package com.twitter.home\_mixer.product.scored\_tweets.scorer

import com.twitter.home\_mixer.model.HomeFeatures.AuthorIdFeature

import com.twitter.home\_mixer.model.HomeFeatures.ScoreFeature

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

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

trait DiversityDiscountProvider {

/\*\*

\* Fetch the ID of the entity to diversify

\*/

def entityId(candidate: CandidateWithFeatures[TweetCandidate]): Option[Long]

/\*\*

\* Compute discount factor for each candidate based on position (zero-based)

\* relative to other candidates associated with the same entity

\*/

def discount(position: Int): Double

/\*\*

\* Return candidate IDs sorted by score in descending order

\*/

def sort(candidates: Seq[CandidateWithFeatures[TweetCandidate]]): Seq[Long] = candidates

.map { candidate =>

(candidate.candidate.id, candidate.features.getOrElse(ScoreFeature, None).getOrElse(0.0))

}

.sortBy(\_.\_2)(Ordering.Double.reverse)

.map(\_.\_1)

/\*\*

\* Group by the specified entity ID (e.g. authors, likers, followers)

\* Sort each group by score in descending order

\* Determine the discount factor based on the position of each candidate

\*/

def apply(

candidates: Seq[CandidateWithFeatures[TweetCandidate]]

): Map[Long, Double] = candidates

.groupBy(entityId)

.flatMap {

case (entityIdOpt, entityCandidates) =>

val sortedCandidateIds = sort(entityCandidates)

if (entityIdOpt.isDefined) {

sortedCandidateIds.zipWithIndex.map {

case (candidateId, index) =>

candidateId -> discount(index)

}

} else sortedCandidateIds.map(\_ -> 1.0)

}

}

object AuthorDiversityDiscountProvider extends DiversityDiscountProvider {

private val Decay = 0.5

private val Floor = 0.25

override def entityId(candidate: CandidateWithFeatures[TweetCandidate]): Option[Long] =

candidate.features.getOrElse(AuthorIdFeature, None)

// Provides an exponential decay based discount by position (with a floor)

override def discount(position: Int): Double =

(1 - Floor) \* Math.pow(Decay, position) + Floor

}