package com.twitter.follow\_recommendations.common.candidate\_sources.sims\_expansion

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.follow\_recommendations.common.candidate\_sources.sims.SwitchingSimsSource

import com.twitter.follow\_recommendations.common.clients.real\_time\_real\_graph.RealTimeRealGraphClient

import com.twitter.follow\_recommendations.common.models.AccountProof

import com.twitter.follow\_recommendations.common.models.CandidateUser

import com.twitter.follow\_recommendations.common.models.Reason

import com.twitter.follow\_recommendations.common.models.SimilarToProof

import com.twitter.hermit.model.Algorithm

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

import com.twitter.product\_mixer.core.model.marshalling.request.HasClientContext

import com.twitter.stitch.Stitch

import com.twitter.timelines.configapi.HasParams

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class RecentEngagementSimilarUsersSource @Inject() (

realTimeRealGraphClient: RealTimeRealGraphClient,

switchingSimsSource: SwitchingSimsSource,

statsReceiver: StatsReceiver)

extends SimsExpansionBasedCandidateSource[HasClientContext with HasParams](

switchingSimsSource) {

override def maxSecondaryDegreeNodes(req: HasClientContext with HasParams): Int = Int.MaxValue

override def maxResults(req: HasClientContext with HasParams): Int =

RecentEngagementSimilarUsersSource.MaxResults

override val identifier: CandidateSourceIdentifier = RecentEngagementSimilarUsersSource.Identifier

private val stats = statsReceiver.scope(identifier.name)

private val calibratedScoreCounter = stats.counter("calibrated\_scores\_counter")

override def scoreCandidate(sourceScore: Double, similarToScore: Double): Double = {

sourceScore \* similarToScore

}

override def calibrateDivisor(req: HasClientContext with HasParams): Double = {

req.params(DBV2SimsExpansionParams.RecentEngagementSimilarUsersDBV2CalibrateDivisor)

}

override def calibrateScore(

candidateScore: Double,

req: HasClientContext with HasParams

): Double = {

calibratedScoreCounter.incr()

candidateScore / calibrateDivisor(req)

}

/\*\*

\* fetch first degree nodes given request

\*/

override def firstDegreeNodes(

target: HasClientContext with HasParams

): Stitch[Seq[CandidateUser]] = {

target.getOptionalUserId

.map { userId =>

realTimeRealGraphClient

.getUsersRecentlyEngagedWith(

userId,

RealTimeRealGraphClient.EngagementScoreMap,

includeDirectFollowCandidates = true,

includeNonDirectFollowCandidates = true

).map(\_.sortBy(-\_.score.getOrElse(0.0d))

.take(RecentEngagementSimilarUsersSource.MaxFirstDegreeNodes))

}.getOrElse(Stitch.Nil)

}

override def aggregateAndScore(

request: HasClientContext with HasParams,

firstDegreeToSecondDegreeNodesMap: Map[CandidateUser, Seq[SimilarUser]]

): Stitch[Seq[CandidateUser]] = {

val inputNodes = firstDegreeToSecondDegreeNodesMap.keys.map(\_.id).toSet

val aggregator = request.params(RecentEngagementSimilarUsersParams.Aggregator) match {

case SimsExpansionSourceAggregatorId.Max =>

SimsExpansionBasedCandidateSource.ScoreAggregator.Max

case SimsExpansionSourceAggregatorId.Sum =>

SimsExpansionBasedCandidateSource.ScoreAggregator.Sum

case SimsExpansionSourceAggregatorId.MultiDecay =>

SimsExpansionBasedCandidateSource.ScoreAggregator.MultiDecay

}

val groupedCandidates = firstDegreeToSecondDegreeNodesMap.values.flatten

.filterNot(c => inputNodes.contains(c.candidateId))

.groupBy(\_.candidateId)

.map {

case (id, candidates) =>

// Different aggregators for final score

val finalScore = aggregator(candidates.map(\_.score).toSeq)

val proofs = candidates.map(\_.similarTo).toSet

CandidateUser(

id = id,

score = Some(finalScore),

reason =

Some(Reason(Some(AccountProof(similarToProof = Some(SimilarToProof(proofs.toSeq))))))

).withCandidateSource(identifier)

}

.toSeq

.sortBy(-\_.score.getOrElse(0.0d))

.take(maxResults(request))

Stitch.value(groupedCandidates)

}

}

object RecentEngagementSimilarUsersSource {

val Identifier = CandidateSourceIdentifier(Algorithm.RecentEngagementSimilarUser.toString)

val MaxFirstDegreeNodes = 10

val MaxResults = 200

}