package com.twitter.cr\_mixer.blender

import com.twitter.cr\_mixer.blender.ImplicitSignalBackFillBlender.BackFillSourceTypes

import com.twitter.cr\_mixer.blender.ImplicitSignalBackFillBlender.BackFillSourceTypesWithVideo

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

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

import com.twitter.cr\_mixer.param.BlenderParams

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

import com.twitter.cr\_mixer.util.InterleaveUtil

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.timelines.configapi.Params

import com.twitter.util.Future

import javax.inject.Inject

case class SourceTypeBackFillBlender @Inject() (globalStats: StatsReceiver) {

private val name: String = this.getClass.getCanonicalName

private val stats: StatsReceiver = globalStats.scope(name)

/\*\*

\* Partition the candidates based on source type

\* Interleave the two partitions of candidates separately

\* Then append the back fill candidates to the end

\*/

def blend(

params: Params,

inputCandidates: Seq[Seq[InitialCandidate]],

): Future[Seq[BlendedCandidate]] = {

// Filter out empty candidate sequence

val candidates = inputCandidates.filter(\_.nonEmpty)

val backFillSourceTypes =

if (params(BlenderParams.SourceTypeBackFillEnableVideoBackFill)) BackFillSourceTypesWithVideo

else BackFillSourceTypes

// partition candidates based on their source types

val (backFillCandidates, regularCandidates) =

candidates.partition(

\_.head.candidateGenerationInfo.sourceInfoOpt

.exists(sourceInfo => backFillSourceTypes.contains(sourceInfo.sourceType)))

val interleavedRegularCandidates = InterleaveUtil.interleave(regularCandidates)

val interleavedBackFillCandidates =

InterleaveUtil.interleave(backFillCandidates)

stats.stat("backFillCandidates").add(interleavedBackFillCandidates.size)

// Append interleaved backfill candidates to the end

val interleavedCandidates = interleavedRegularCandidates ++ interleavedBackFillCandidates

stats.stat("candidates").add(interleavedCandidates.size)

val blendedCandidates = BlendedCandidatesBuilder.build(inputCandidates, interleavedCandidates)

Future.value(blendedCandidates)

}

}

object ImplicitSignalBackFillBlender {

final val BackFillSourceTypesWithVideo: Set[SourceType] = Set(

SourceType.UserRepeatedProfileVisit,

SourceType.VideoTweetPlayback50,

SourceType.VideoTweetQualityView)

final val BackFillSourceTypes: Set[SourceType] = Set(SourceType.UserRepeatedProfileVisit)

}