December 11, 2015

Exercise 2

w205 Data Storage and Retrieval

Alejandro Rojas

Context

This application was developed within the social of context of the Venezuelan parliamentary elections where opposition parties managed to get 2/3 supramajority after an election held on December 6, 2015.

Purpose

The application taps twitter messages that address Venezuelan political issues to measure what are the topics that Venezuelans are discussing. Each tweet is broken down into usable words that are then counted live using streaming capabilities on Storm. Counted data is stored in Postgres database and results are displayed on a Tableau interface.

Tracking the popularity of words that are part of tweets being sent provides a useful way to summarize public attention to political figures and topics as well as overall sentiment at any point in time when the application is turn on.

Data Architecture

The application was developed using a topology that incudes three Storm components that populate a Postgres database that serves a summary chart table on Tableau.

To tap tweets directly from the Twitter stream, our application connects to the Twitter API to request permission to access all tweets that address terms that have a political connotation.

This data is collected using a Tweet-Spout component that uses the Tweepy library to access Twitter's API to track Spanish tweets that contain certain terms that are relevant for Venezuelan politics.[1] Once these tweets are collected, they are sent to a Parser-Bolt that breaks each tweet into valid words, allowing only ascii characters and filtering out some characters like @, RT, etc.[2] After this bolt is executed, a Counter-Bolt uses the parsed data to then populate a Postgres database named tcount with a single table called Tweetwordcount that simply

---

[1] Some terms included are ["Capriles", "Leopoldo", "Maduro", "Cabello", "Chavistas", "chavista", "oposicion", "opositor", "cambio", "capriles", "asamblea"]

[2] Some characters that were filtered out include: \ " ? > < , ' . : ;) RT @ # http

tracks words and the number of times that word has been mentioned on the tweets collected.

Key Files and Data Directory

EX2Tweetwordcount/

Twittercredentials.py: file that contains authentication information required to connect to Twitter's API.

Finalresults.py: script that returns words and their corresponding counts. Adding a term after running the script on the CLI will also give you the specific amount of occurrences of that specific term.

dropaddtable.py: script to start counter from scratch.

EX2Tweetwordcount/src/spouts

tweets.py: file that collects tweets through Twitter's API.

EX2Tweetwordcount/src/bolts

parse.py: file that breaks each tweets into valid words.

wordcount.py: file that stores words and its corresponding counts in a Postgres database.

EX2Tweetwordcount/src/topologies

tweetwordcount.clj: file that describes how spouts and bolts work together.

Postgres:

Database: tcount
Table: Tweetwordcount

Instructions to run application

1. Connect to instance
2. Start counter from scratch. CLI: $python dropaddtable.py
3. Turn on the data hose by typing "sparse run" on the EX@Tweetwordcount directory where all files reside. CLI: $sparse run
4. To find out the count associated to a specific term run the following line on the CLI: $python finalresults.py ,<search_term>
5. To list all terms and their counts at any moment run the following line on the CLI: $python finalresults.py
6. Use Tableau to connect to the Postgres database to generate visualizations making sure that port 5432.