package com.twitter.frigate.pushservice.util

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.scio.nsfw\_user\_segmentation.thriftscala.NSFWUserSegmentation

object NsfwPersonalizationUtil {

def computeNsfwUserStats(

targetNsfwInfo: Option[NsfwInfo]

)(

implicit statsReceiver: StatsReceiver

): Unit = {

def computeNsfwProfileVisitStats(sReceiver: StatsReceiver, nsfwProfileVisits: Long): Unit = {

if (nsfwProfileVisits >= 1)

sReceiver.counter("nsfwProfileVisits\_gt\_1").incr()

if (nsfwProfileVisits >= 2)

sReceiver.counter("nsfwProfileVisits\_gt\_2").incr()

if (nsfwProfileVisits >= 3)

sReceiver.counter("nsfwProfileVisits\_gt\_3").incr()

if (nsfwProfileVisits >= 5)

sReceiver.counter("nsfwProfileVisits\_gt\_5").incr()

if (nsfwProfileVisits >= 8)

sReceiver.counter("nsfwProfileVisits\_gt\_8").incr()

}

def computeRatioStats(

sReceiver: StatsReceiver,

ratio: Int,

statName: String,

intervals: List[Double] = List(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9)

): Unit = {

intervals.foreach { i =>

if (ratio > i \* 10000)

sReceiver.counter(f"${statName}\_greater\_than\_${i}").incr()

}

}

val sReceiver = statsReceiver.scope("nsfw\_personalization")

sReceiver.counter("AllUsers").incr()

(targetNsfwInfo) match {

case (Some(nsfwInfo)) =>

val sensitive = nsfwInfo.senstiveOptIn.getOrElse(false)

val nsfwFollowRatio =

nsfwInfo.nsfwFollowRatio

val totalFollows = nsfwInfo.totalFollowCount

val numNsfwProfileVisits = nsfwInfo.nsfwProfileVisits

val nsfwRealGraphScore = nsfwInfo.realGraphScore

val nsfwSearchScore = nsfwInfo.searchNsfwScore

val totalSearches = nsfwInfo.totalSearches

val realGraphScore = nsfwInfo.realGraphScore

val searchScore = nsfwInfo.searchNsfwScore

if (sensitive)

sReceiver.counter("sensitiveOptInEnabled").incr()

else

sReceiver.counter("sensitiveOptInDisabled").incr()

computeRatioStats(sReceiver, nsfwFollowRatio, "nsfwRatio")

computeNsfwProfileVisitStats(sReceiver, numNsfwProfileVisits)

computeRatioStats(sReceiver, nsfwRealGraphScore.toInt, "nsfwRealGraphScore")

if (totalSearches >= 10)

computeRatioStats(sReceiver, nsfwSearchScore.toInt, "nsfwSearchScore")

if (searchScore == 0)

sReceiver.counter("lowSearchScore").incr()

if (realGraphScore < 500)

sReceiver.counter("lowRealScore").incr()

if (numNsfwProfileVisits == 0)

sReceiver.counter("lowProfileVisit").incr()

if (nsfwFollowRatio == 0)

sReceiver.counter("lowFollowScore").incr()

if (totalSearches > 10 && searchScore > 5000)

sReceiver.counter("highSearchScore").incr()

if (realGraphScore > 7000)

sReceiver.counter("highRealScore").incr()

if (numNsfwProfileVisits > 5)

sReceiver.counter("highProfileVisit").incr()

if (totalFollows > 10 && nsfwFollowRatio > 7000)

sReceiver.counter("highFollowScore").incr()

if (searchScore == 0 && realGraphScore <= 500 && numNsfwProfileVisits == 0 && nsfwFollowRatio == 0)

sReceiver.counter("lowIntent").incr()

if ((totalSearches > 10 && searchScore > 5000) || realGraphScore > 7000 || numNsfwProfileVisits > 5 || (totalFollows > 10 && nsfwFollowRatio > 7000))

sReceiver.counter("highIntent").incr()

case \_ =>

}

}

}

case class NsfwInfo(nsfwUserSegmentation: NSFWUserSegmentation) {

val scalingFactor = 10000 // to convert float to int as custom fields cannot be float

val senstiveOptIn: Option[Boolean] = nsfwUserSegmentation.nsfwView

val totalFollowCount: Long = nsfwUserSegmentation.totalFollowCnt.getOrElse(0L)

val nsfwFollowCnt: Long =

nsfwUserSegmentation.nsfwAdminOrHighprecOrAgathaGtP98FollowsCnt.getOrElse(0L)

val nsfwFollowRatio: Int = {

if (totalFollowCount != 0) {

(nsfwFollowCnt \* scalingFactor / totalFollowCount).toInt

} else 0

}

val nsfwProfileVisits: Long =

nsfwUserSegmentation.nsfwAdminOrHighPrecOrAgathaGtP98Visits

.map(\_.numProfilesInLast14Days).getOrElse(0L)

val realGraphScore: Int =

nsfwUserSegmentation.realGraphMetrics

.map { rm =>

if (rm.totalOutboundRGScore != 0)

rm.totalNsfwAdmHPAgthGtP98OutboundRGScore \* scalingFactor / rm.totalOutboundRGScore

else 0d

}.getOrElse(0d).toInt

val totalSearches: Long =

nsfwUserSegmentation.searchMetrics.map(\_.numNonTrndSrchInLast14Days).getOrElse(0L)

val searchNsfwScore: Int = nsfwUserSegmentation.searchMetrics

.map { sm =>

if (sm.numNonTrndNonHshtgSrchInLast14Days != 0)

sm.numNonTrndNonHshtgGlobalNsfwSrchInLast14Days.toDouble \* scalingFactor / sm.numNonTrndNonHshtgSrchInLast14Days

else 0

}.getOrElse(0d).toInt

val hasReported: Boolean =

nsfwUserSegmentation.notifFeedbackMetrics.exists(\_.notifReportMetrics.exists(\_.countTotal != 0))

val hasDisliked: Boolean =

nsfwUserSegmentation.notifFeedbackMetrics

.exists(\_.notifDislikeMetrics.exists(\_.countTotal != 0))

}