package com.twitter.visibility.builder.tweets

import com.twitter.contenthealth.toxicreplyfilter.thriftscala.FilterState

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.tweetypie.thriftscala.Tweet

import com.twitter.visibility.builder.FeatureMapBuilder

import com.twitter.visibility.features.ToxicReplyFilterConversationAuthorIsViewer

import com.twitter.visibility.features.ToxicReplyFilterState

class ToxicReplyFilterFeature(

statsReceiver: StatsReceiver) {

def forTweet(tweet: Tweet, viewerId: Option[Long]): FeatureMapBuilder => FeatureMapBuilder = {

builder =>

requests.incr()

builder

.withConstantFeature(ToxicReplyFilterState, isTweetFilteredFromAuthor(tweet))

.withConstantFeature(

ToxicReplyFilterConversationAuthorIsViewer,

isRootAuthorViewer(tweet, viewerId))

}

private[this] def isRootAuthorViewer(tweet: Tweet, maybeViewerId: Option[Long]): Boolean = {

val maybeAuthorId = tweet.filteredReplyDetails.map(\_.conversationAuthorId)

(maybeViewerId, maybeAuthorId) match {

case (Some(viewerId), Some(authorId)) if viewerId == authorId => {

rootAuthorViewerStats.incr()

true

}

case \_ => false

}

}

private[this] def isTweetFilteredFromAuthor(

tweet: Tweet,

): FilterState = {

val result = tweet.filteredReplyDetails.map(\_.filterState).getOrElse(FilterState.Unfiltered)

if (result == FilterState.FilteredFromAuthor) {

filteredFromAuthorStats.incr()

}

result

}

private[this] val scopedStatsReceiver =

statsReceiver.scope("toxicreplyfilter")

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

private[this] val rootAuthorViewerStats =

scopedStatsReceiver.scope(ToxicReplyFilterConversationAuthorIsViewer.name).counter("requests")

private[this] val filteredFromAuthorStats =

scopedStatsReceiver.scope(ToxicReplyFilterState.name).counter("requests")

}