W205

Exercise 2

Clay Miller

Overview

In this exercise we are tasked with streaming data from twitter, parsing and counting the words from the tweets and storing the word counts in a postgres database. This exercise expands on many foundations that we learned from lab 6 where we "streamed" some hard coded tweets with the same Apache Storm architecture.

Before we turn on our twitter stream, however, we need to get a database set up so that we can store the information we will extract from the data. This is done via setup_pg_database.py. This simple script creates a database called tcount and a table in that database called tweetwordcount. The columns in tweetwordcount are word and count (not surprisingly). Now that we have this database and table created, we can write to it as we run storm.

We used Apache Storm as the architecture to bring in the data and process it. Just as in Lab 6, the back bone of this architecture is the topology comprised of spouts and bolts. In this case we have just the one spout, which is where the twitter data come in from and they get passed to the subsequent bolts. The first bolt in the topology simply parses the tweets into discrete words taking care to filter out special characters such as "#" and "@". In addition to these filters, I also added a few extra checks for better filtering. The first thing it does is to convert everything to lower case to avoid duplication of words like "Rabbit" and "rabbit". Secondly, I used the nltk python package to get a list of common stopwords (words that are very common) so that I can also filter those out because they will not be very interesting in this analysis. The words that do not get filtered then get passed to the next bolt that takes each word and adds it to a dictionary of all the words we have seen for all the tweets. It will add 1 to the current count for that word. Once the dictionary has been updated, it also updates the entry for that word in the postgres database.

Details

The directory containing the relevant files is called extweetwordcount. As mentioned above, the python script to create the postgres database is called setup_pg_database.py. The storm programs reside in the topologies and src directories. The topology directory contains the file tweetwordcount.clj which describes the topology and the src directory contains the bolt and spout subdirectories. The spouts directory contains tweets.py which is responsible for getting the twitter data and starting the flow of data. In order to make this work with live twitter data we need to use the twitter api and our twitter credentials to stream the data. It is best practice to not put one's api keys directly on EC2 or to submit this via github. Instead, I have stored these credentials as environmental variables and called them accordingly in the script. As an example, my twitter consumer key gets called by os.environ["TWITTER_CONSUMER_KEY"].

The bolts directory contains parse.py and wordcount.py which are responsible for parsing and counting the words from the data.

On top of the actual data we parse and store, we were also asked to create some scripts which query the postgres db for different metrics. These files are called finalresults.py and histogram.py and are found in base directory.

Setup

Running this program should be fairly straightforward, but to use the nltk package, you will first have to install nltk and numpy. Then there is an extra step which was rather confusing, but I figured it out so you don't have to. In python, import the aforementioned packages and then type nltk.download("stopwords"). This will download all the stopwords used in filtering. If this step is not run, storm will not run and throw an error.

Running

To run, cd to the extweetwordcount directory and type sparse run and then you should have to hit enter one more time. Then watch the streaming happen! This assumes you have a working version of stream parse and tweepy and of course twitter api credentials.

Histogram

Some notes on the top 20 words. I ran the twitter stream for 45 minutes and gathered a fair amount of data. While it did a good job to filter out common words, many snuck through. The most common word by far is rt, or "retweet" which should have been checked for. After that, we have many more "common" words. It is interesting that the word "white" made it into the top 20. Whether this had anything to do with what happened this weekend at the Klan rally in Virginia is unknown. Outside of that, the next most interesting word was "fuck" which had a count of 117.