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 UnifiedUserActionToUserTweetGraphPlusBuilder(

userTweetEntityEdgeBuilder: UserTweetEntityEdgeBuilder

)(

override implicit val statsReceiver: StatsReceiver)

extends EventToMessageBuilder[UuaEngagementEventDetails, UserTweetEntityEdge] {

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

event.userTweetEngagement.action match {

case Action.Click | Action.VideoQualityView => Future(true)

case Action.Favorite | Action.Retweet | Action.Share => Future(true)

case Action.NotificationOpen | Action.EmailClick => Future(true)

case Action.Quote | Action.Reply => Future(true)

case Action.TweetNotInterestedIn | Action.TweetNotRelevant | Action.TweetSeeFewer |

Action.TweetReport | Action.TweetMuteAuthor | Action.TweetBlockAuthor =>

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(Seq(\_))

}

override def filterEdges(

event: UuaEngagementEventDetails,

edges: Seq[UserTweetEntityEdge]

): Future[Seq[UserTweetEntityEdge]] = {

Future(edges)

}

}