EECS E6893 Big Data Analytics - Fall 2021

Homework Assignment 2: Classification and Twitter data analysis with Spark Streaming

Due Friday, October 21st, 2022, by 5:00pm

TL;DR
1. Process data with Spark Dataframe and perform classification with Spark MLlib
2. Do Twitter streaming analysis using Spark Streaming
3. Count hashtags and special words
4. Submit your codes and report to Canvas

Abstract:
In part I, you will learn how to use the built-in Spark MLlib library to conduct supervised and unsupervised learning, then have experience of processing data with *ML Pipeline* and *Dataframe*.

You will load data into Spark Dataframe and perform **binary classification** with Spark MLlib. Use Logistic regression, Random Forest, Naive Bayes, Decision Tree, Gradient Boosting Trees, Multi-layer perceptron, Linear Support Vector Machine, One-vs-Rest to do classification to the same dataset, then evaluate each result and compare the performances of these methods.

In part II, you will implement a streaming analysis process. The architecture is as follows. A socket requests data from Twitter API and sends data to the Spark streaming process. Spark reads real-time data to do analysis. It also saves temp streaming results to Google Storage. After the streaming process terminates, Spark reads the final data from Google Storage and saves it to BigQuery, and then cleans the data in Storage. Finally, you will try how to use LDA to classify the data in the streaming.

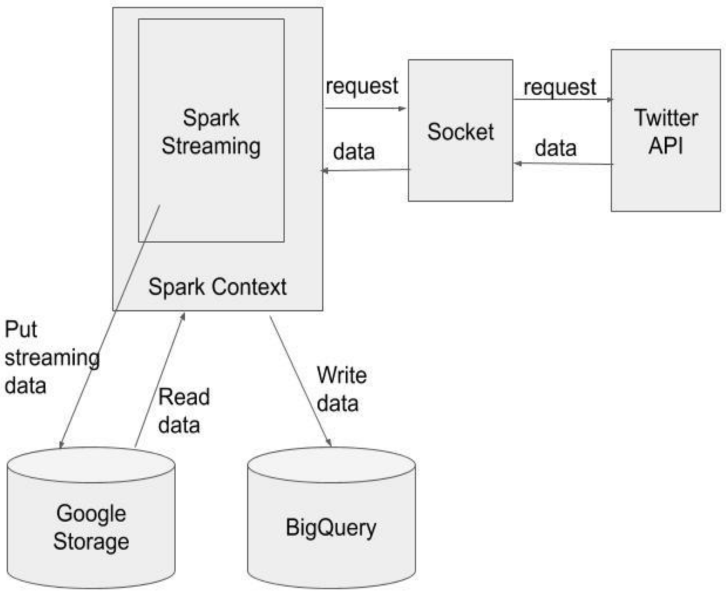


Fig 1. Twitter streaming architecture

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

Part I Tasks(40%):

In this question, we are going to use the Adult dataset, which is publicly available at the UCI Machine Learning Repository. You can download the *adult.data* from this link, and add .csv extension to the download file.

This data derives from census data and consists of information about 48k individuals and their annual income. We are interested in using the information to predict if an individual earns less than 50K or larger than 50k per year. It is natural to formulate the problem as binary classification.

The model we use for binary classification is logistic regression, Random Forest, Naive Bayes, Decision Tree, Gradient Boosting Trees, Multi-layer perceptron, Linear Support Vector Machine, One-vs-Rest, totally 9 models. Take Logistic Regression as an example, it takes one more step than linear regression: apply the sigmoid function to compress the output values to 0~1 and compare with the threshold to see which cluster it should belong to. Linear regression outputs continuous values while logistic regression outputs probabilities in the range 0 to 1 which can then be mapped to two or more discrete classes. You could learn more details like how Spark performs optimization here.

Steps:

1. Data loading: (5%)
   Read the csv file into a Dataframe. You could set *"inferschema"* to true and rename the columns with the following information: "age", "workclass", "fnlwgt", "education", "education_num", "marital_status", "occupation", "relationship", "race", "sex", "capital_gain", "capital_loss", "hours_per_week", "native_country", "income". You could learn more about Dataframe from this doc.
2. Data preprocessing: (5%)
   Convert the categorical variables into numeric variables with ML Pipelines and Feature Transformers. You will probably need *OneHotEncoderEstimator, StringIndexer, and VectorAssembler.* Split your data into the training and test sets, respectively, with a ratio of 70% and 30% and set the seed to 100.
3. Modelling: (15%)
   Train logistic regression, Random Forest, Naive Bayes, Decision Tree, Gradient Boosting Trees, Multi-layer perceptron, Linear Support Vector Machine, One-vs-Rest from Spark MLlib with the training set.
4. Evaluation: (15%)
   Apply all the models on the test set, analyze and compare their accuracies to rank these methods.

Part II Tasks(60%):

1. Calculate the accumulated hashtags count sum for 600 seconds and sort it by descending order of the count. Hashtag usually starts with "#" followed by a series of alphanumeric. Hashtags are case insensitive. (15%)
2. Filter the chosen 5 words (you can pick up any five words as your interest) 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 filtering the data. (15%)
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. (15%)
4. Do Clustering to your streaming using LDA. (15%)

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 task (2), it is ok if you don't see all of the words in your results every time.
6. For task (3), create a table first before saving results to it.
7. You can stop your cluster instance in the Compute 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 are the same as you submit the file to Dataproc.

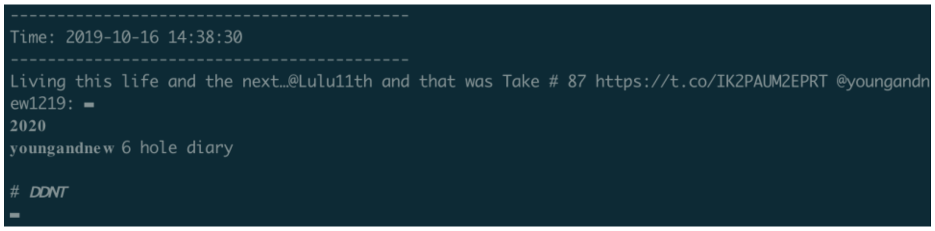
Steps:

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

5. Use the following command to create a cluster.
```
gcloud beta dataproc clusters create <cluster-name> \
--region=<region> \
--optional-components=ANACONDA,JUPYTER \
--image-version=1.4 \
--enable-component-gateway \
--initialization-actions=gs://dataproc-initialization-actions/python/pip-
install.sh,gs://dataproc-initialization-actions/connectors/connectors.sh \
--metadata "PIP_PACKAGES=requests_oauthlib google-cloud-bigquery tweepy==3.1.0" \
--metadata gcs-connector-version=1.9.16 \
--metadata bigquery-connector-version=1.0.0 \
--project <ProjectID> \
--bucket <bucket-name> \
--single-node
```
6. Download the start code from Canvas.
7. Open *twitterHTTPClient.py* Replace the Twitter API credential with your own and run on Dataproc. Remember to stop the job after you finish streaming.
8. Open *sparkStreaming.py.* You can first comment code from line 158 (words=...) to line 166 (wordCount.print()) and from line 186 (saveToBigQuery) to line 187 (saveToBigQuery). And then run it on dataproc.
9. 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.
10. You are expected to see a tweet stream printed out like this. Ignore all warnings while running.



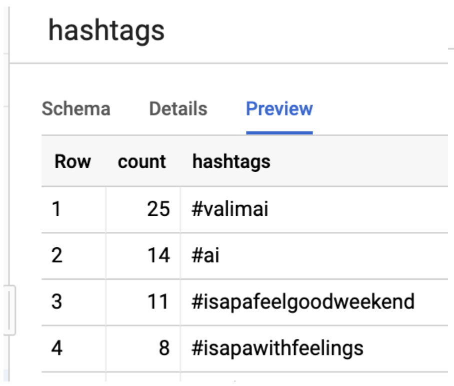
```
-------------------------------------------
Time: 2019-10-16 14:38:30
-------------------------------------------
Living this life and the next…@Lulu11th and that was Take # 87 https://t.co/IK2PAUM2EPRT @youngandn
ew1219: ▬
2020
youngandnew 6 hole diary

# DDNT
▬
```

11. You can test sparkStreaming.py multiple times and leave twitterHTTPClient.py running
12. 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

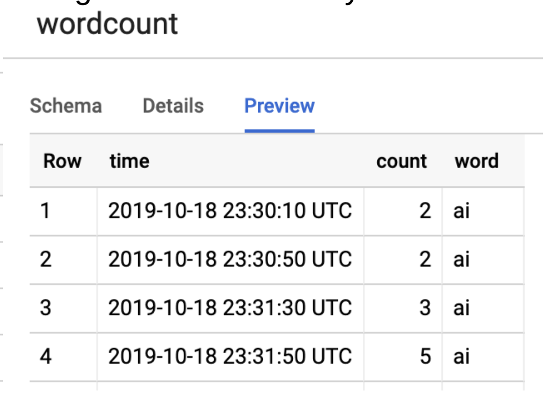
1. Once you successfully get the stream, you can start writing your own code. The analytics tasks are listed before *Tasks.*
2. You can first change the global variable "STREAMTIME" to a smaller value when testing your algorithm. And then collect the data for 600 seconds.
3. You can first set your streaming 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

Schema Details Preview

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



wordcount

Schema Details Preview

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

Step4: Do Clustering to your Stream data by LDA.

1. Load your streaming (the data loaded for LDA model should be the raw data you get from Twitter streaming in previous steps. You can use the 600s data and only keep English if it appears some issues in later modeling.)
2. Do clustering
3. Check the weight of every topic distribution
4. Output topic and vocabulary distribution (DenseMatrix is enough)

Homework Submissions

1. Part I: save the ipynb file as pdf format, containing your code and result.
2. Part II: (Task1-3) A report includes:
    a. Screenshot of your code to do all the tasks.
    b. Screenshot of the preview of your data stored in BigQuery. You have to include two tables: *hashtags* and *wordcount.*
3. Part II: (Task4) save the ipynb file as pdf format, containing your code and result.

Useful links:
https://spark.apache.org/docs/latest/streaming-programming-guide.html
https://docs.python.org/3/library/re.html
https://github.com/apache/spark/blob/branch-2.1/mllib/src/main/scala/org/apache/spark/ml/clustering/LDA.scala