package com.twitter.frigate.pushservice.predicate

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.pushservice.model.TopTweetImpressionsPushCandidate

import com.twitter.frigate.pushservice.params.{PushFeatureSwitchParams => FS}

import com.twitter.frigate.thriftscala.CommonRecommendationType

import com.twitter.hermit.predicate.NamedPredicate

import com.twitter.hermit.predicate.Predicate

object TopTweetImpressionsPredicates {

def topTweetImpressionsFatiguePredicate(

implicit stats: StatsReceiver

): NamedPredicate[TopTweetImpressionsPushCandidate] = {

val name = "top\_tweet\_impressions\_fatigue"

val scopedStats = stats.scope(name)

val bucketImpressionCounter = scopedStats.counter("bucket\_impression\_count")

Predicate

.fromAsync { candidate: TopTweetImpressionsPushCandidate =>

val interval = candidate.target.params(FS.TopTweetImpressionsNotificationInterval)

val maxInInterval = candidate.target.params(FS.MaxTopTweetImpressionsNotifications)

val minInterval = candidate.target.params(FS.TopTweetImpressionsFatigueMinIntervalDuration)

bucketImpressionCounter.incr()

val fatiguePredicate = FatiguePredicate.recTypeOnly(

interval = interval,

maxInInterval = maxInInterval,

minInterval = minInterval,

recommendationType = CommonRecommendationType.TweetImpressions

)

fatiguePredicate.apply(Seq(candidate)).map(\_.head)

}

.withStats(stats.scope(s"predicate\_${name}"))

.withName(name)

}

def topTweetImpressionsThreshold(

)(

implicit statsReceiver: StatsReceiver

): NamedPredicate[TopTweetImpressionsPushCandidate] = {

val name = "top\_tweet\_impressions\_threshold"

val scopedStats = statsReceiver.scope(name)

val meetsImpressionsCounter = scopedStats.counter("meets\_impressions\_count")

val bucketImpressionCounter = scopedStats.counter("bucket\_impression\_count")

Predicate

.from[TopTweetImpressionsPushCandidate] { candidate =>

val meetsImpressionsThreshold =

candidate.impressionsCount >= candidate.target.params(FS.TopTweetImpressionsThreshold)

if (meetsImpressionsThreshold) meetsImpressionsCounter.incr()

bucketImpressionCounter.incr()

meetsImpressionsThreshold

}

.withStats(statsReceiver.scope(s"predicate\_${name}"))

.withName(name)

}

}