package com.twitter.recosinjector.edges

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.recos.util.Action

import com.twitter.recosinjector.util.UuaEngagementEventDetails

import com.twitter.util.Future

class UnifiedUserActionToUserVideoGraphBuilder(

userTweetEntityEdgeBuilder: UserTweetEntityEdgeBuilder

)(

override implicit val statsReceiver: StatsReceiver)

extends EventToMessageBuilder[UuaEngagementEventDetails, UserTweetEntityEdge] {

private val numVideoPlayback50EdgeCounter = statsReceiver.counter("num\_video\_playback50\_edge")

private val numUnVideoPlayback50Counter = statsReceiver.counter("num\_non\_video\_playback50\_edge")

override def shouldProcessEvent(event: UuaEngagementEventDetails): Future[Boolean] = {

event.userTweetEngagement.action match {

case Action.VideoPlayback50 => Future(true)

case \_ => Future(false)

}

}

override def buildEdges(details: UuaEngagementEventDetails): Future[Seq[UserTweetEntityEdge]] = {

val engagement = details.userTweetEngagement

val tweetDetails = engagement.tweetDetails

Future

.value(

UserTweetEntityEdge(

sourceUser = engagement.engageUserId,

targetTweet = engagement.tweetId,

action = engagement.action,

metadata = engagement.engagementTimeMillis,

cardInfo = engagement.tweetDetails.map(\_.cardInfo.toByte),

entitiesMap = None,

tweetDetails = tweetDetails

)

).map { edge =>

edge match {

case videoPlayback50 if videoPlayback50.action == Action.VideoPlayback50 =>

numVideoPlayback50EdgeCounter.incr()

case \_ =>

numUnVideoPlayback50Counter.incr()

}

Seq(edge)

}

}

override def filterEdges(

event: UuaEngagementEventDetails,

edges: Seq[UserTweetEntityEdge]

): Future[Seq[UserTweetEntityEdge]] = {

Future(edges)

}

}