package com.twitter.follow\_recommendations.common.candidate\_sources.stp

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.follow\_recommendations.common.candidate\_sources.stp.OnlineSTPSourceParams.SetPredictionDetails

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

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

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

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

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

import com.twitter.onboarding.relevance.features.strongtie.{

StrongTieFeatures => StrongTieFeaturesWrapper

}

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

import com.twitter.stitch.Stitch

import com.twitter.timelines.configapi.HasParams

import com.twitter.util.logging.Logging

import com.twitter.wtf.scalding.jobs.strong\_tie\_prediction.STPRecord

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

class OnlineSTPSourceWithEPScorer @Inject() (

epStpScorer: EpStpScorer,

stpGraphBuilder: STPGraphBuilder,

baseStatReceiver: StatsReceiver)

extends BaseOnlineSTPSource(stpGraphBuilder, baseStatReceiver)

with Logging {

private val epScorerUsedCounter = statsReceiver.counter("ep\_scorer\_used")

override def getCandidates(

records: Seq[STPRecord],

request: HasClientContext with HasParams with HasRecentFollowedUserIds,

): Stitch[Seq[CandidateUser]] = {

epScorerUsedCounter.incr()

val possibleCandidates: Seq[Stitch[Option[CandidateUser]]] = records.map { trainingRecord =>

val scoredResponse =

epStpScorer.getScoredResponse(trainingRecord.record, request.params(SetPredictionDetails))

scoredResponse.map(\_.map { response: ScoredResponse =>

logger.debug(response)

CandidateUser(

id = trainingRecord.destinationId,

score = Some(response.score),

reason = Some(

Reason(

Some(

AccountProof(followProof =

Some(FollowProof(trainingRecord.socialProof, trainingRecord.socialProof.size)))

)))

).withCandidateSourceAndFeatures(

identifier,

Seq(StrongTieFeaturesWrapper(trainingRecord.features)))

})

}

Stitch.collect(possibleCandidates).map { \_.flatten.sortBy(-\_.score.getOrElse(0.0)) }

}

}