package com.twitter.simclusters\_v2.scalding.mbcg

import com.twitter.ml.api.DataRecord

import com.twitter.ml.api.embedding.Embedding

import com.twitter.ml.api.FeatureContext

import com.twitter.ml.api.FloatTensor

import com.twitter.ml.api.GeneralTensor

import com.twitter.ml.api.IRecordOneToOneAdapter

import com.twitter.ml.api.util.FDsl.\_

import com.twitter.simclusters\_v2.thriftscala.ClustersUserIsInterestedIn

import com.twitter.simclusters\_v2.thriftscala.PersistentSimClustersEmbedding

import scala.collection.JavaConverters.\_

/\*

Adapters to convert data from MBCG input sources into DataRecords

\*/

object TweetSimclusterRecordAdapter

extends IRecordOneToOneAdapter[(Long, PersistentSimClustersEmbedding, Embedding[Float])] {

override def getFeatureContext: FeatureContext = TweetAllFeatures.featureContext

override def adaptToDataRecord(

tweetFeatures: (Long, PersistentSimClustersEmbedding, Embedding[Float])

) = {

val dataRecord = new DataRecord()

val tweetId = tweetFeatures.\_1

val tweetEmbedding = tweetFeatures.\_2

val f2vEmbedding = tweetFeatures.\_3

val simclusterWithScores = tweetEmbedding.embedding.embedding

.map { simclusterWithScore =>

// Cluster ID and score for that cluster

(simclusterWithScore.\_1.toString, simclusterWithScore.\_2)

}.toMap.asJava

dataRecord.setFeatureValue(TweetAllFeatures.tweetId, tweetId)

dataRecord.setFeatureValue(TweetAllFeatures.tweetSimclusters, simclusterWithScores)

dataRecord.setFeatureValue(

TweetAllFeatures.authorF2vProducerEmbedding,

GeneralTensor.floatTensor(

new FloatTensor(f2vEmbedding.map(Double.box(\_)).asJava)

)

)

dataRecord

}

}

object UserSimclusterRecordAdapter

extends IRecordOneToOneAdapter[(Long, ClustersUserIsInterestedIn, Embedding[Float])] {

override def getFeatureContext: FeatureContext = TweetAllFeatures.featureContext

override def adaptToDataRecord(

userSimclusterEmbedding: (Long, ClustersUserIsInterestedIn, Embedding[Float])

) = {

val dataRecord = new DataRecord()

val userId = userSimclusterEmbedding.\_1

val userEmbedding = userSimclusterEmbedding.\_2

val simclusterWithScores = userEmbedding.clusterIdToScores

.filter {

case (\_, score) =>

score.logFavScore.map(\_ >= 0.0).getOrElse(false)

}

.map {

case (clusterId, score) =>

(clusterId.toString, score.logFavScore.get)

}.toMap.asJava

val f2vEmbedding = userSimclusterEmbedding.\_3

dataRecord.setFeatureValue(UserAllFeatures.userId, userId)

dataRecord.setFeatureValue(UserAllFeatures.userSimclusters, simclusterWithScores)

dataRecord.setFeatureValue(

UserAllFeatures.userF2vConsumerEmbedding,

GeneralTensor.floatTensor(

new FloatTensor(f2vEmbedding.map(Double.box(\_)).asJava)

)

)

dataRecord

}

}