Architecture Overview

Exercise 2
Manuel Moreno
w205-1

Application Description

The purpose of this project is to determine the occurrence of words in the Twitter stream during the execution time period. Because of the nature of Twitter, rather than relying on batch processing, the application sets up a streaming connection where the Twitter service actively pushes new data to the application. As such, the application requires the following modules:

1. A connection module that sets up the connection and receives data from the server.
2. A parsing module that cleans the data received and pushes out individual words to the counter.
3. A counting module that receives words and keeps a running total of them. This module also connects to the PostgreSQL database and updates the totals.

Database Connection and Schema

The database connection is established through the psycopg2 package for python 2.7. The connection string is as follows:

psycopg2.connect(database="tcount", user="w205", password="", host="localhost", port="5432")

The database, user accounts and connections were created using the following method (courtesy of James Peng):

```
psql --username=postgres
CREATE USER w205 WITH PASSWORD 'postgres';
CREATE DATABASE Tcount;
ALTER DATABASE Tcount OWNER TO w205;
GRANT ALL ON DATABASE Tcount TO w205;
```

The data table was created using the following:

```
psql --host=localhost --username=w205 --password --dbname=tcount
CREATE TABLE Tweetwordcount (word TEXT PRIMARY KEY NOT NULL, count INT NOT
NULL)
```

Application Design

Due to the nature of this application, StreamParse 2.0.1 was used to create the processes. The following figure illustrates the Application Topology.

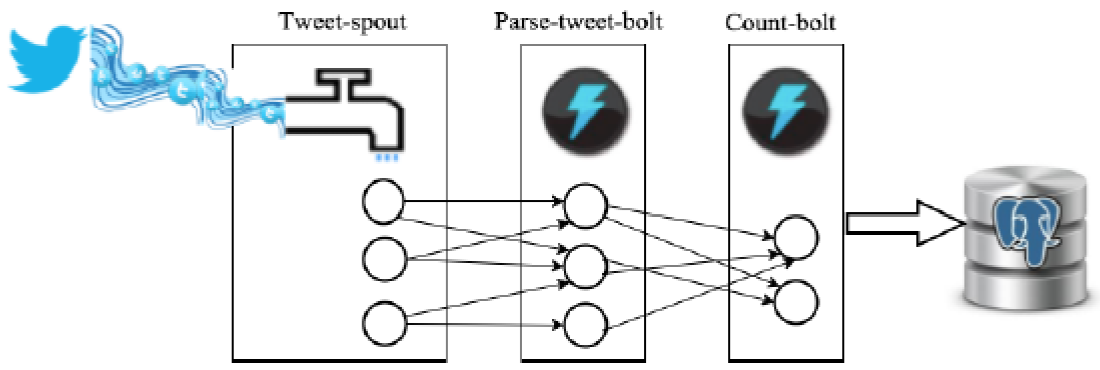

Figure 1: Application Topology

Tweet-spout:

The spout was configured to handle the authentication to Twitter and receipt of the data. Metadata is stripped from the tweet and then passed to the parser bolt.

Parse-tweet-bolt:

The parser determines the language of the tweet and, if it is English, it will clean up invalid characters, hastags and user references to until only the content words remain. These will then be passed to the counter bolt.

Count-bolt:

The counter keeps track of the occurrences of each word and will:

1. Create a new entry when a new word is encountered
2. Update the count for repeated words

The application will listen until interrupted.

Data-serving scripts

In addition to the main application, data serving scripts were created to leverage the data gathered by this application. These include:

finalresults.py

When executed with arguments, this script will return the number of occurrences for the first argument provided. When executed with no arguments, this script will return the full list of words, on for each line.

histogram.py

This script will take a single argument that consists of two integers separated by a comma (e.g. 4,6). After validating the argument, this script will return items that occurred a number of times that falls between the two integers provided (inclusive) or indicate that no items were found.

barplot.py

This script will simply create a bar chart of the 20 most common words. This will be saved to testplot.png.

Repository Structure

The repository consist of the following files and folders:

Exercise2 – StreamParse folder. This folder contains all the files required to execute the StreamParse application. This name was chosen due to errors with other names (see error screenshots)

Screenshots – This folder contains the error and application demonstration screenshots.

Architecture Overview.pdf – This document. Provides an overview of the project.

barplot.py – While not part of the deliverable, this script was used to generate the Plot.png file.

finalresults.py – Script that returns the final count for either one or all the words encountered.

histogram.py – Script that returns words that fall within a specified threshold.

readme.txt – Provides instruction on how to execute the application and its supporting scripts.

Plot.png – Bar chart of the top 20 words encountered.