog pid:6970, name:tweet-spout Empty queue exception
35033 [Thread-13-tweet-spout] INFO  backtype.storm.spout.ShellSpout - ShellLog pid:6968, name:tweet-spout Empty queue exception
35132 [Thread-29-tweet-spout] INFO  backtype.storm.spout.ShellSpout - ShellLog pid:7000, name:tweet-spout Empty queue exception
35205 [Thread-15-tweet-spout] INFO  backtype.storm.spout.ShellSpout - ShellLog pid:6970, name:tweet-spout Empty queue exception
35234 [Thread-13-tweet-spout] INFO  backtype.storm.spout.ShellSpout - ShellLog pid:6968, name:tweet-spout Empty queue exception
35334 [Thread-29-tweet-spout] INFO  backtype.storm.spout.ShellSpout - ShellLog pid:7000, name:tweet-spout Empty queue exception
35407 [Thread-15-tweet-spout] INFO  backtype.storm.spout.ShellSpout - ShellLog pid:6970, name:tweet-spout Empty queue exception
35436 [Thread-13-tweet-spout] INFO  backtype.storm.spout.ShellSpout - ShellLog pid:6968, name:tweet-spout Empty queue exception
35536 [Thread-29-tweet-spout] INFO  backtype.storm.spout.ShellSpout - ShellLog pid:7000, name:tweet-spout Empty queue exception
35609 [Thread-15-tweet-spout] INFO  backtype.storm.spout.ShellSpout - ShellLog pid:6970, name:tweet-spout Empty queue exception
35638 [Thread-13-tweet-spout] INFO  backtype.storm.spout.ShellSpout - ShellLog pid:6968, name:tweet-spout Empty queue exception
35738 [Thread-29-tweet-spout] INFO  backtype.storm.spout.ShellSpout - ShellLog pid:7000, name:tweet-spout Empty queue exception
35819 [Thread-15-tweet-spout] INFO  backtype.storm.spout.ShellSpout - ShellLog pid:6970, name:tweet-spout Empty queue exception
```

Fig 1: Initial Tweet Stream Errors

Parse.py

The parse.py bolt script cleans every tweet as it is passed randomly from the spouts. The tweet is split into its constituent words, which are then cleaned of extraneous symbols and converted to lowercase. Note, a number of symbols were added to this script to ensure the only words that are eventually placed in the Postgres database are legitimate. Furthermore, the python .lower() string method was used to eliminate duplicate words, which differ only in case.

WordCount.py

The wordcount.py bolt script receives cleaned words from the Parse.py bolts. It is essential that the parse bolts send unique words to any wordcount bolt instance, otherwise it is possible that duplication of word counts could take place. For this reason, the topology dictates that that only one instance will receive any given word.

Upon receiving a word, the script increments its count and updates the database with appropriate count value if the word already exists, otherwise it initializes the word with a single count. In the event that two bolts attempt to update the same entry in the table, an IntegrityError is caught, at which point the second attempted update is dropped. Given the infrequency of this concurrent event, it was deemed that this was suitable to estimate the number or times a word appears in tweets.

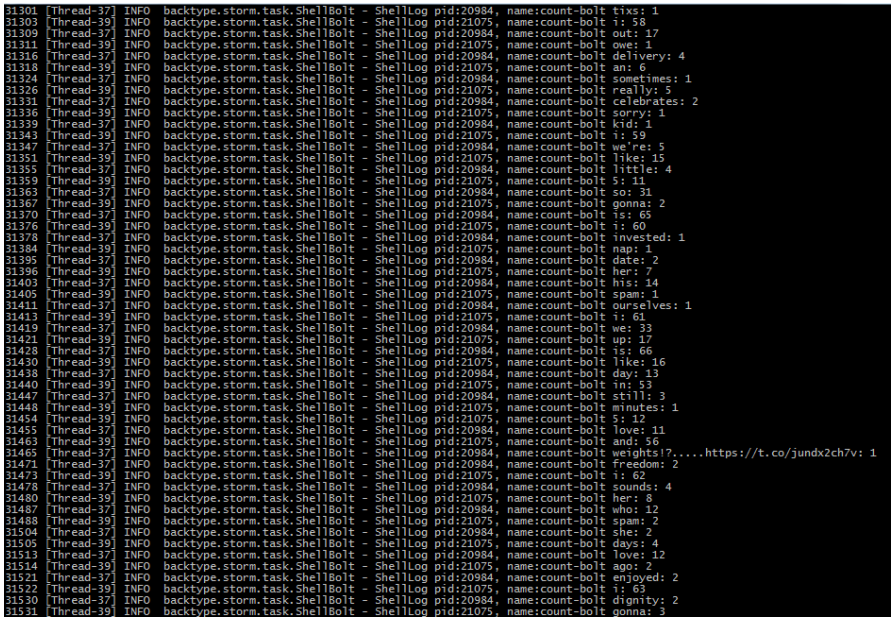


Fig 2: Execution of TweetWordCount Storm Configuration

FinalResults.py

The final results script queries the Postgres database. The user has the ability to pass a word into the script as an argument and see how many times word appeared in the stream just viewed. In the event that the word does not exist in the database, a message informs the user that no such instance was found. If no argument is passed, the script will return the first 50 words, sorted alphabetically by name.

```
[root@ip-172-31-13-92 tweetwordcount]# python finalresults.py mom
Total occurrences of "mom": 2
[root@ip-172-31-13-92 tweetwordcount]# python finalresults.py trump
Total occurrences of "trump": 23
[root@ip-172-31-13-92 tweetwordcount]# python finalresults.py obama
Total occurrences of "obama": 2
[root@ip-172-31-13-92 tweetwordcount]# python finalresults.py clinton
No instances found.
[root@ip-172-31-13-92 tweetwordcount]#
```

Fig 3: Example results from FinalResults.py

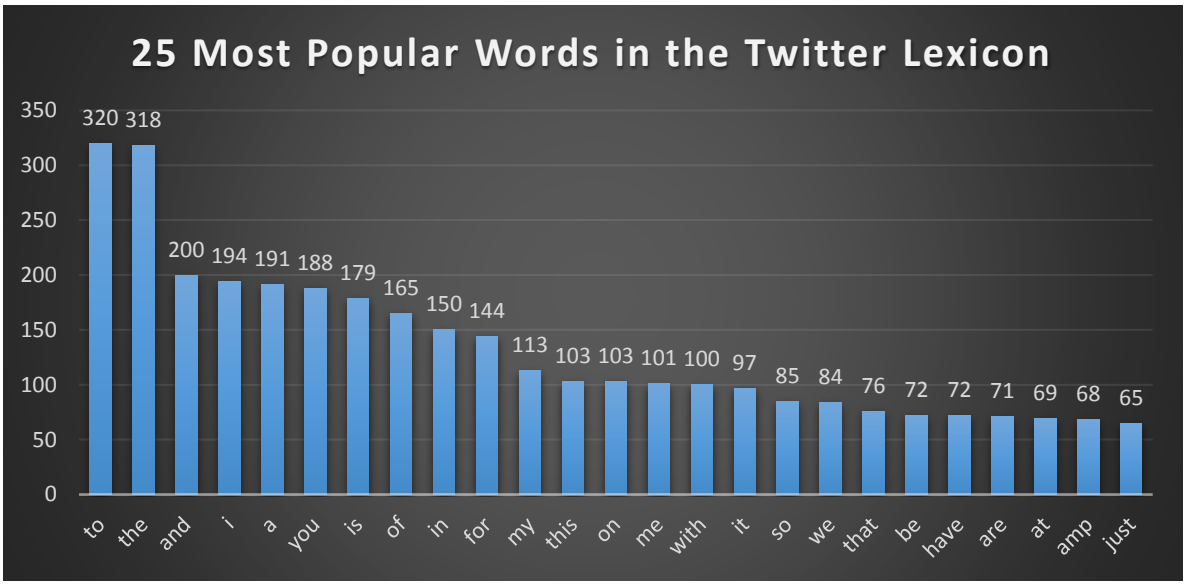
Histogram.py

The histogram script takes two number as its parameters. From these two numbers, a range is passed to a Postgres query, which returns all values falling between these two arguments. If no value is specified, the first 50 words are returned, sorted in descending order by counts. Note, ensure when using this function, no space is passed between the integer values after the comma (e.g. 5,6 NOT 5, 6).

```
[root@ip-172-31-13-92 tweetwordcount]# python histogram.py 34,57
word = now      count = 53
word = not      count = 52
word = love     count = 48
word = by       count = 45
word = will     count = 45
word = from     count = 44
word = your     count = 44
word = if       count = 43
word = don't    count = 42
word = out      count = 42
word = about    count = 41
word = us       count = 41
word = new      count = 39
word = as       count = 38
word = all      count = 37
word = it's     count = 37
word = can      count = 35
word = get      count = 35
word = like     count = 35
word = people   count = 35
word = up       count = 35
word = was      count = 35
word = who      count = 35
word = i'm      count = 34
```

Fig 3: Using the Final and Histogram Python Scripts

Histogram



Note: This histogram was made using Microsoft Excel. Output was copied directly from the console and then converted from text. A pivot chart was then used to display the histogram shown above.