The Application

The application, *extweetwordcount*, analyzes live Twitter stream and saves individual words contained in each Tweet. The application reads each stream of tweets from the Twitter streaming API, parses the tweets by word, counts the occurrences of each word, and saves the word and count into a Postgres database named *tcount*. The table, *tweetwordcount*, in the *tcount* database will hold all words and can be used to run various analysis on the data collected from the Tweets. These analyses consist of returning all words and counts, querying the table for the count of a specific term or returning all words that have a count between two values.

Expanding the scope of this application could provide an even more real-time understanding social data and social insights. Twitter allows users to write and share short thoughts to their network. When a large event is occurring, it is common to see an uptick of tweets about that specific event. The ability to notice the increase in related tweets allows an advertiser, for example, to understand the information in real-time and react appropriately to their target audience faster and more pertinent than their competition. Additionally, storing this information over time can provide insight into social trends or cyclical trends over longer periods of time that can lead to a better understanding of markets and how to be read next time that cycle will come around.

Description of Architecture

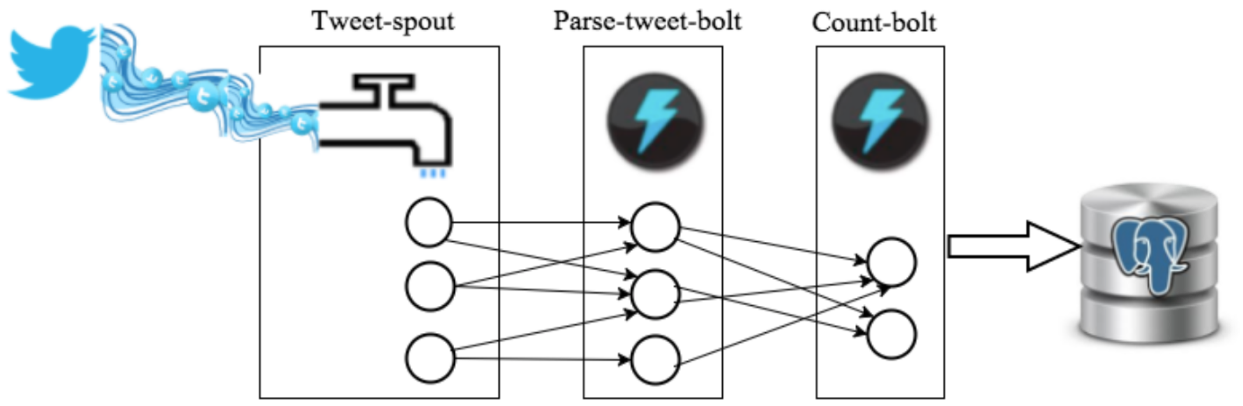

Figure 1: Application Topology

The processing application *extweetwordcount* application has the workflow shown in figure1.

Application Processing Layer

- The *tweet-spout* connects and accepts a stream of tweets from the Twitter API.
- The *tweet-spout:*
    - Connects to the twitter API using credentials
    - Receives incoming tweets from twitter API stream
    - Has 3 parallel workers to process the tweets.

- Then the *tweet-spout* passes the tweet to the *parse-tweet-bolt*
- The *parse-tweet-bolt*:
    - Parses each incoming tweet into singular words.
    - Filters out hashtags (#), mentions (@), re-tweets(RT), urls and punctuation from the words that have been parsed.
    - Has 3 parallel workers to process incoming tweets
- Then *parse-tweet-bolt* sends the parsed words to the *count-word-bolt*
- The *count-word-bolt:*
    - Upon initialization
        - Creates the *tcount* database in postgres
        - Creates the *tweetwordcount* table in the *tcount* database to hold words being processed
    - Processes each word incoming from *parse-tweet-bolt* by checking if the word exists in the *tweetwordcount* table
        - If the word does exist in the table, the count is increased by one and committed back to the table
        - If the word does not exists in the table (the word is new), the word is committed to the table with a count of 1

Serving layer

- The serving layer consists of two python scripts *finalresults.py* and *histograms.py*
- *finalresults.py* - Queries the *tweetwordcout* in *tcount* postgres database for the words and their counts.
    - When run without a passed argument, returns all the words and their counts in the *tweetwordcout* table that had appeared in the twitter stream.
    - When run with a word as a passed argument, returns the count of the word requested
- *histograms.py* - Queries the *tweetwordcout* in *tcount* postgres database for the words and their counts.
    - Returns the returns all the words with a total number of occurrences greater than or equal to passed argument *k1* and less than or equal to passed argument *k2*.

Directory and File Structure

Extweetwordcount application is contained within the directory w205_2017_summer/exercise_2/
README.txt contains step-by-step instructions for running the application.

Application *extweetwordcount* File Structure

- **extweetwordcount** (folder) contains the full application
    - To run application navigate to extweetwordcount and run “sparse run”
    - **Screenshots**: contains screenshots of application running
    - **topologies/**
        - **tweetwordcount.clj** - topology for the application
    - **src/**
        - **spouts/**
            - **tweets.py** - spout for connecting collecting and dispensing tweets from twitter API

        - **bolts/**
            - **Parse.py -** bolt responsible for parsing incoming tweets
            - **Wordcount.py -** bolt responsible for counting parsed words.
    - **Finalresults.py -** a serving script for final words and counts in *tcount* postgres database after *extweetwordcount* application has been run.
    - **Histogram.py -** - serving script for final words and counts in *tcount* postgres database after *extweetwordcount* application has been run.
    - **create_barChart.py -** creates a bar chart for the top 20 words counted in the *tcount* database, *tweetwordcount* table.
    - Plot.png - bar chart of top 20 words counted in database

File Dependencies for use of Application

All File dependencies:

- Apache Hadoop
- Apache Storm
- Apache Spark
- Amazon EC2 instance
- Python, Pyspark
    - Numpy, Matplotlib, Counter from collections
- Twitter API (Twitter account credentials)
- Tweepy
- Streamparse
- Postgres
- Psycopg2

Dependencies by File:
create_barCart.py

- Python, Numpy, Matplotlib, Psycopg2

Finalresults.py

- Psycopg2

Histogram.py

- Psycopg2

Parse.py

- Counter from collections, Psycopg2

Wordcount.py

- Counter from collections, psycopg2

Other Information

Twitter credentials consist of 4 pieces

- A consumer key that identifies your application.
- A consumer secret that acts as a password for your application.
- An access token that identifies your authorized access.
- An access token secret that acts as a password for that authorized access.

You can access the database and table through Postgres. This can be useful for testing and doing direct queries on data outside of the serving layer information.