

Extract and Learn

We were required to utilize spark structured streaming to extract the data from a Kafka server and by using the ML model created in the previous task – predict the label of every row in the streamed data.

Constraints:

1. Not allowed to predict data that the model was already trained on
2. Must predict **ALL** the data

Solution:

We were asked to explain the training process and provide the accuracy achieved throughout the running of the code.

Training process explained

For each batch created by a trigger, we wanted to predict the label of every row in the batch.

For the first batch, we still don't have any data to train the model, therefore we decided to predict all the labels as one of the labels in the batch.

Now we already had made a prediction and we can start presenting the total accuracy (correct predictions / total predictions). After making a prediction we trained the model on all the data accumulated including the last batch we made predictions on.

For each batch from the second batch onwards we would: predict the labels of the new batch by using the most updated model (trained on all the data except for the data in the current batch), calculate the number of correct predictions and total predictions made, present the current accuracy, update the model by training it on all the data including the current batch and repeat for each new batch collected from the stream.

Details

We used the writestream sink function “forEachBatch()” to isolate the new batch as a df and by maintaining global variables to assist with the information gathered.

The global variables used:

1. Spark data frame containing all the data collected until the newest batch.
2. Int representing the current number of correct predictions.
3. Int representing the total predictions.
4. ML model which is updated at the end of every batch and used for predicting the next batch.

First, we needed to solve the server version constraints. StringIndexer used in the ml lib from the previous question was not supported on the 2.4 spark version installed on the server, therefore, we decided to implement the idea behind the string indexer on our own. We used a dictionary and a the “replace” transformation in spark to update the ‘gt’ column from a string to an index. For the ‘user’ index we implemented a OneHotEncoder with spark transformations to prepare the data to be assembled by the VectorAssembler supported in spark 2.4.

After preparing the data collected from each batch, we would predict using the most updated model the labels, calculate the correct predictions and append the processed data from the batch to the

unbounded data frame maintained (using the 'union' function in spark). After the union we would update the model by training it on the updated unbounded data frame.

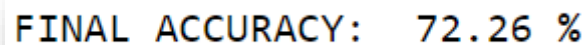
While running the code on the server, we came across heap size issues and lengthy running times when training the ML model on large amount of data . The way we were able to solve these issues without changing the server configurations was to stop training the model after being trained on enough data. We trained the data on 40 batches of 100,000 rows, by setting the max off set per trigger to 100,000 and by making sure there was no “memory” sink in the code. We assured we weren't writing to a memory sink by using the `foreachBatch` sink option. This way we would train our model on a total of 4 million rows (approximately two thirds of the total dynamic stream).

All we did from this point onwards was to predict new data that came into the stream by using the model we trained until now. We found the accuracy results to be the same as training on all the data, therefore we didn't loose any accuracy by not training the model on all the model until the end of the stream. We found the accuracy to converge to the accuracy we found in the previous part of the project.

Once the stream was not streaming any new data, we used `stream_obj.stop()` terminate the program by using a global flag. We tried using the `awaitTermination()` and from within the `foreachBatch` function call the `.stop()` but it wasn't terminating the program successfully, so we found a creative solution which work well on the server.

The final code runtime we submitted took 48 minutes to run,

Our **final accuracy achieved for the bonus section** assignment as shown in the logs on the following pages:

A screenshot of a terminal window with a dark background. The text "FINAL ACCURACY: 72.26 %" is displayed in a light blue, monospaced font. The text is centered horizontally and appears to be part of a larger log or output file.

FINAL ACCURACY: 72.26 %

Results (as shown on the server output logs):

```
Container: container_1662792524764_0020_01_000001 on wn4-spark9.qbgu2r2vo3buliqsaungd2sltg.bx.internal.cloudapp.net_30050_1662809123024
LogAggregationType: AGGREGATED
=====
LogType:stdout
LogLastModifiedTime:Sat Sep 10 11:25:23 +0000 2022
LogLength:7968
LogContents:
(Batch: 1 ) Current Accuracy: 15.26 % --- (total predicted: 100000 , total correct: 15263 , correct in batch: 15263 )
(Batch: 2 ) Current Accuracy: 42.16 % --- (total predicted: 200000 , total correct: 84327 , correct in batch: 69064 )
(Batch: 3 ) Current Accuracy: 51.55 % --- (total predicted: 300000 , total correct: 154641 , correct in batch: 70314 )
(Batch: 4 ) Current Accuracy: 57.39 % --- (total predicted: 400000 , total correct: 229541 , correct in batch: 74900 )
(Batch: 5 ) Current Accuracy: 60.01 % --- (total predicted: 500000 , total correct: 300071 , correct in batch: 70530 )
(Batch: 6 ) Current Accuracy: 62.31 % --- (total predicted: 600000 , total correct: 373831 , correct in batch: 73760 )
(Batch: 7 ) Current Accuracy: 63.75 % --- (total predicted: 700000 , total correct: 446219 , correct in batch: 72388 )
(Batch: 8 ) Current Accuracy: 64.96 % --- (total predicted: 800000 , total correct: 519658 , correct in batch: 73439 )
(Batch: 9 ) Current Accuracy: 66.01 % --- (total predicted: 900000 , total correct: 594059 , correct in batch: 74401 )
(Batch: 10 ) Current Accuracy: 66.81 % --- (total predicted: 1000000 , total correct: 668122 , correct in batch: 74063 )
(Batch: 11 ) Current Accuracy: 67.43 % --- (total predicted: 1100000 , total correct: 741775 , correct in batch: 73653 )
(Batch: 12 ) Current Accuracy: 67.96 % --- (total predicted: 1200000 , total correct: 815527 , correct in batch: 73752 )
(Batch: 13 ) Current Accuracy: 68.59 % --- (total predicted: 1300000 , total correct: 891676 , correct in batch: 76149 )
(Batch: 14 ) Current Accuracy: 69.01 % --- (total predicted: 1400000 , total correct: 966208 , correct in batch: 74532 )
(Batch: 15 ) Current Accuracy: 69.27 % --- (total predicted: 1500000 , total correct: 1039055 , correct in batch: 72847 )
(Batch: 16 ) Current Accuracy: 69.47 % --- (total predicted: 1600000 , total correct: 1111522 , correct in batch: 72467 )
(Batch: 17 ) Current Accuracy: 69.64 % --- (total predicted: 1700000 , total correct: 1183931 , correct in batch: 72409 )
(Batch: 18 ) Current Accuracy: 69.78 % --- (total predicted: 1800000 , total correct: 1256028 , correct in batch: 72097 )
(Batch: 19 ) Current Accuracy: 70.03 % --- (total predicted: 1900000 , total correct: 1330613 , correct in batch: 74585 )
(Batch: 20 ) Current Accuracy: 70.12 % --- (total predicted: 2000000 , total correct: 1402411 , correct in batch: 71798 )
(Batch: 21 ) Current Accuracy: 70.24 % --- (total predicted: 2100000 , total correct: 1475083 , correct in batch: 72672 )
(Batch: 22 ) Current Accuracy: 70.38 % --- (total predicted: 2200000 , total correct: 1548338 , correct in batch: 73255 )
(Batch: 23 ) Current Accuracy: 70.52 % --- (total predicted: 2300000 , total correct: 1622058 , correct in batch: 73720 )
(Batch: 24 ) Current Accuracy: 70.52 % --- (total predicted: 2400000 , total correct: 1692371 , correct in batch: 70313 )
(Batch: 25 ) Current Accuracy: 70.67 % --- (total predicted: 2500000 , total correct: 1766689 , correct in batch: 74318 )
(Batch: 26 ) Current Accuracy: 70.78 % --- (total predicted: 2600000 , total correct: 1840306 , correct in batch: 73617 )
(Batch: 27 ) Current Accuracy: 70.94 % --- (total predicted: 2700000 , total correct: 1915397 , correct in batch: 75091 )
(Batch: 28 ) Current Accuracy: 70.97 % --- (total predicted: 2800000 , total correct: 1987091 , correct in batch: 71694 )
(Batch: 29 ) Current Accuracy: 70.95 % --- (total predicted: 2900000 , total correct: 2057661 , correct in batch: 70570 )
(Batch: 30 ) Current Accuracy: 70.95 % --- (total predicted: 3000000 , total correct: 2128387 , correct in batch: 70726 )
(Batch: 31 ) Current Accuracy: 71.04 % --- (total predicted: 3100000 , total correct: 2202106 , correct in batch: 73719 )
(Batch: 32 ) Current Accuracy: 71.1 % --- (total predicted: 3200000 , total correct: 2275315 , correct in batch: 73209 )
(Batch: 33 ) Current Accuracy: 71.19 % --- (total predicted: 3300000 , total correct: 2349344 , correct in batch: 74029 )
(Batch: 34 ) Current Accuracy: 71.25 % --- (total predicted: 3400000 , total correct: 2422574 , correct in batch: 73230 )
(Batch: 35 ) Current Accuracy: 71.28 % --- (total predicted: 3500000 , total correct: 2494730 , correct in batch: 72156 )
(Batch: 36 ) Current Accuracy: 71.41 % --- (total predicted: 3600000 , total correct: 2570787 , correct in batch: 76057 )
(Batch: 37 ) Current Accuracy: 71.44 % --- (total predicted: 3700000 , total correct: 2643101 , correct in batch: 72314 )
(Batch: 38 ) Current Accuracy: 71.47 % --- (total predicted: 3800000 , total correct: 2715896 , correct in batch: 72795 )
(Batch: 39 ) Current Accuracy: 71.47 % --- (total predicted: 3900000 , total correct: 2787283 , correct in batch: 71387 )
(Batch: 40 ) Current Accuracy: 71.52 % --- (total predicted: 4000000 , total correct: 2860922 , correct in batch: 73639 )
(Batch: 41 ) Current Accuracy: 71.56 % --- (total predicted: 4100000 , total correct: 2933895 , correct in batch: 72973 )
(Batch: 42 ) Current Accuracy: 71.6 % --- (total predicted: 4200000 , total correct: 3007180 , correct in batch: 73285 )
(Batch: 43 ) Current Accuracy: 71.67 % --- (total predicted: 4300000 , total correct: 3081698 , correct in batch: 74518 )
(Batch: 44 ) Current Accuracy: 71.7 % --- (total predicted: 4400000 , total correct: 3154671 , correct in batch: 72973 )
(Batch: 45 ) Current Accuracy: 71.73 % --- (total predicted: 4500000 , total correct: 3227781 , correct in batch: 73110 )
(Batch: 46 ) Current Accuracy: 71.78 % --- (total predicted: 4600000 , total correct: 3301846 , correct in batch: 74065 )
(Batch: 47 ) Current Accuracy: 71.82 % --- (total predicted: 4700000 , total correct: 3375709 , correct in batch: 73863 )
(Batch: 48 ) Current Accuracy: 71.84 % --- (total predicted: 4800000 , total correct: 3448486 , correct in batch: 72777 )
(Batch: 49 ) Current Accuracy: 71.89 % --- (total predicted: 4900000 , total correct: 3522514 , correct in batch: 74028 )
(Batch: 50 ) Current Accuracy: 71.93 % --- (total predicted: 5000000 , total correct: 3596358 , correct in batch: 73844 )
```

```
(Batch: 51 ) Current Accuracy: 71.96 % --- (total predicted: 5100000 , total correct: 3670155 , correct in batch: 73797 )
(Batch: 52 ) Current Accuracy: 71.97 % --- (total predicted: 5200000 , total correct: 3742398 , correct in batch: 72243 )
(Batch: 53 ) Current Accuracy: 72.03 % --- (total predicted: 5300000 , total correct: 3817767 , correct in batch: 75369 )
(Batch: 54 ) Current Accuracy: 72.05 % --- (total predicted: 5400000 , total correct: 3890799 , correct in batch: 73032 )
(Batch: 55 ) Current Accuracy: 72.08 % --- (total predicted: 5500000 , total correct: 3964245 , correct in batch: 73446 )
(Batch: 56 ) Current Accuracy: 72.1 % --- (total predicted: 5600000 , total correct: 4037563 , correct in batch: 73318 )
(Batch: 57 ) Current Accuracy: 72.14 % --- (total predicted: 5700000 , total correct: 4112242 , correct in batch: 74679 )
(Batch: 58 ) Current Accuracy: 72.15 % --- (total predicted: 5800000 , total correct: 4184953 , correct in batch: 72711 )
(Batch: 59 ) Current Accuracy: 72.17 % --- (total predicted: 5900000 , total correct: 4257755 , correct in batch: 72802 )
(Batch: 60 ) Current Accuracy: 72.21 % --- (total predicted: 6000000 , total correct: 4332682 , correct in batch: 74927 )
(Batch: 61 ) Current Accuracy: 72.23 % --- (total predicted: 6100000 , total correct: 4405822 , correct in batch: 73140 )
(Batch: 62 ) Current Accuracy: 72.25 % --- (total predicted: 6200000 , total correct: 4479655 , correct in batch: 73833 )
(Batch: 63 ) Current Accuracy: 72.26 % --- (total predicted: 6240991 , total correct: 4509795 , correct in batch: 30140 )
```

```
----- No more data being streamed, FINAL ACCURACY: 72.26 % -----
```

End of LogType:stdout

Note:

Adding the ID of the output for you to be able to look up the logs if needed.

Very happy to see this question finally working. Learnt a lot from the journey.

Your code is now being checked. The job id is 1575.
You can check the status at [this link](#)

[List of all jobs from everyone](#)
[Check the Leaderboard](#)

Job 1575 completed in 0.000 seconds.STDOUT:

```
>>> Uploading source file /tmp/tmp7ftzdljc/323371385_FINAL.py

>>> Sending source for execution
{"id":13,"state":"starting","appId":null,"appInfo":{"driverLogUrl":null,"sparkUiUrl":null},"log":["stdout: ","\nstderr: ","\nYARN Diagnostics: "]}
=====
>>> Job is starting...
BATCH ID = 13
=====
To see the logs: http://jobs.eastus.cloudapp.azure.com/spark/logs?batchId=13
=====

To manually delete this job, visit the link below. WARNING: no confirmation! I will try to delete without questions!
http://jobs.eastus.cloudapp.azure.com/spark/delete?batchId=13
```

STDERR: