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

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

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

import com.twitter.stitch.Stitch

abstract class SalsaExpansionBasedCandidateSource[Target](salsaExpander: SalsaExpander)

extends CandidateSource[Target, CandidateUser] {

// Define first/second degree as empty sequences in cases of subclasses

// that don't implement one or the other.

// Example: MagicRecs only uses first degree nodes, and can ignore implementing secondDegreeNodes

//

// This allows apply(target) to combine both in the base class

def firstDegreeNodes(target: Target): Stitch[Seq[Long]] = Stitch.value(Seq())

def secondDegreeNodes(target: Target): Stitch[Seq[Long]] = Stitch.value(Seq())

// max number output results

def maxResults(target: Target): Int

override def apply(target: Target): Stitch[Seq[CandidateUser]] = {

val nodes = Stitch.join(firstDegreeNodes(target), secondDegreeNodes(target))

nodes.flatMap {

case (firstDegreeCandidates, secondDegreeCandidates) => {

salsaExpander(firstDegreeCandidates, secondDegreeCandidates, maxResults(target))

.map(\_.map(\_.withCandidateSource(identifier)).sortBy(-\_.score.getOrElse(0.0)))

}

}

}

}