package com.twitter.cr\_mixer.blender

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

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

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

import com.twitter.simclusters\_v2.common.TweetId

import scala.collection.mutable

object BlendedCandidatesBuilder {

/\*\*

\* @param inputCandidates input candidate prior to interleaving

\* @param interleavedCandidates after interleaving. These tweets are de-duplicated.

\*/

def build(

inputCandidates: Seq[Seq[InitialCandidate]],

interleavedCandidates: Seq[InitialCandidate]

): Seq[BlendedCandidate] = {

val cgInfoLookupMap = buildCandidateToCGInfosMap(inputCandidates)

interleavedCandidates.map { interleavedCandidate =>

interleavedCandidate.toBlendedCandidate(cgInfoLookupMap(interleavedCandidate.tweetId))

}

}

/\*\*

\* The same tweet can be generated by different sources.

\* This function tells you which CandidateGenerationInfo generated a given tweet

\*/

private def buildCandidateToCGInfosMap(

candidateSeq: Seq[Seq[InitialCandidate]],

): Map[TweetId, Seq[CandidateGenerationInfo]] = {

val tweetIdMap = mutable.HashMap[TweetId, Seq[CandidateGenerationInfo]]()

candidateSeq.foreach { candidates =>

candidates.foreach { candidate =>

val candidateGenerationInfoSeq = {

tweetIdMap.getOrElse(candidate.tweetId, Seq.empty)

}

val candidateGenerationInfo = candidate.candidateGenerationInfo

tweetIdMap.put(

candidate.tweetId,

candidateGenerationInfoSeq ++ Seq(candidateGenerationInfo))

}

}

tweetIdMap.toMap

}

}