package com.twitter.home\_mixer.product.list\_recommended\_users.filter

import com.twitter.home\_mixer.product.list\_recommended\_users.model.ListRecommendedUsersFeatures.ScoreFeature

import com.twitter.product\_mixer.component\_library.model.candidate.UserCandidate

import com.twitter.product\_mixer.core.functional\_component.filter.Filter

import com.twitter.product\_mixer.core.functional\_component.filter.FilterResult

import com.twitter.product\_mixer.core.model.common.CandidateWithFeatures

import com.twitter.product\_mixer.core.model.common.identifier.FilterIdentifier

import com.twitter.product\_mixer.core.pipeline.PipelineQuery

import com.twitter.stitch.Stitch

object DropMaxCandidatesByAggregatedScoreFilter extends Filter[PipelineQuery, UserCandidate] {

override val identifier: FilterIdentifier = FilterIdentifier("DropMaxCandidatesByAggregatedScore")

private val MaxSimilarUserCandidates = 150

override def apply(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[UserCandidate]]

): Stitch[FilterResult[UserCandidate]] = {

val userIdToAggregatedScoreMap = candidates

.groupBy(\_.candidate.id)

.map {

case (userId, candidates) =>

val aggregatedScore = candidates.map(\_.features.getOrElse(ScoreFeature, 0.0)).sum

(userId, aggregatedScore)

}

val sortedCandidates = candidates.sortBy(candidate =>

-userIdToAggregatedScoreMap.getOrElse(candidate.candidate.id, 0.0))

val (kept, removed) = sortedCandidates.map(\_.candidate).splitAt(MaxSimilarUserCandidates)

Stitch.value(FilterResult(kept, removed))

}

}