Overview:

The goal of MP2 is to complete two tasks using the Apache Kafka and Spark to complete a system that can take the output of Twitter Stream API, prepare the data for processing using Kafka, send that data to Spark for sentiment analysis, and finally persist the final data to execute queries for analysis.

Task 1: Data Input using Apache Kafka

Objectives:

1. Become familiar with Kafka by setting it up on your local machine.
2. Explore Kafka topics in detail, to manipulate streaming data.
3. Understand why you might want to use Kafka in real life.

Input:

The input to this task is the provided "tweets.txt" file. Each line in the file is a JSON object, obtained using Twitter’s Stream API (<https://dev.twitter.com/streaming/public>). Since the focus is on learning different tools, we provide the data in a text file to “simulate” tweet streaming instead of requiring the usage of Twitter API.

In the raw\_data folder, the file "tweets.txt" has 9974 lines each with 1 JSON object. 9 of those objects are of the form:

* {"limit": {"track": 8, "timestamp\_ms": "1489874598078"}}

The remaining 9974 - 9 = 9965 JSON objects each represent a "Tweet" which has a lot of information a subset of which is presented below:

* {"created\_at": "Sat Mar 18 20:43:24 +0000 2017", "text": "RT @nielsberglund: [Blog]:\"Microsoft SQL Server R Services - Internals I\" https://t.co/h5HBbhFOBB\n#datascience #sqlserver #RStats @tednewa…", "user": {"screen\_name": "TheRockstarDBA"}}

The file “download\_tweets.py” is provided to let you know how the data were collected. You are not required to understand or run it.

In this task, you are required to simulate the Twitter API by loading the tweets onto a Kafka topic, process the tweets and write it to a second topic. This two-step process is intended to provide you the experience of both reading and writing Kafka topics. The instructions to setup Kafka are provided in "*kafka\_setup\_instructions.txt*".

Subtask 1:

1. Read each line of "*tweets.txt*".
2. Write each line to a Kafka topic called "*mock\_twitter\_stream*".

Subtask 2:

1. Read each message from the Kafka topic called "*mock\_twitter\_stream*".
2. If the message is one of the 9 non-Tweet messages described above then ignore the message.
3. Otherwise write the message to a Kafka topic called "*spark\_input*"

Requirements:

You may use any language and Kafka API that you want, so long as your solution can be run by using a single shell script "run\_kafka\_task.sh" that you provide.

Running that script should result in:

* All 9974 lines of "tweets.txt" to be written to "mock\_twitter\_stream".
* The 9965 Tweets to be written to "spark\_input".

Verifying correctness:

A file "kafka\_tail.py" has been provided which reads messages from Kafka. You can verify the content of the messages being written to a topic by running the following commands before executing "run\_kafka\_task.sh"

* $ ./setup\_venv.sh
* $ source venv/bin/activate
* $ python kafka\_tail.py -t mock\_twitter\_stream -n 0

This will start reading the mock\_twitter\_stream topic from its most recent offset and only print messages (along with their count), written since it was started. Thus, you can verify that the content and count of the messages are being sent to each topic.

Suggestions:

* You can use the Kafka-python API used in the provided "kafka\_tail.py": <http://kafka-python.readthedocs.io/en/master/>
* If you go that route, you should be able to complete the task without manually setting partitions or seeking to offsets unlike "kafka\_tail.py" although that file might be a good starting point.
* You may need to use the "*flush()*" method on producers to ensure all messages are written from internal buffers.
* You can use kafka-manager to create topics and view other information about your Kafka instance.

Task 2: Data analysis using Apache Spark

Objectives:

1. Become familiar with Spark setup.
2. Learn the details of how to consume streaming data (Spark streaming API).
3. Perform sentiment analysis of real-time streaming data using Spark MLLib.
4. Persist streaming data + spark querying.

Input:

The input to this task is the *spark\_input* Kafka topic created in Task 1, that provides a stream of tweets along with associated meta-data. In this task, you are required to read the tweets using Spark Streaming API, perform sentiment analysis and write out the results. Then, you will explore querying on Spark. We split the process into two steps so that you get experience reading/writing/processing live data, which is mostly a technical concept. On the other hand, the second step provides you the experience of querying on spark, to contrast with MongoDB, SQL, etc. You will need to install Spark framework for your desired language.

Subtask 1:

1. Open a connection to the existing Kafka and *spark\_input* topic using Spark.
2. Use Spark Streaming API to listen for messages (only once this is running should you run Kafka process to start sending messages).
3. Use Spark MLLib to perform “sentiment” analysis of each tweet and add this new information to each record. Write the messages (along with sentiments) to store on disk. Here, you are free to explore any method for performing sentiment analysis.

Subtask 2:

Answer following Queries:

1. Find out the latest date of each tweet (i.e., find out when it was last retweeted, if this tweet is original tweet, then return its created date).
2. For each tweet, find out the retweet count and favorite count under its original tweet.
3. Find out the tweet with:
   1. most retweets
   2. highest favorite count (hint: you may want to use map reduce functions to calculate total counts of favorite counts under this tweet and its retweets)

Requirements:

You may use any language that supports the Spark API that you want, provided it supports Spark Streaming and Kafka existing libraries. We do expect you to leverage these existing libraries and focus on the streaming, analysis and writing parts only. Export the query results into a file for each query with name spark\_query\_<query\_number>.txt.

Verifying correctness:

Once the script is up and running and you execute your kafka task to send the messages, you should see 9965 objects once you read the created *parquet* file into a dataframe. Use df.count() to check if all objects have been stored correctly.

Additionally, using the command df.head() should return a single tweet with all fields in the original JSON message plus your new attribute of sentiment.

Suggestions:

* Make use of existing libraries for spark streaming and Kafka. Pay attention to versioning as different versions may not be compatible with each other and with certain languages.
* The following site is useful for getting started with Spark Streaming: <https://spark.apache.org/docs/2.1.0/streaming-programming-guide.html>
* The following site is useful for getting started with Spark Streaming with Kafka messages (pay attention to the Kafka version!): <https://spark.apache.org/docs/latest/streaming-kafka-integration.html>

**Submission Requirements:**

Create a zip file named mp2\_netid.zip containing the following files and submit under the MP2 assignment on compass2g.

README.txt: A file to provide a brief overview of your solution as well as any noteworthy deviations from this document.

* run\_kafka\_task.sh: as described in the Kafka portion of this document
* Any and all files that run\_kafka\_task.sh requires to operate successfully.
* Spark streaming script.
* Spark\_streaming\_output.txt. This text file should contain 3 things: 1) the results of running df.count() in Spark on loading the saved parquet file and 2) the results of running df.head() in Spark.
* Pdf file with query details and screenshots of their results on Spark.