package com.twitter.visibility.builder.tweets

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.servo.util.Gate

import com.twitter.spam.rtf.thriftscala.SafetyLabel

import com.twitter.spam.rtf.thriftscala.SafetyLabelType

import com.twitter.spam.rtf.thriftscala.SafetyLabelValue

import com.twitter.stitch.Stitch

import com.twitter.visibility.builder.FeatureMapBuilder

import com.twitter.visibility.common.stitch.StitchHelpers

import com.twitter.visibility.features.TweetId

import com.twitter.visibility.features.TweetSafetyLabels

import com.twitter.visibility.features.TweetTimestamp

import com.twitter.visibility.models.TweetSafetyLabel

class TweetIdFeatures(

statsReceiver: StatsReceiver,

enableStitchProfiling: Gate[Unit]) {

private[this] val scopedStatsReceiver: StatsReceiver = statsReceiver.scope("tweet\_id\_features")

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

private[this] val tweetSafetyLabels =

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

private[this] val tweetTimestamp =

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

private[this] val labelFetchScope: StatsReceiver =

scopedStatsReceiver.scope("labelFetch")

private[this] def getTweetLabels(

tweetId: Long,

labelFetcher: Long => Stitch[Map[SafetyLabelType, SafetyLabel]]

): Stitch[Seq[TweetSafetyLabel]] = {

val stitch =

labelFetcher(tweetId).map { labelMap =>

labelMap

.map { case (labelType, label) => SafetyLabelValue(labelType, label) }.toSeq

.map(TweetSafetyLabel.fromThrift)

}

if (enableStitchProfiling()) {

StitchHelpers.profileStitch(

stitch,

Seq(labelFetchScope)

)

} else {

stitch

}

}

def forTweetId(

tweetId: Long,

labelFetcher: Long => Stitch[Map[SafetyLabelType, SafetyLabel]]

): FeatureMapBuilder => FeatureMapBuilder = {

requests.incr()

tweetSafetyLabels.incr()

tweetTimestamp.incr()

\_.withFeature(TweetSafetyLabels, getTweetLabels(tweetId, labelFetcher))

.withConstantFeature(TweetTimestamp, TweetFeatures.tweetTimestamp(tweetId))

.withConstantFeature(TweetId, tweetId)

}

def forTweetId(

tweetId: Long,

constantTweetSafetyLabels: Seq[TweetSafetyLabel]

): FeatureMapBuilder => FeatureMapBuilder = {

requests.incr()

tweetSafetyLabels.incr()

tweetTimestamp.incr()

\_.withConstantFeature(TweetSafetyLabels, constantTweetSafetyLabels)

.withConstantFeature(TweetTimestamp, TweetFeatures.tweetTimestamp(tweetId))

.withConstantFeature(TweetId, tweetId)

}

}