package com.twitter.usersignalservice.signals

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.simclusters\_v2.common.UserId

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

import com.twitter.strato.client.Client

import com.twitter.strato.data.Conv

import com.twitter.strato.thrift.ScroogeConv

import com.twitter.usersignalservice.base.Query

import com.twitter.wtf.candidate.thriftscala.CandidateSeq

import com.twitter.wtf.candidate.thriftscala.Candidate

import com.twitter.usersignalservice.base.StratoSignalFetcher

import com.twitter.usersignalservice.thriftscala.Signal

import com.twitter.usersignalservice.thriftscala.SignalType

import com.twitter.util.Future

import com.twitter.util.Timer

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

case class RealGraphOonFetcher @Inject() (

stratoClient: Client,

timer: Timer,

stats: StatsReceiver)

extends StratoSignalFetcher[UserId, Unit, CandidateSeq] {

import RealGraphOonFetcher.\_

override type RawSignalType = Candidate

override val name: String = this.getClass.getCanonicalName

override val statsReceiver: StatsReceiver = stats.scope(name)

override val stratoColumnPath: String = RealGraphOonFetcher.stratoColumnPath

override val stratoView: Unit = None

override protected val keyConv: Conv[UserId] = Conv.ofType

override protected val viewConv: Conv[Unit] = Conv.ofType

override protected val valueConv: Conv[CandidateSeq] =

ScroogeConv.fromStruct[CandidateSeq]

override protected def toStratoKey(userId: UserId): UserId = userId

override protected def toRawSignals(

realGraphOonCandidates: CandidateSeq

): Seq[RawSignalType] = realGraphOonCandidates.candidates

override def process(

query: Query,

rawSignals: Future[Option[Seq[RawSignalType]]]

): Future[Option[Seq[Signal]]] = {

rawSignals

.map {

\_.map(

\_.sortBy(-\_.score)

.collect {

case c if c.score >= MinRgScore =>

Signal(

SignalType.RealGraphOon,

RealGraphOonFetcher.DefaultTimestamp,

Some(InternalId.UserId(c.userId)))

}.take(query.maxResults.getOrElse(Int.MaxValue)))

}

}

}

object RealGraphOonFetcher {

val stratoColumnPath = "recommendations/real\_graph/realGraphScoresOon.User"

// quality threshold for real graph score

private val MinRgScore = 0.0

// no timestamp for RealGraph Candidates, set default as 0L

private val DefaultTimestamp = 0L

}