package com.twitter.simclusters\_v2.scalding.tweet\_similarity.evaluation

import com.twitter.ml.api.Feature.Continuous

import com.twitter.ml.api.DailySuffixFeatureSource

import com.twitter.ml.api.DataSetPipe

import com.twitter.ml.api.RichDataRecord

import com.twitter.scalding.\_

import com.twitter.scalding.typed.TypedPipe

import com.twitter.scalding\_internal.job.TwitterExecutionApp

import com.twitter.simclusters\_v2.tweet\_similarity.TweetSimilarityFeatures

import com.twitter.twml.runtime.scalding.TensorflowBatchPredictor

import java.util.TimeZone

/\*\*

\* Scalding execution app for scoring a Dataset against an exported Tensorflow model.

\*\* Arguments:

\* dataset\_path - Path for the dataset on hdfs

\* date - Date for the dataset paths, required if Daily dataset.

\* model\_source - Path of the exported model on HDFS. Must start with hdfs:// scheme.

\* output\_path - Path of the output result file

scalding remote run --target src/scala/com/twitter/simclusters\_v2/scalding/tweet\_similarity:model\_eval-adhoc \

--user cassowary \

--submitter hadoopnest2.atla.twitter.com \

--main-class com.twitter.simclusters\_v2.scalding.tweet\_similarity.ModelEvalAdhocApp -- \

--date 2020-02-19 \

--dataset\_path /user/cassowary/adhoc/training\_data/2020-02-19\_class\_balanced/test \

--model\_path hdfs:///user/cassowary/tweet\_similarity/2020-02-07-15-20-15/exported\_models/1581253926 \

--output\_path /user/cassowary/adhoc/training\_data/2020-02-19\_class\_balanced/test/prediction\_v1

\*\*/

object ModelEvalAdhocApp extends TwitterExecutionApp {

implicit val timeZone: TimeZone = DateOps.UTC

implicit val dateParser: DateParser = DateParser.default

/\*\*

\* Get predictor for the given model path

\* @param modelName name of the model

\* @param modelSource path of the exported model on HDFS. Must start with hdfs:// scheme.

\* @return

\*/

def getPredictor(modelName: String, modelSource: String): TensorflowBatchPredictor = {

val defaultInputNode = "request:0"

val defaultOutputNode = "response:0"

TensorflowBatchPredictor(modelName, modelSource, defaultInputNode, defaultOutputNode)

}

/\*\*

\* Given input pipe and predictor, return the predictions in TypedPipe

\* @param dataset dataset for prediction

\* @param batchPredictor predictor

\* @return

\*/

def getPrediction(

dataset: DataSetPipe,

batchPredictor: TensorflowBatchPredictor

): TypedPipe[(Long, Long, Boolean, Double, Double)] = {

val featureContext = dataset.featureContext

val predictionFeature = new Continuous("output")

batchPredictor

.predict(dataset.records)

.map {

case (originalDataRecord, predictedDataRecord) =>

val prediction = new RichDataRecord(predictedDataRecord, featureContext)

.getFeatureValue(predictionFeature).toDouble

val richDataRecord = new RichDataRecord(originalDataRecord, featureContext)

(

richDataRecord.getFeatureValue(TweetSimilarityFeatures.QueryTweetId).toLong,

richDataRecord.getFeatureValue(TweetSimilarityFeatures.CandidateTweetId).toLong,

richDataRecord.getFeatureValue(TweetSimilarityFeatures.Label).booleanValue,

richDataRecord.getFeatureValue(TweetSimilarityFeatures.CosineSimilarity).toDouble,

prediction

)

}

}

override def job: Execution[Unit] =

Execution.withId { implicit uniqueId =>

Execution.withArgs { args: Args =>

implicit val dateRange: DateRange = DateRange.parse(args.list("date"))

val outputPath: String = args("output\_path")

val dataset: DataSetPipe = DailySuffixFeatureSource(args("dataset\_path")).read

val modelSource: String = args("model\_path")

val modelName: String = "tweet\_similarity"

getPrediction(dataset, getPredictor(modelName, modelSource))

.writeExecution(TypedTsv[(Long, Long, Boolean, Double, Double)](outputPath))

}

}

}