package com.twitter.cr\_mixer.util

import com.twitter.cr\_mixer.model.CandidateGenerationInfo

import com.twitter.cr\_mixer.model.RankedCandidate

import com.twitter.cr\_mixer.model.SourceInfo

import com.twitter.cr\_mixer.thriftscala.SourceType

import com.twitter.cr\_mixer.thriftscala.TweetRecommendation

import javax.inject.Inject

import com.twitter.finagle.stats.StatsReceiver

import javax.inject.Singleton

import com.twitter.relevance\_platform.common.stats.BucketTimestampStats

@Singleton

class SignalTimestampStatsUtil @Inject() (statsReceiver: StatsReceiver) {

import SignalTimestampStatsUtil.\_

private val signalDelayAgePerDayStats =

new BucketTimestampStats[TweetRecommendation](

BucketTimestampStats.MillisecondsPerDay,

\_.latestSourceSignalTimestampInMillis.getOrElse(0),

Some(SignalTimestampMaxDays))(

statsReceiver.scope("signal\_timestamp\_per\_day")

) // only stats past 90 days

private val signalDelayAgePerHourStats =

new BucketTimestampStats[TweetRecommendation](

BucketTimestampStats.MillisecondsPerHour,

\_.latestSourceSignalTimestampInMillis.getOrElse(0),

Some(SignalTimestampMaxHours))(

statsReceiver.scope("signal\_timestamp\_per\_hour")

) // only stats past 24 hours

private val signalDelayAgePerMinStats =

new BucketTimestampStats[TweetRecommendation](

BucketTimestampStats.MillisecondsPerMinute,

\_.latestSourceSignalTimestampInMillis.getOrElse(0),

Some(SignalTimestampMaxMins))(

statsReceiver.scope("signal\_timestamp\_per\_min")

) // only stats past 60 minutes

def statsSignalTimestamp(

tweets: Seq[TweetRecommendation],

): Seq[TweetRecommendation] = {

signalDelayAgePerMinStats.count(tweets)

signalDelayAgePerHourStats.count(tweets)

signalDelayAgePerDayStats.count(tweets)

}

}

object SignalTimestampStatsUtil {

val SignalTimestampMaxMins = 60 // stats at most 60 mins

val SignalTimestampMaxHours = 24 // stats at most 24 hours

val SignalTimestampMaxDays = 90 // stats at most 90 days

def buildLatestSourceSignalTimestamp(candidate: RankedCandidate): Option[Long] = {

val timestampSeq = candidate.potentialReasons

.collect {

case CandidateGenerationInfo(Some(SourceInfo(sourceType, \_, Some(sourceEventTime))), \_, \_)

if sourceType == SourceType.TweetFavorite =>

sourceEventTime.inMilliseconds

}

if (timestampSeq.nonEmpty) {

Some(timestampSeq.max(Ordering.Long))

} else {

None

}

}

}