Exercise 2 Apache Storm

Drew Plant

Storm Architecture -- Dec. 9, 2015

Directory Structure

```
   Because this project is a Storm architecture its main file
structure is the following:
# The current directory cloned from github.
   $PWD

# The EX2Tweetwordcount Storm project directory
   $PWD/EX2Tweetwordcount/

# The storm topologies sub-dir
   $PWD/EX2Tweetwordcount/topologies

# The storm clojure file
   $PWD/EX2Tweetwordcount/topologies/EX2Tweetwordcount.clj

# The storm source sub-dir
   $PWD/EX2Tweetwordcount/src

# The tweets.py spouts source
   $PWD/EX2Tweetwordcount/src/spouts
   $PWD/EX2Tweetwordcount/src/spouts/tweets.py

# The storm bolts sub-dir
   $PWD/EX2Tweetwordcount/src/bolts
   $PWD/EX2Tweetwordcount/src/bolts/parse.py

# ...wordcount.py bolt which collects words from
# ...parse bolt and outputs (word,count) tuples
# ...as well as writing and updating to a PostgreSQL
# ...database table
   $PWD/EX2Tweetwordcount/src/bolts/wordcount.py
```

```
 # 3 screenshots of self-describing app behavior.

   $PWD/screenshots/[].png

# Scripts finalresults.py and histogram.py
   $PWD/scripts

# Bar plot showing top 20 Tweet words collected over a 1 hour period.

   $PWD/Plot.png

# Description Readme file for running the app and serving
applications.
   $PWD/Readme.txt
```

```
<h2>Application Idea</h2>
```

```
The question which this application seeks to answer is "how can I
produce a list of word counts from tweets as they are streaming in?"

Tweets are generated using API calls to twitter using "consumer_key",
"consumer_secret", "access_token" and "access_token_secret" as
provided by twitter when signing up for a developer account.
```

Description of Architecture

```
The Tweets(Spout) is the data generator "spout" for the storm
project, using the "tweepy" python module.

The first ParseTweet(Bolt) collects each "tweet" item from the
Tweets(Spout) and breaks it up into words, emitting each word
individually to the next bolt.

The second WordCounter(Bolt) collects each word element from the
first "bolt" and increments a counter for each word, emitting the
words as a (word, count) tuple.  This bolt also logs words and counts
into a PostgreSQL database (called Tcount) and table (called
(Tweewordtcount).  WordCounter implements a Counter() count hash and
emits (word,count) tuples.  Additionally it uses the psycopg2 module
to connect with a PostgreSQL Database / Table ("Tcount" /
"Tweetwordcount") in order to update the PostgreSQL table with the
word counts.

Serving layer applications allow a user to query the database
PostgreSQL table.
   finalresults.py -- allows a user to query a word and find out its
count in the database.
   histogram.py k1 k2 -- allows a user to query the list of words
whose counts range from values k1 <= count <= k2.
```

File Dependencies

```
In general, this storm application naturally requires having a
running PostgreSQL server.  One can be started as described in the
next section "Information on Running the Application."

Also it relies on the PostgreSQL database (Tcount) and table being
setup properly within PostgreSQL by user "postgres" and password =
"", (again, see the next section), because table dropping and
creation don't appear to be supported within the bolt layer using
psycopg2 (or at least it wasn't apparent to me how drop / create
table would be handled after repeated tries.)
```

Information on Running the Application

```
As indicated in the Readme.txt file and repeated here for
convenience:
To run this example, do the following:

1. Start the postgresql server
    /data/start_postgres.sh

2. Run postgresql as user postgres:
    psql -U postgres

2. Create a database called Tcount at the "postgres=#" prompt:
   postgres=#  create database Tcount;

3. Connect to the database Tcount at the "postgres=#" prompt:
   postgres=#  \c tcount;
```

```
4. Create a table called tweetwordcount at the tcount=# prompt:
   tcount=# CREATE TABLE tweetwordcount
   tcount=# (word TEXT PRIMARY KEY NOT NULL,
   tcount=# COUNT INT NOT NULL);

5. You can exit out of psql:
   tcount=# \q

6. Run the storm architecture from within directory
$PWD/EX2Tweetwordcount
   cd EX2Tweetwordcount
   sparse run
```

```
7. In another terminal on the same machine, navigate to directory
$PWD/scripts/

8. Run the scripts:
    A. python finalresults.py you
        Will print the number of occurrences of the word 'you'
    B. python finalresults.py
        Will print out set of (word, number) tuples sorted
alphabetically by word.

    C. python histogram.py 50 75
        Will print out the words with counts between 50 and 75
inclusive.
```

```
This final section shows how to generate a bar-chat which was used to
illustrate the top 20-twitter words by count:
```

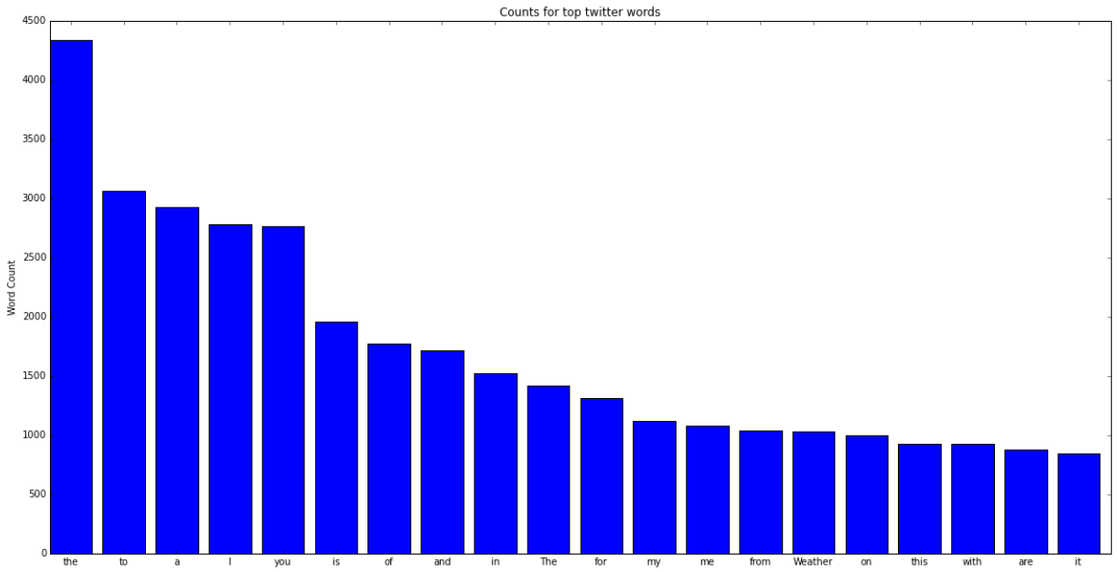
In [2]:
```
# This tells matplotlib not to try opening a new window for each plot.
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt
```

In [16]:
```
N=20
Xlocs = np.arange(N)
width = 0.8
Counts = (4338, 3061, 2928, 2781, 2763, 1962, 1772, 1713, 1520, 1414,
Words = ('the', 'to', 'a', 'I', 'you', 'is', 'of', 'and', 'in', 'The',
Counts=Counts[:N]
Words = Words[:N]

fig, ax = plt.subplots()
fig.set_size_inches(20,10)
rects = ax.bar(Xlocs, Counts, width, color='b')
ax.set_ylabel('Word Count')
ax.set_title('Counts for top twitter words')
ax.set_xticks(Xlocs+width/2)
ax.set_xticklabels(Words)

plt.show()
```



In [ ]:
```
```