package com.twitter.graph.batch.job.tweepcred

/\*\*

\* helper class to calculate reputation, borrowed from repo reputations

\*/

object Reputation {

/\*\*

\* convert pagerank to tweepcred between 0 and 100,

\* take from repo reputations, util/Utils.scala

\*/

def scaledReputation(raw: Double): Byte = {

if (raw == 0 || (raw < 1.0e-20)) {

0

} else {

// convert log(pagerank) to a number between 0 and 100

// the two parameters are from a linear fit by converting

// max pagerank -> 95

// min pagerank -> 15

val e: Double = 130d + 5.21 \* scala.math.log(raw) // log to the base e

val pos = scala.math.rint(e)

val v = if (pos > 100) 100.0 else if (pos < 0) 0.0 else pos

v.toByte

}

}

// these constants are take from repo reputations, config/production.conf

private val threshAbsNumFriendsReps = 2500

private val constantDivisionFactorGt\_threshFriendsToFollowersRatioReps = 3.0

private val threshFriendsToFollowersRatioUMass = 0.6

private val maxDivFactorReps = 50

/\*\*

\* reduce pagerank of users with low followers but high followings

\*/

def adjustReputationsPostCalculation(mass: Double, numFollowers: Int, numFollowings: Int) = {

if (numFollowings > threshAbsNumFriendsReps) {

val friendsToFollowersRatio = (1.0 + numFollowings) / (1.0 + numFollowers)

val divFactor =

scala.math.exp(

constantDivisionFactorGt\_threshFriendsToFollowersRatioReps \*

(friendsToFollowersRatio - threshFriendsToFollowersRatioUMass) \*

scala.math.log(scala.math.log(numFollowings))

)

mass / ((divFactor min maxDivFactorReps) max 1.0)

} else {

mass

}

}

}