package com.twitter.recosinjector.edges

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.recos.util.Action

import com.twitter.recosinjector.util.TweetCreateEventDetails

import com.twitter.util.Future

/\*\*

\* Given a tweet creation event, parse for UserUserGraph edges. Specifically, when a new tweet is

\* created, extract the valid mentioned and mediatagged users in the tweet and create edges for them

\*/

class TweetEventToUserUserGraphBuilder(

)(

override implicit val statsReceiver: StatsReceiver)

extends EventToMessageBuilder[TweetCreateEventDetails, UserUserEdge] {

private val tweetOrQuoteEventCounter = statsReceiver.counter("num\_tweet\_or\_quote\_event")

private val nonTweetOrQuoteEventCounter = statsReceiver.counter("num\_non\_tweet\_or\_quote\_event")

private val mentionEdgeCounter = statsReceiver.counter("num\_mention\_edge")

private val mediatagEdgeCounter = statsReceiver.counter("num\_mediatag\_edge")

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

// For user interactions, only new tweets and quotes are considered (no replies or retweets)

event.userTweetEngagement.action match {

case Action.Tweet | Action.Quote =>

tweetOrQuoteEventCounter.incr()

Future(true)

case \_ =>

nonTweetOrQuoteEventCounter.incr()

Future(false)

}

}

override def buildEdges(event: TweetCreateEventDetails): Future[Seq[UserUserEdge]] = {

val mentionEdges = event.validMentionUserIds

.map(\_.map { mentionUserId =>

UserUserEdge(

sourceUser = event.userTweetEngagement.engageUserId,

targetUser = mentionUserId,

action = Action.Mention,

metadata = Some(System.currentTimeMillis())

)

}).getOrElse(Nil)

val mediatagEdges = event.validMediatagUserIds

.map(\_.map { mediatagUserId =>

UserUserEdge(

sourceUser = event.userTweetEngagement.engageUserId,

targetUser = mediatagUserId,

action = Action.MediaTag,

metadata = Some(System.currentTimeMillis())

)

}).getOrElse(Nil)

mentionEdgeCounter.incr(mentionEdges.size)

mediatagEdgeCounter.incr(mediatagEdges.size)

Future(mentionEdges ++ mediatagEdges)

}

override def filterEdges(

event: TweetCreateEventDetails,

edges: Seq[UserUserEdge]

): Future[Seq[UserUserEdge]] = {

Future(edges)

}

}