package com.twitter.home\_mixer.product.scored\_tweets.feature\_hydrator

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.home\_mixer.param.HomeMixerInjectionNames.TwhinUserEngagementFeatureRepository

import com.twitter.home\_mixer.product.scored\_tweets.feature\_hydrator.adapters.twhin\_embeddings.TwhinUserEngagementEmbeddingsAdapter

import com.twitter.ml.api.DataRecord

import com.twitter.ml.api.{thriftscala => ml}

import com.twitter.product\_mixer.core.feature.Feature

import com.twitter.product\_mixer.core.feature.FeatureWithDefaultOnFailure

import com.twitter.product\_mixer.core.feature.datarecord.DataRecordInAFeature

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMap

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMapBuilder

import com.twitter.product\_mixer.core.functional\_component.feature\_hydrator.QueryFeatureHydrator

import com.twitter.product\_mixer.core.model.common.identifier.FeatureHydratorIdentifier

import com.twitter.product\_mixer.core.pipeline.PipelineQuery

import com.twitter.servo.repository.KeyValueRepository

import com.twitter.stitch.Stitch

import com.twitter.util.Return

import com.twitter.util.Throw

import javax.inject.Inject

import javax.inject.Named

import javax.inject.Singleton

import scala.collection.JavaConverters.\_

object TwhinUserEngagementFeature

extends DataRecordInAFeature[PipelineQuery]

with FeatureWithDefaultOnFailure[PipelineQuery, DataRecord] {

override def defaultValue: DataRecord = new DataRecord()

}

@Singleton

class TwhinUserEngagementQueryFeatureHydrator @Inject() (

@Named(TwhinUserEngagementFeatureRepository)

client: KeyValueRepository[Seq[Long], Long, ml.FloatTensor],

statsReceiver: StatsReceiver)

extends QueryFeatureHydrator[PipelineQuery] {

override val identifier: FeatureHydratorIdentifier =

FeatureHydratorIdentifier("TwhinUserEngagement")

override val features: Set[Feature[\_, \_]] = Set(TwhinUserEngagementFeature)

private val scopedStatsReceiver = statsReceiver.scope(getClass.getSimpleName)

private val keyFoundCounter = scopedStatsReceiver.counter("key/found")

private val keyLossCounter = scopedStatsReceiver.counter("key/loss")

private val keyFailureCounter = scopedStatsReceiver.counter("key/failure")

override def hydrate(query: PipelineQuery): Stitch[FeatureMap] = {

val userId = query.getRequiredUserId

Stitch.callFuture(client(Seq(userId))).map { results =>

val embedding: Option[ml.FloatTensor] = results(userId) match {

case Return(value) =>

if (value.exists(\_.floats.nonEmpty)) keyFoundCounter.incr()

else keyLossCounter.incr()

value

case Throw(\_) =>

keyFailureCounter.incr()

None

case \_ =>

None

}

val dataRecord =

TwhinUserEngagementEmbeddingsAdapter.adaptToDataRecords(embedding).asScala.head

FeatureMapBuilder()

.add(TwhinUserEngagementFeature, dataRecord)

.build()

}

}

}