Application idea

The application is developed for analysis tweet data at word level. It allows user to collect live streaming tweets in real time and aggregate the tweet in word counts. Users can use the words level information for text analysis and create visual charts.

Directory and File structure

There are mainly two parts in the application. Data collection part and data analysis part.

All data collection part is placed in the extweetwordcount folder, which used the Apache Storm and twitter stream API for data collection.

The data analysis part is place in the tweet analysis folder. All analysis files are placed under this folder. The main files are:

- finalresults.py, when running "python finalresults.py <word>" with a single word as argument, it will the total number count of word in the stream. If running finalresults.py without an argument returns all the words in the stream, and their total count of occurrences, sorted alphabetically, one word per line

- histogram.py. The script gets two integers k1,k2 and returns all the words with a total number of occurrences greater than or equal to k1, and less than or equal to k2

- plot.png. The picture that shows the top 20 words in the twitter stream.

Name of the program	Location	Description
tweets.py	/exer2/extweetwordcount/src/spouts/	Tweet-spouts
parse.py	/exer2/extweetwordcount/src/bolts/	parse-tweet-bolt
wordcount.py	/exer2/extweetwordcount/src/bolts/	count-bolt
tweetwordcount.clj	/exer2/extweetwordcount/topologies/	Topology
finalresults.py	/exer2/	Run count for words
Histogram.py	/exer2/	Run counts within range
Plot.png	/exer2/	Bar plot for top 20 words
Create_table.py	/exer2/	Code to create database and table
Readme.txt	/exer2/	Step by step instruction of the application
Architecture.pdf	/exer2/	How the application is implemented
screenshot-twitterStream.png	/exer2/screenshot	Screenshot of streaming result for "trump"
screenshot-extractresult.PNG	/exer2/screenshot	Screenshot of tweetwordcount table
screenshot-finalresult-ouput.PNG	/exer2/screenshot	Screenshot of the finalresults.py output

screenshot-histogram-output.png	/exer2/screenshot	Screentshot of the histogram.py output
screenshot-stormComponents.png	/exer2/screenshot	Screenshot of the storm running result

Description of the architecture,

The application use Apache Storm for data collection and data preprocess. There are one spout and two blots in the architecture.

The spout provides the data feed, which use the tweep library. It reads live stream of tweets from twitter. The spout passes tweets to the following blots.

The first bolt in the Storm architecture is the parse-tweet-bolt. It will filter out all hashtags, user mentions, retweets tags, urls and leading/trailing functions. The first blot will accept raw tweets from the spout as input and pass valid words in the tweets to the second blots.

The second bolt in the Storm architecture is the count-bolt, which counts the number of each word in the received tuples, and updates the counts associated with each word in the tweetwordcount table inside the tcount Postgres database.

The data is stored in the postgres table “tweetwordcount” (which belong to the tcount database). The table has two fields: word and count. The word field is a unique primary key for table and count is just the corresponding count for the word. User can freely interact with the table with postgre sql command.

If users are free to start or stop the collection at any time, the counts will be automatically aggregated in the tweetwordcount table.

A few basic analyses have been done in the finalresults.py and histogram.py, which use psycopg2 to interact with the tcount postgres database.

File dependencies

create_table.py → wordcount.py

Please make sure that you run the create_table.py before running Apache storm. The database and table needs to be created before Apache storm writes in any data.

tweets.py → wordcount.py

Please make sure that the consumer key, consumer secret, access key and access secret. The api won’t be able to collect tweets.

wordcount.py → finalresults.py/histogram.py

The Storm topology needs to be done before running the finalresults.py and histogram.py. Otherwise, the script will only return empty result.