EECS E6893 Big Data Analytics - Fall 2019

Homework Assignment 3: Twitter data analysis with Spark Streaming

Due Friday, November 1st, 2019 by 5:00pm

TL;DR

1. Do twitter streaming analysis using Spark Streaming
2. Count hashtags and special words
3. Submit your codes and report to Canvas

Abstract:

In this assignment, you will implement a streaming analysis process. The architecture is as follows. A socket request data from twitter API and send data to spark streaming process. Spark read real time data and do analysis. It also save temp streaming results to Google Storage. After the streaming process terminate, it reads the final data from Google Storage and save it to BigQuery, and then clean the data in Storage.

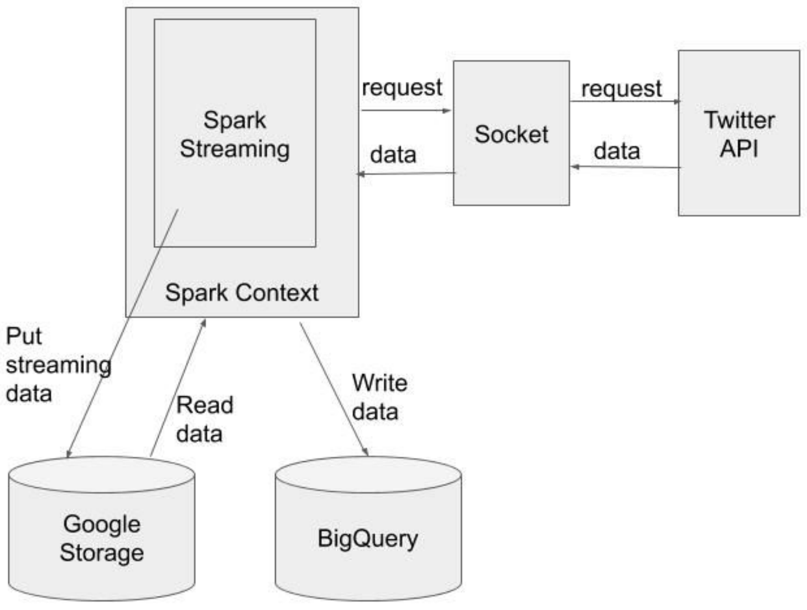


Fig 1. Twitter streaming architecture

You will be asked to do two analysis tasks based on the tweets you get. And you need to figure out how to save temp steaming results to google storage.

Tasks:

1. Calculate the accumulated hashtags count sum for 600 seconds and sort it by descending order of the count. Hashtag usually starts with "#" and followed by a series of alphanumeric. Hashtags are case insensitive. [40%]
2. Filter the chosen 5 words and calculate the appearance frequency of them in 60 seconds for every 60 seconds (no overlap). For example, word ‘data’ appears 30 times from T1 to T2 (T2-T1=60s), and 20 times from T2 to T3(T3-T2=60s), then the output DStream should have inner structure (RDD@T2(‘data’, 30, T2), RDD@T3(‘data’,20, T3), RDD@T4 …). Collect the data for 600 seconds. The words are: 'data', 'spark', 'ai', 'movie', 'good'. Ignore case sensitive when filter the data. [40%]
3. Save results to google BigQuery. You only have to write code to save temp steaming results to google storage. For hashtags count, you only need to save the latest rdd in DStream. For word count, you should save each rdd in DStream. [20%]

Some important notes:

1. Remember to start the socket first (run twitterHTTPClient.py) and then the streaming process (run sparkStreaming.py).
2. You **don’t need to restart the socket process** every time you rerun the streaming process. When a streaming process stops, the connection socket will automatically close. And the listening socket will continue listening to the same IP and Port and wait for the next connection. That is, you can leave twitterHTTPClient.py running and submit sparkStreaming.py multiple times.
3. Remember to stop the socket program on cluster when you don’t want to stream data from it.
4. You will get “port already in use” error when you run socket program multiple times in a short time period, or you try to run multiple socket programs at the same time.
5. For tasks (2), it is ok if you don’t see all of the words in your results every time.
6. For tasks (3), create a table first before saving results to it.
7. You can stop your cluster instance in Computer Engine page and keep your cluster. If you stop your instance, you won’t pay too much for having that cluster.
8. You can use Jupyter Notebook, just copy and paste the code. But **remember to create two different notebooks** and paste the code of two files in separate notebooks. The steps to run the code is the same as you submit the file to dataproc.

Steps:

Step1: Register on Twitter Apps (Please do this step ASAP)

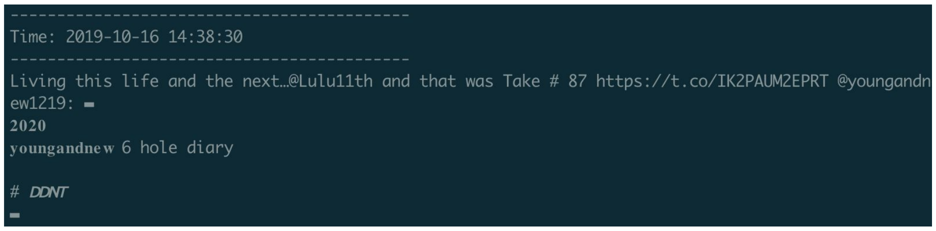
1. Go to https://developer.twitter.com/en/apply-for-access.html and apply for a twitter developer account.
2. Login at https://apps.twitter.com/
3. Click “Create New App”, fill out the form, and click “Create your Twitter application”. For the Website URL, you can put any URL here. You only need to fill in all the required places.
4. In the next page, click on “API keys” tab, and copy your “API key” and “API secret”.
5. Scroll down and click “Create my access token”, copy your “Access token” and “Access token secret”.

Step2: Create a cluster and get streaming data

1. Use the following command to create a cluster.
```
gcloud beta dataproc clusters create <cluster-name> \
--optional-components=ANACONDA,JUPYTER \
--image-version=preview --enable-component-gateway \
--metadata 'PIP_PACKAGES=requests_oauthlib google-cloud-bigquery
tweepy' \
--metadata gcs-connector-version=1.9.16 \
--metadata bigquery-connector-version=0.13.16 \
--project <ProjectID> \
--bucket <Bucket> \
--initialization-actions=gs://dataproc-initialization-actions/pyth
on/pip-install.sh,gs://dataproc-initialization-actions/connectors/
connectors.sh \
--single-node
```
2. Download start code from canvas.
3. Open *twitterHTTPClient.py* Replace the Twitter API credential with your own and run on dataproc. Remember to stop the job after you finish streaming.
4. Open *sparkStreaming.py.* You can first comment code from line 158 (words=...) to line 166 (wordCount.pprint()) and from line 186 (saveToBigQuery) to line 187 (saveToBigQuery). And then run it on dataproc.
5. To stream the data:
    a. First submit twitterHTTPClient.py. (Jupyter Notebook: run the related cells in one notebook) Wait for “Waiting for TCP connection…” appears.
    b. Then open another terminal, submit sparkStreaming.py. (Jupyter Notebook: create a new notebook, run the related cells in the new notebook) You will see “Connected...Starting getting tweets” appears in

the terminal where you run twitterHTTPClient.py and some tweets start to be printed out in both terminals.

6. You are expected to see a tweet stream printed out like this. Ignore all warnings while running.



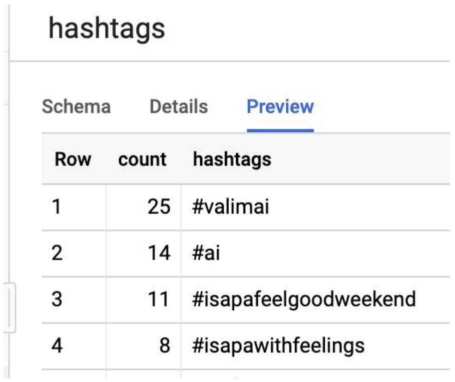
```
-------------------------------------------
Time: 2019-10-16 14:38:30
-------------------------------------------
Living this life and the next…@Lulu11th and that was Take # 87 https://t.co/IK2PAUM2EPRT @youngandn
ew1219: ▬
2020
youngandnew 6 hole diary

# DDNT
▬
```

7. You can test sparkStreaming.py multiple times and leave twitterHTTPClient.py running
8. Stop twitterHTTPClient.py. You can go to job page of cluster and cancel it. Or you can use `gcloud dataproc jobs kill` **JOB**. (Jupyter Notebook: stop the cell)

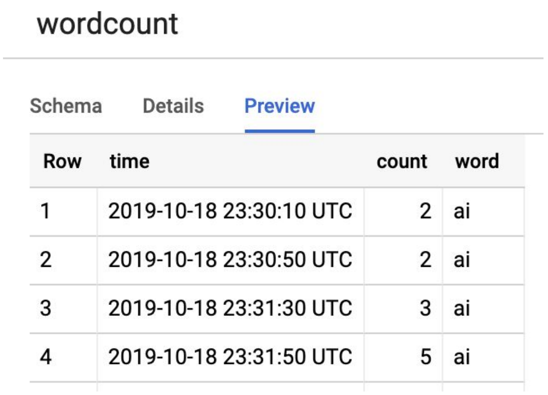
Step3: Analyse Stream data and store data in BigQuery

1. Once you successfully get the stream, you can start writing your own code. The analytics tasks are listed before. Tasks
2. You can first change the global variable “STREAMTIME” to a smaller value when testing your algorithm. And then collect the data for 600 seconds.
3. You can first set your streaming window size to a smaller value (less than 60 seconds). After you make sure your code is correct and then you can change it back to 60 seconds and start collecting the data.
4. To save the data, remember to:
    a. Replace the bucket global variable in the code with your own bucket.
    b. Create a BigQuery dataset first using
        bq mk <your dataset name>.
5. You are expected to see the following tables created in your dataset



hashtags

Row	count	hashtags
1	25	#valimai
2	14	#ai
3	11	#isapafeelgoodweekend
4	8	#isapawithfeelings



wordcount

Row	time	count	word
1	2019-10-18 23:30:10 UTC	2	ai
2	2019-10-18 23:30:50 UTC	2	ai
3	2019-10-18 23:31:30 UTC	3	ai
4	2019-10-18 23:31:50 UTC	5	ai

Homework Submissions

1. A report includes:
    a. Screenshot of your code to do all the tasks. Tasks
    b. Screenshot of the preview of your data stored in BigQuery. You have to include two tables: *hashtags* and *wordcount.*
    It’s fine that you rows in hashtags is not ordered. You can just provide a screenshot of the tuples printed in your terminal to prove you have correctly sorted the data.
2. Your code

Useful links:
https://spark.apache.org/docs/latest/streaming-programming-guide.html
https://docs.python.org/3/library/re.html