package com.twitter.frigate.pushservice.rank

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

/\*\*

\* This Ranker re-ranks MR candidates, boosting input CRTs.

\* Relative ranking between input CRTs and rest of the candidates doesn't change

\*

\* Ex: T: Tweet candidate, F: input CRT candidatess

\*

\* T3, F2, T1, T2, F1 => F2, F1, T3, T1, T2

\*/

case class CRTBoostRanker(statsReceiver: StatsReceiver) {

private val recsToBoostStat = statsReceiver.stat("recs\_to\_boost")

private val otherRecsStat = statsReceiver.stat("other\_recs")

private def boostCrtToTop(

inputCandidates: Seq[CandidateDetails[PushCandidate]],

crtToBoost: CommonRecommendationType

): Seq[CandidateDetails[PushCandidate]] = {

val (upRankedCandidates, otherCandidates) =

inputCandidates.partition(\_.candidate.commonRecType == crtToBoost)

recsToBoostStat.add(upRankedCandidates.size)

otherRecsStat.add(otherCandidates.size)

upRankedCandidates ++ otherCandidates

}

final def boostCrtsToTop(

inputCandidates: Seq[CandidateDetails[PushCandidate]],

crtsToBoost: Seq[CommonRecommendationType]

): Seq[CandidateDetails[PushCandidate]] = {

crtsToBoost.headOption match {

case Some(crt) =>

val upRankedCandidates = boostCrtToTop(inputCandidates, crt)

boostCrtsToTop(upRankedCandidates, crtsToBoost.tail)

case None => inputCandidates

}

}

final def boostCrtsToTopStableOrder(

inputCandidates: Seq[CandidateDetails[PushCandidate]],

crtsToBoost: Seq[CommonRecommendationType]

): Seq[CandidateDetails[PushCandidate]] = {

val crtsToBoostSet = crtsToBoost.toSet

val (upRankedCandidates, otherCandidates) = inputCandidates.partition(candidateDetail =>

crtsToBoostSet.contains(candidateDetail.candidate.commonRecType))

upRankedCandidates ++ otherCandidates

}

}