CS 4371.501 Introduction to Big Data Management and Analytics
(Fall 2020)

Project: Spark Streaming Background

This project deals with real-time streaming data arriving from twitter streams.

1. Spark Streaming

Spark Streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams. Data can be ingested from many sources like Kafka, Flume, Kinesis, or TCP sockets, and can be processed using complex algorithms expressed with high-level functions like *map*, *reduce*, *join* and *window*.

Finally, processed data can be pushed out to filesystems, databases, and live dashboards. In fact, you can apply Spark's **machine learning** and **graph processing** algorithms on data streams.

Internally, it works as follows. Spark Streaming receives live input data streams and divides the data into batches, which are then processed by the Spark engine to generate the final stream of results in batches.

Spark Streaming provides a high-level abstraction called discretized stream or DStream, which represents a continuous stream of data. DStreams can be created either from input data streams from sources such as Kafka, Flume, and Kinesis, or by applying high-level operations on other DStreams. Internally, a DStream is represented as a sequence of RDDs.

Please checkout this website for details.

2. Source

In this assignment, we will use twitter as our stream source. To access twitter stream, please see the following instructions:

2.1. First create your own account at https://apps.twitter.com/

2.2. Click on *Create an App* to create your own Twitter app.

2.3. Fill out the form and submit

2.4. Once you have created your twitter app, you will observe a form similar to this:

2.5. Click on **Keys and tokens**, generate your keys (you will need them to add in stream.py code)

   Consumer Key (API key)
        Consumer Secret (API secret key)
        Access Token
        Access Token Secret

2.6. Crawling and Create a Stream

        Crawling – refer to *stream.py* for details
        Create a Stream – refer to *spark.py* for details

This setup code related to stream.py and spark.py are available on elearning website.

Once you run these codes (run stream.py first and then run spark.py), you are going to observe such a window
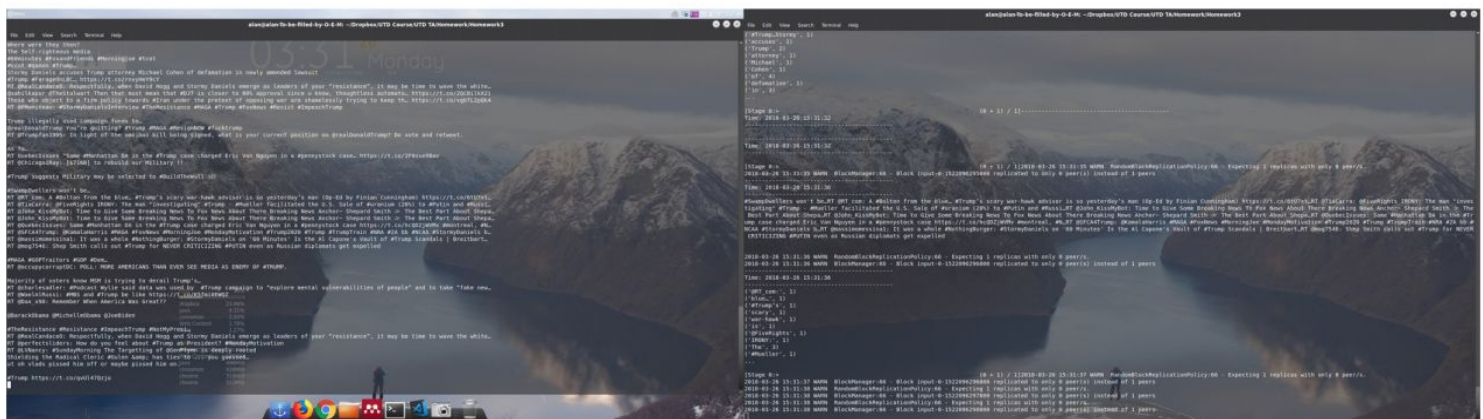
3. Task

You are asked to implement a framework using Apache Spark Streaming to perform sentiment analysis of particular hashtags in twitter data in real time. For example, suppose we want to do the sentiment analysis for all the tweets containing the hashtag #guncontrolnow and show their (e.g.,positive, neutral, negative) so we can later show the statistics.

The first step for this task is to get the tweets via scrapper (stream.py). Next, you will apply any pre-trained model for sentiment analysis over the tweet messages. Note that gathering data related to location (where the tweet was generated) may also be useful for analyzing later (Don't worry, stream.py already gets the data for you). Finally, you will visualize your findings using either Spark SQL or ElasticSearch/Kibana (preferable!).

4. Putting all together

The flow for the entire project will be:

Scrapper ---> Sentiment Analyzer ---> Visualizing (Spark SQL or ElasticSearch/Kibana)

As aforementioned, the skeleton of the framework is provided in python on eLearning. The parts you will need to change are marked as comments. Always remember that this is YOUR project, and you are free to change 100% of the codes, including the provided skeleton.

4.1. Scrapper: We provide a sample scrapper (stream.py). However, you need to extend the code to support the following functionality described below.

The scrapper collects tweets and pre-process them for analytics. It is a standalone program written in Python and should perform the following:

(i) Collect tweets in real-time with particular hashtag. For example, we will collect all tweets with #guncontrolnow.

(ii) After getting tweets, we filter them by removing emoji symbols and special characters and discard any noisy tweet that do not belong to #guncontrolnow. Note that the returned tweet contains both the metadata (e.g., location) and text contents. You will have to keep at least the text content and the location metadata.

(iii) After filtering, you need to convert the location metadata of each tweet back to its geolocation info by calling either google geo API or geopy (or any other you prefer) and send the text and geolocation info to spark streaming. Note that, this step can also be performed in spark.py side.

(iv) Your scrapper program should run infinitely and should take hashtags as input parameters while running.

4.2. Sentiment Analyzer: Sentiment Analyzer determines whether a piece of tweet is positive, neutral or negative. For example,

"President Donald Trump approaches his first big test this week from a position of unusual weakness." - has positive sentiment.

"Trump has the lowest standing in public opinion of any new president in modern history." - has neutral sentiment.

"Trump has displayed little interest in the policy itself, casting it as a thankless chore to be done before getting to tax-cut legislation he values more." - has negative sentiment.

You can use any third-party sentiment analyzer like Stanford CoreNLP, NLTK or any other software or pre-trained model for sentiment analysis.

In summary, for each hashtag, you perform sentiment analysis using sentiment analysis tools discussed above and output sentiment and geolocation of each tweet to some external bases (either save in a json file or send them to elasticsearch index for visualization on Kibana).

4.3. Visualizer: To visualize the real-time collected data you have two options:

(a) Save the data in an external json file and later show summarized results in collected data through Spark SQL.
(b) (optional*) Show the results in dashboards using the visualization tool Kibana. For that, you will need to:
  (i)   Install ElasticSearch and Kibana.
  (ii)   Create an index for visualization on elasticsearch and also in Kibana.
  (iii)   Create visualization objects to show the overall analysis over tweets.
  (iv)   Create a dashboard based on visualization objects.
  (v)   (optional*) Create heat-maps, maps or any other geo-based graph to show the geolocation distribution of positive and negative tweets. More specifically, first heat map would show the geolocation distribution of all tweets, regardless of sentiment related to the hashtag analyzed. Second and Third heat maps would show geolocation distributions of positive tweets and negative tweets, respectively. When you send data from spark to ElasticSearch, you can add a timestamp and then in the dashboard, you set the refresh time to 2 min as an example. This way, your graphs will keep changing as you receive data in real-time.

Example of a simple Dashboard in Kibana

Example of heat-maps in a dashboard in Kibana:
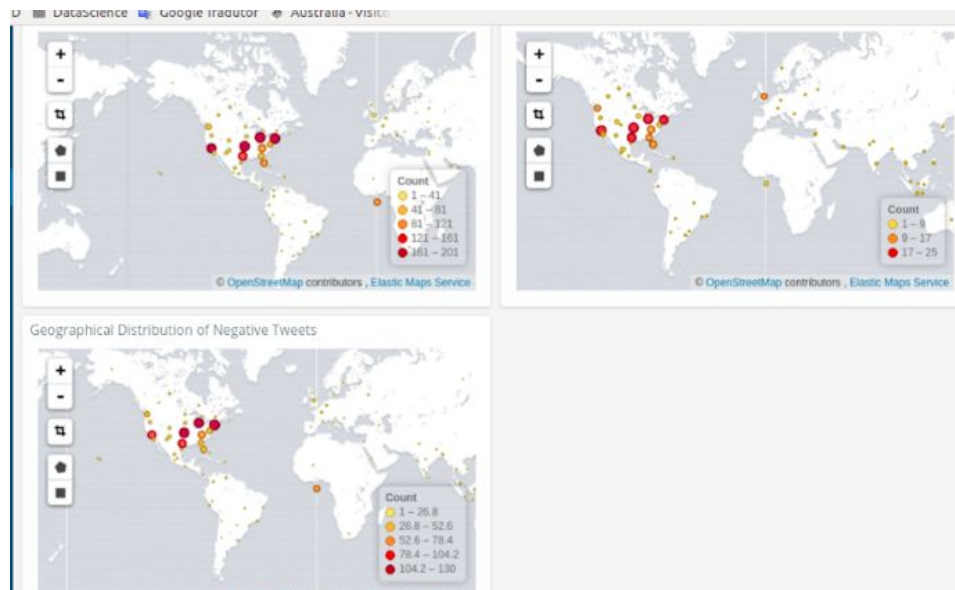
Grading Criteria:

- Completing the code for cleaning the tweets: 20 points
- Applying Sentiment Analysis: 30 points
- Applying geo tool to retrieve the correct location of the tweet: 25 points
- Visualizing/Analyzing real-time collected data:
    - Using Spark SQL: 25 points
    - (optional) Using dashboard in Kibana: 25 + 20 (extra points)
    - (optional) Using heat-maps in a dashboard in Kibana 25 + 30 (extra points)

Notes:

(1) You can work in groups of at most 3 students. However, all students will need to be present in the project demonstration.

(2) In the case you choose to work with Spark SQL for the Visualization step, I may ask you to run any Spark SQL queries.

(3) Questions during the demonstration will be directed to the students and not to the group. Therefore, all team members should work hard on the tasks.

References for the Project:

For getting Geolocation of the tweet:

https://pypi.org/project/python-gmaps/   # For GoogleMaps
https://pypi.org/project/geopy/   # For Geopy

For Sentiment Analysis:

https://medium.com/@b.terryjack/nlp-pre-trained-sentiment-analysis-1eb52a9d742c

For downloading and installing Elasticsearch:

https://www.elastic.co/downloads/elasticsearch

Elasticsearch for beginners:

https://medium.com/naukri-engineering/elasticsearch-tutorial-for-beginners-using-python-b9cb48edcedc

Official Low-level client for ElasticSearch on Python:

https://pypi.org/project/elasticsearch/

For downloading and installing Kibana :

https://www.elastic.co/downloads/kibana

For creating indexes inside Kibana:

https://www.elastic.co/guide/en/kibana/current/tutorial-define-index.html

For GeoShape datatype:

https://www.elastic.co/guide/en/elasticsearch/reference/current/geo-shape.html