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

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

import com.twitter.stitch.Stitch

/\*\*

\* base trait for two-hop expansion based algorithms, e.g. online\_stp, phonebook\_prediction,

\* recent following sims, recent engagement sims, ...

\*

\* @tparam Target target type

\* @tparam FirstDegree type of first degree nodes

\* @tparam SecondaryDegree type of secondary degree nodes

\* @tparam Candidate output candidate types

\*/

trait TwoHopExpansionCandidateSource[-Target, FirstDegree, SecondaryDegree, +Candidate]

extends CandidateSource[Target, Candidate] {

/\*\*

\* fetch first degree nodes given request

\*/

def firstDegreeNodes(req: Target): Stitch[Seq[FirstDegree]]

/\*\*

\* fetch secondary degree nodes given request and first degree nodes

\*/

def secondaryDegreeNodes(req: Target, node: FirstDegree): Stitch[Seq[SecondaryDegree]]

/\*\*

\* aggregate and score the candidates to generate final results

\*/

def aggregateAndScore(

req: Target,

firstDegreeToSecondDegreeNodesMap: Map[FirstDegree, Seq[SecondaryDegree]]

): Stitch[Seq[Candidate]]

/\*\*

\* Generate a list of candidates for the target

\*/

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

for {

firstDegreeNodes <- firstDegreeNodes(target)

secondaryDegreeNodes <- Stitch.traverse(firstDegreeNodes)(secondaryDegreeNodes(target, \_))

aggregated <- aggregateAndScore(target, firstDegreeNodes.zip(secondaryDegreeNodes).toMap)

} yield aggregated

}

}