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

import com.twitter.finagle.Service

import com.twitter.finagle.http.Request

import com.twitter.finagle.http.Response

import com.twitter.finagle.http.Status

import com.twitter.finagle.http.Version

import com.twitter.frigate.common.util.HTMLUtil

import com.twitter.graphjet.algorithms.TweetIDMask

import com.twitter.graphjet.bipartite.segment.BipartiteGraphSegment

import com.twitter.graphjet.bipartite.MultiSegmentIterator

import com.twitter.graphjet.bipartite.MultiSegmentPowerLawBipartiteGraph

import com.twitter.logging.Logger

import com.twitter.util.Future

import java.util.Random

import scala.collection.mutable.ListBuffer

class UserTweetGraphEdgeHttpHandler(graph: MultiSegmentPowerLawBipartiteGraph)

extends Service[Request, Response] {

private val log = Logger("UserTweetGraphEdgeHttpHandler")

private val tweetIDMask = new TweetIDMask()

def getCardInfo(rightNode: Long): String = {

val bits: Long = rightNode & TweetIDMask.METAMASK

bits match {

case TweetIDMask.PHOTO => "Photo"

case TweetIDMask.PLAYER => "Video"

case TweetIDMask.SUMMARY => "Url"

case TweetIDMask.PROMOTION => "Promotion"

case \_ => "Regular"

}

}

private def getUserEdges(userId: Long): ListBuffer[Edge] = {

val random = new Random()

val iterator =

graph

.getRandomLeftNodeEdges(userId, 10, random).asInstanceOf[MultiSegmentIterator[

BipartiteGraphSegment

]]

val tweets = new ListBuffer[Edge]()

if (iterator != null) {

while (iterator.hasNext) {

val rightNode = iterator.nextLong()

val edgeType = iterator.currentEdgeType()

tweets += Edge(

tweetIDMask.restore(rightNode),

UserVideoEdgeTypeMask(edgeType).toString,

getCardInfo(rightNode),

)

}

}

tweets

}

def apply(httpRequest: Request): Future[Response] = {

log.info("UserTweetGraphEdgeHttpHandler params: " + httpRequest.getParams())

val time0 = System.currentTimeMillis

val tweetId = httpRequest.getLongParam("tweetId")

val queryTweetDegree = graph.getRightNodeDegree(tweetId)

val tweetEdges = getTweetEdges(tweetId)

val userId = httpRequest.getLongParam("userId")

val queryUserDegree = graph.getLeftNodeDegree(userId)

val response = Response(Version.Http11, Status.Ok)

val userEdges = getUserEdges(userId)

val elapsed = System.currentTimeMillis - time0

val comment = ("Please specify \"userId\" or \"tweetId\" param." +

"\n query tweet degree = " + queryTweetDegree +

"\n query user degree = " + queryUserDegree +

"\n done in %d ms<br>").format(elapsed)

val tweetContent = userEdges.toList

.map { edge =>

s"<b>TweetId</b>: ${edge.tweetId},\n<b>Action type</b>: ${edge.actionType},\n<b>Card type</b>: ${edge.cardType}"

.replaceAll("\n", " ")

}.mkString("\n<br>\n")

response.setContentString(

HTMLUtil.html.replace("XXXXX", comment + tweetContent + "\n<hr/>\n" + tweetEdges.toString()))

Future.value(response)

}

private def getTweetEdges(tweetId: Long): ListBuffer[Long] = {

val random = new Random()

val iterator =

graph

.getRandomRightNodeEdges(tweetId, 500, random).asInstanceOf[MultiSegmentIterator[

BipartiteGraphSegment

]]

val terms = new ListBuffer[Long]()

if (iterator != null) {

while (iterator.hasNext) { terms += iterator.nextLong() }

}

terms.distinct

}

}

case class Edge(tweetId: Long, actionType: String, cardType: String)