package com.twitter.recos.user\_tweet\_graph

import com.twitter.recos.graph\_common.MultiSegmentPowerLawBipartiteGraphBuilder.GraphBuilderConfig

/\*\*

\* The class holds all the config parameters for recos graph.

\*/

object RecosConfig {

val maxNumSegments: Int = 8

val maxNumEdgesPerSegment: Int =

(1 << 28) // 268M edges per segment, should be able to include 2 days' data

val expectedNumLeftNodes: Int =

(1 << 26) // should correspond to 67M nodes storage

val expectedMaxLeftDegree: Int = 64

val leftPowerLawExponent: Double = 16.0 // steep power law as most nodes will have a small degree

val expectedNumRightNodes: Int = (1 << 26) // 67M nodes

val expectedMaxRightDegree: Int = scala.math.pow(1024, 2).toInt // some nodes will be very popular

val rightPowerLawExponent: Double = 4.0 // this will be less steep

val graphBuilderConfig = GraphBuilderConfig(

maxNumSegments = maxNumSegments,

maxNumEdgesPerSegment = maxNumEdgesPerSegment,

expectedNumLeftNodes = expectedNumLeftNodes,

expectedMaxLeftDegree = expectedMaxLeftDegree,

leftPowerLawExponent = leftPowerLawExponent,

expectedNumRightNodes = expectedNumRightNodes,

expectedMaxRightDegree = expectedMaxRightDegree,

rightPowerLawExponent = rightPowerLawExponent

)

println("RecosConfig - maxNumSegments " + maxNumSegments)

println("RecosConfig - maxNumEdgesPerSegment " + maxNumEdgesPerSegment)

println("RecosConfig - expectedNumLeftNodes " + expectedNumLeftNodes)

println("RecosConfig - expectedMaxLeftDegree " + expectedMaxLeftDegree)

println("RecosConfig - leftPowerLawExponent " + leftPowerLawExponent)

println("RecosConfig - expectedNumRightNodes " + expectedNumRightNodes)

println("RecosConfig - expectedMaxRightDegree " + expectedMaxRightDegree)

println("RecosConfig - rightPowerLawExponent " + rightPowerLawExponent)

}