import os

import time

from twitter.cortex.ml.embeddings.deepbird.grouped\_metrics.computation import (

write\_grouped\_metrics\_to\_mldash,

)

from twitter.cortex.ml.embeddings.deepbird.grouped\_metrics.configuration import (

ClassificationGroupedMetricsConfiguration,

NDCGGroupedMetricsConfiguration,

)

import twml

from .light\_ranking\_metrics import (

CGRGroupedMetricsConfiguration,

ExpectedLossGroupedMetricsConfiguration,

RecallGroupedMetricsConfiguration,

)

import numpy as np

import tensorflow.compat.v1 as tf

from tensorflow.compat.v1 import logging

# checkstyle: noqa

def run\_group\_metrics(trainer, data\_dir, model\_path, parse\_fn, group\_feature\_name="meta.user\_id"):

start\_time = time.time()

logging.info("Evaluating with group metrics.")

metrics = write\_grouped\_metrics\_to\_mldash(

trainer=trainer,

data\_dir=data\_dir,

model\_path=model\_path,

group\_fn=lambda datarecord: str(

datarecord.discreteFeatures[twml.feature\_id(group\_feature\_name)[0]]

),

parse\_fn=parse\_fn,

metric\_configurations=[

ClassificationGroupedMetricsConfiguration(),

NDCGGroupedMetricsConfiguration(k=[5, 10, 20]),

],

total\_records\_to\_read=1000000000,

shuffle=False,

mldash\_metrics\_name="grouped\_metrics",

)

end\_time = time.time()

logging.info(f"Evaluated Group Metics: {metrics}.")

logging.info(f"Group metrics evaluation time {end\_time - start\_time}.")

def run\_group\_metrics\_light\_ranking(

trainer, data\_dir, model\_path, parse\_fn, group\_feature\_name="meta.trace\_id"

):

start\_time = time.time()

logging.info("Evaluating with group metrics.")

metrics = write\_grouped\_metrics\_to\_mldash(

trainer=trainer,

data\_dir=data\_dir,

model\_path=model\_path,

group\_fn=lambda datarecord: str(

datarecord.discreteFeatures[twml.feature\_id(group\_feature\_name)[0]]

),

parse\_fn=parse\_fn,

metric\_configurations=[

CGRGroupedMetricsConfiguration(lightNs=[50, 100, 200], heavyKs=[1, 3, 10, 20, 50]),

RecallGroupedMetricsConfiguration(n=[50, 100, 200], k=[1, 3, 10, 20, 50]),

ExpectedLossGroupedMetricsConfiguration(lightNs=[50, 100, 200]),

],

total\_records\_to\_read=10000000,

num\_batches\_to\_load=50,

batch\_size=1024,

shuffle=False,

mldash\_metrics\_name="grouped\_metrics\_for\_light\_ranking",

)

end\_time = time.time()

logging.info(f"Evaluated Group Metics for Light Ranking: {metrics}.")

logging.info(f"Group metrics evaluation time {end\_time - start\_time}.")

def run\_group\_metrics\_light\_ranking\_in\_bq(trainer, params, checkpoint\_path):

logging.info("getting Test Predictions for Light Ranking Group Metrics in BigQuery !!!")

eval\_input\_fn = trainer.get\_eval\_input\_fn(repeat=False, shuffle=False)

info\_pool = []

for result in trainer.estimator.predict(

eval\_input\_fn, checkpoint\_path=checkpoint\_path, yield\_single\_examples=False

):

traceID = result["trace\_id"]

pred = result["prediction"]

label = result["target"]

info = np.concatenate([traceID, pred, label], axis=1)

info\_pool.append(info)

info\_pool = np.concatenate(info\_pool)

locname = "/tmp/000/"

if not os.path.exists(locname):

os.makedirs(locname)

locfile = locname + params.pred\_file\_name

columns = ["trace\_id", "model\_prediction", "meta\_\_ranking\_\_weighted\_oonc\_model\_score"]

np.savetxt(locfile, info\_pool, delimiter=",", header=",".join(columns))

tf.io.gfile.copy(locfile, params.pred\_file\_path + params.pred\_file\_name, overwrite=True)

if os.path.isfile(locfile):

os.remove(locfile)

logging.info("Done Prediction for Light Ranking Group Metrics in BigQuery.")