package com.twitter.frigate.pushservice.refresh\_handler

import com.twitter.finagle.stats.Stat

import com.twitter.finagle.stats.StatsReceiver

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

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

import com.twitter.frigate.thriftscala.CommonRecommendationType

class RFPHStatsRecorder(implicit statsReceiver: StatsReceiver) {

private val selectedCandidateScoreStats: StatsReceiver =

statsReceiver.scope("score\_of\_sent\_candidate\_times\_10000")

private val emptyScoreStats: StatsReceiver =

statsReceiver.scope("score\_of\_sent\_candidate\_empty")

def trackPredictionScoreStats(candidate: PushCandidate): Unit = {

candidate.mrWeightedOpenOrNtabClickRankingProbability.foreach {

case Some(s) =>

selectedCandidateScoreStats

.stat("weighted\_open\_or\_ntab\_click\_ranking")

.add((s \* 10000).toFloat)

case None =>

emptyScoreStats.counter("weighted\_open\_or\_ntab\_click\_ranking").incr()

}

candidate.mrWeightedOpenOrNtabClickFilteringProbability.foreach {

case Some(s) =>

selectedCandidateScoreStats

.stat("weighted\_open\_or\_ntab\_click\_filtering")

.add((s \* 10000).toFloat)

case None =>

emptyScoreStats.counter("weighted\_open\_or\_ntab\_click\_filtering").incr()

}

candidate.mrWeightedOpenOrNtabClickRankingProbability.foreach {

case Some(s) =>

selectedCandidateScoreStats

.scope(candidate.commonRecType.toString)

.stat("weighted\_open\_or\_ntab\_click\_ranking")

.add((s \* 10000).toFloat)

case None =>

emptyScoreStats

.scope(candidate.commonRecType.toString)

.counter("weighted\_open\_or\_ntab\_click\_ranking")

.incr()

}

}

def refreshRequestExceptionStats(

exception: Throwable,

bStats: StatsReceiver

): Unit = {

bStats.counter("failures").incr()

bStats.scope("failures").counter(exception.getClass.getCanonicalName).incr()

}

def loggedOutRequestExceptionStats(

exception: Throwable,

bStats: StatsReceiver

): Unit = {

bStats.counter("logged\_out\_failures").incr()

bStats.scope("failures").counter(exception.getClass.getCanonicalName).incr()

}

def rankDistributionStats(

candidatesDetails: Seq[CandidateDetails[PushCandidate]],

numRecsPerTypeStat: (CommonRecommendationType => Stat)

): Unit = {

candidatesDetails

.groupBy { c =>

c.candidate.commonRecType

}

.mapValues { s =>

s.size

}

.foreach { case (crt, numRecs) => numRecsPerTypeStat(crt).add(numRecs) }

}

}