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

import com.google.inject.Singleton

import com.twitter.follow\_recommendations.common.candidate\_sources.stp.OfflineStpSourceParams.UseDenserPmiMatrix

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

import com.twitter.hermit.model.Algorithm

import com.twitter.product\_mixer.component\_library.model.candidate.UserCandidate

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

import com.twitter.util.logging.Logging

import com.twitter.product\_mixer.core.functional\_component.candidate\_source.CandidateSource

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

object OfflineStpScore extends Feature[UserCandidate, Option[Double]]

/\*\*

\* Main source for strong-tie-prediction candidates generated offline.

\*/

@Singleton

class OfflineStrongTiePredictionSource @Inject() (

offlineStpSourceWithLegacyPmiMatrix: OfflineStpSourceWithLegacyPmiMatrix,

offlineStpSourceWithDensePmiMatrix: OfflineStpSourceWithDensePmiMatrix)

extends CandidateSource[HasParams with HasClientContext, CandidateUser]

with Logging {

override val identifier: CandidateSourceIdentifier = OfflineStrongTiePredictionSource.Identifier

override def apply(request: HasParams with HasClientContext): Stitch[Seq[CandidateUser]] = {

if (request.params(UseDenserPmiMatrix)) {

logger.info("Using dense PMI matrix.")

offlineStpSourceWithDensePmiMatrix(request)

} else {

logger.info("Using legacy PMI matrix.")

offlineStpSourceWithLegacyPmiMatrix(request)

}

}

}

object OfflineStrongTiePredictionSource {

val Identifier: CandidateSourceIdentifier =

CandidateSourceIdentifier(Algorithm.StrongTiePredictionRec.toString)

}