W205 Exercise 2 ExTweetWordCount Architecture

David Huber

Application Idea:

Streaming from tweets from Twitter is a simple and effective way to demonstrate the basic capabilities of Apache Storm. This application allows the user to choose which words to use to filter a stream of tweets. Each word contained in each of those tweets is counted. That word count data is stored in a database and the provided tools can be used to investigate the results. Before each run of the application, any word counts from prior runs are removed from the database so that results are strictly live data from the most recent run only.

Description of Architecture:

The *ExTweetWordCount* application uses the Apache Storm real time streaming computation framework. This framework consists of a data stream called a “spout” and data transformation and processing modules called “bolts.”

The spout module uses Twitter’s Tweepy API to connect to Twitter and stream tweets into the application and emit tweets one at a time to be processed. There are two layers of bolts: the parsing bolt and the counting bolt. The parsing bolt receives the tweets output by the spout module, breaks each tweet up into a list of words, filters out words and characters not useful to us, and then emits those words one at a time. When the words are emitted, they are received one at a time by the counting bolt. The counting bolt increments the word’s count in the database and the local count. The local count is for demonstration purposes and has no persistent usefulness. If the word hasn’t been encountered yet, it is inserted in the database and local counter with a count of one. As instructed, 3 tweet spouts, 3 parse bolts, and 2 word count bolts were used.

After program execution, the results can be seen in two ways. The *finalresults* script shows either the number of times an input word was encountered, or the list of counts of all words. The *histogram* script takes an upper and lower bound and outputs the words with counts between those bounds.

Directory and File Structure:

Directory/File	Location	Description
exercise_2	---	Top level directory of repository
extweetwordcount	exercise_2/	Directory containing all storm and streamparse components
screenshots	exercise_2/	Contains the end-to-end screenshots
runDemo.sh	exercise_2/	Script to run application
config.ini	exercise_2/	Configuration file to store credentials
finalresults.py	exercise_2/	Results script
histogram.py	exercise_2/	Histogram of results
myDBObj.py	exercise_2/	Helper class for database access
Architecture.pdf	exercise_2/	Documentation
Readme.txt	exercise_2/	Quickstart - how to run the application
tweets.py	exercise_2/extweetwordcount/src/spouts/	tweet-spout
parse.py	exercise_2/extweetwordcount/src/bolts/	parse-tweet-bolt
wordcount.py	exercise_2/extweetwordcount/src/bolts/	count-bolt
extweetwordcount.clj	exercise_2/extweetwordcount/topologies/	Topology for the application
ex2Files	~	Temp folder only present during application runs.

Other files and folders in the structure are autogenerated.

Description of runDemo.sh

Creates temp folder, ex2Files, in home directory(~) of user
Copies .py and .ini files to temp folder
Parses command line and updates the copied config.ini file if the word track option is present

Runs the topology
After program exit (Ctrl+c)
Removes temp folder

Running the Application:

To run the application, edit the config.ini file and fill in Twitter Application credentials. Make sure Postgres is running. Run the *runDemo.sh* script to see the stream in action. The script has one optional parameter which is a comma separated list of words to track. That is, only tweets containing words in this list will be tracked. If this option is not supplied, the default value will be read from the *config.ini* file, and as installed, the list contains “a,and,an,the,u,you”, however it can be edited manually if so desired. Note that it is one parameter, and while spaces are not required, if they appear, “quotes” must be used on the command line. The application runs until terminated with “Ctrl+c”. Keep in mind that if left running, it will periodically stop because of Twitter’s rate limiting policy.

Extensions:

A useful extension to the current functionality would be more application parameters to control the filter and track more varied cases, like words associated with users via the ‘@’ sign. It would also be useful to track tweets that contain pairs or groupings of words. Smoother program flow could be attained by adding some rate limiting or throttling logic, but this would need further investigation. Finally, the ability to save prior runs of the application in the database for comparison purposes would be essential for any real world application.

The *finalresults* script might benefit from more output modes such as a way to control the sorting behavior or target interesting groupings. The *histogram* script could be extended to output the results into a csv file that could make the process of visualizing the results easier. Also, additional output controls like “Top 10” or “Top 10%” would make the script more useful.