package com.twitter.visibility.builder.tweets

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.notificationservice.model.notification.ActivityNotification

import com.twitter.notificationservice.model.notification.MentionNotification

import com.twitter.notificationservice.model.notification.MentionQuoteNotification

import com.twitter.notificationservice.model.notification.Notification

import com.twitter.notificationservice.model.notification.QuoteTweetNotification

import com.twitter.servo.util.Gate

import com.twitter.stitch.Stitch

import com.twitter.visibility.builder.FeatureMapBuilder

import com.twitter.visibility.common.TweetSource

import com.twitter.visibility.features.NotificationIsOnCommunityTweet

import com.twitter.visibility.models.CommunityTweet

object CommunityNotificationFeatures {

def ForNonCommunityTweetNotification: FeatureMapBuilder => FeatureMapBuilder = {

\_.withConstantFeature(NotificationIsOnCommunityTweet, false)

}

}

class CommunityNotificationFeatures(

tweetSource: TweetSource,

enableCommunityTweetHydration: Gate[Long],

statsReceiver: StatsReceiver) {

private[this] val scopedStatsReceiver = statsReceiver.scope("community\_notification\_features")

private[this] val requestsCounter = scopedStatsReceiver.counter("requests")

private[this] val hydrationsCounter = scopedStatsReceiver.counter("hydrations")

private[this] val notificationIsOnCommunityTweetCounter =

scopedStatsReceiver.scope(NotificationIsOnCommunityTweet.name).counter("true")

private[this] val notificationIsNotOnCommunityTweetCounter =

scopedStatsReceiver.scope(NotificationIsOnCommunityTweet.name).counter("false")

def forNotification(notification: Notification): FeatureMapBuilder => FeatureMapBuilder = {

requestsCounter.incr()

val isCommunityTweetResult = getTweetIdOption(notification) match {

case Some(tweetId) if enableCommunityTweetHydration(notification.target) =>

hydrationsCounter.incr()

tweetSource

.getTweet(tweetId)

.map {

case Some(tweet) if CommunityTweet(tweet).nonEmpty =>

notificationIsOnCommunityTweetCounter.incr()

true

case \_ =>

notificationIsNotOnCommunityTweetCounter.incr()

false

}

case \_ => Stitch.False

}

\_.withFeature(NotificationIsOnCommunityTweet, isCommunityTweetResult)

}

private[this] def getTweetIdOption(notification: Notification): Option[Long] = {

notification match {

case n: MentionNotification => Some(n.mentioningTweetId)

case n: MentionQuoteNotification => Some(n.mentioningTweetId)

case n: QuoteTweetNotification => Some(n.quotedTweetId)

case n: ActivityNotification[\_] if n.visibilityTweets.contains(n.objectId) => Some(n.objectId)

case \_ => None

}

}

}