package com.twitter.frigate.pushservice.util

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.base.CandidateDetails

import com.twitter.frigate.pushservice.model.PushTypes.PushCandidate

object MediaAnnotationsUtil {

val mediaIdToCategoryMapping = Map("0" -> "0")

val nudityCategoryId = "0"

val beautyCategoryId = "0"

val singlePersonCategoryId = "0"

val sensitiveMediaCategoryFeatureName =

"tweet.mediaunderstanding.tweet\_annotations.sensitive\_category\_probabilities"

def updateMediaCategoryStats(

candidates: Seq[CandidateDetails[PushCandidate]]

)(

implicit statsReceiver: StatsReceiver

) = {

val statScope = statsReceiver.scope("mediaStats")

val filteredCandidates = candidates.filter { candidate =>

!candidate.candidate.sparseContinuousFeatures

.getOrElse(sensitiveMediaCategoryFeatureName, Map.empty[String, Double]).contains(

nudityCategoryId)

}

if (filteredCandidates.isEmpty)

statScope.counter("emptyCandidateListAfterNudityFilter").incr()

else

statScope.counter("nonEmptyCandidateListAfterNudityFilter").incr()

candidates.foreach { candidate =>

statScope.counter("totalCandidates").incr()

val mediaFeature = candidate.candidate.sparseContinuousFeatures

.getOrElse(sensitiveMediaCategoryFeatureName, Map.empty[String, Double])

if (mediaFeature.nonEmpty) {

val mediaCategoryByMaxScore = mediaFeature.maxBy(\_.\_2).\_1

statScope

.scope("mediaCategoryByMaxScore").counter(mediaIdToCategoryMapping

.getOrElse(mediaCategoryByMaxScore, "undefined")).incr()

mediaFeature.keys.map { feature =>

statScope

.scope("mediaCategory").counter(mediaIdToCategoryMapping

.getOrElse(feature, "undefined")).incr()

}

}

}

}

}