package com.twitter.home\_mixer.functional\_component.feature\_hydrator

import com.twitter.home\_mixer.model.HomeFeatures.RealGraphInNetworkScoresFeature

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

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

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.stitch.Stitch

import com.twitter.storehaus.ReadableStore

import com.twitter.wtf.candidate.{thriftscala => wtf}

import javax.inject.Inject

import javax.inject.Named

import javax.inject.Singleton

@Singleton

case class RealGraphInNetworkScoresQueryFeatureHydrator @Inject() (

@Named(RealGraphInNetworkScores) store: ReadableStore[Long, Seq[wtf.Candidate]])

extends QueryFeatureHydrator[PipelineQuery] {

override val identifier: FeatureHydratorIdentifier =

FeatureHydratorIdentifier("RealGraphInNetworkScores")

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

private val RealGraphCandidateCount = 1000

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

Stitch.callFuture(store.get(query.getRequiredUserId)).map { realGraphFollowedUsers =>

val realGraphScoresFeatures = realGraphFollowedUsers

.getOrElse(Seq.empty)

.sortBy(-\_.score)

.map(candidate => candidate.userId -> scaleScore(candidate.score))

.take(RealGraphCandidateCount)

.toMap

FeatureMapBuilder().add(RealGraphInNetworkScoresFeature, realGraphScoresFeatures).build()

}

}

// Rescale Real Graph v2 scores from [0,1] to the v1 scores distribution [1,2.97]

private def scaleScore(score: Double): Double =

if (score >= 0.0 && score <= 1.0) score \* 1.97 + 1.0 else score

}