package com.twitter.home\_mixer.product.scored\_tweets.feature\_hydrator

import com.twitter.home\_mixer.product.scored\_tweets.param.ScoredTweetsParam

import com.twitter.ml.api.DataRecord

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

import com.twitter.product\_mixer.core.feature.Feature

import com.twitter.product\_mixer.core.feature.FeatureWithDefaultOnFailure

import com.twitter.product\_mixer.core.feature.datarecord.DataRecordInAFeature

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMap

import com.twitter.product\_mixer.core.feature.featuremap.FeatureMapBuilder

import com.twitter.product\_mixer.core.functional\_component.feature\_hydrator.BulkCandidateFeatureHydrator

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

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

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

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

import com.twitter.product\_mixer.core.util.OffloadFuturePools

import com.twitter.stitch.Stitch

import com.twitter.timelines.clients.strato.twistly.SimClustersRecentEngagementSimilarityClient

import com.twitter.timelines.configapi.decider.BooleanDeciderParam

import com.twitter.timelines.prediction.adapters.twistly.SimClustersRecentEngagementSimilarityFeaturesAdapter

import javax.inject.Inject

import javax.inject.Singleton

object SimClustersEngagementSimilarityFeature

extends DataRecordInAFeature[PipelineQuery]

with FeatureWithDefaultOnFailure[PipelineQuery, DataRecord] {

override def defaultValue: DataRecord = new DataRecord()

}

@Singleton

class SimClustersEngagementSimilarityFeatureHydrator @Inject() (

simClustersEngagementSimilarityClient: SimClustersRecentEngagementSimilarityClient)

extends BulkCandidateFeatureHydrator[PipelineQuery, TweetCandidate]

with Conditionally[PipelineQuery] {

override val identifier: FeatureHydratorIdentifier =

FeatureHydratorIdentifier("SimClustersEngagementSimilarity")

override val features: Set[Feature[\_, \_]] = Set(SimClustersEngagementSimilarityFeature)

private val simClustersRecentEngagementSimilarityFeaturesAdapter =

new SimClustersRecentEngagementSimilarityFeaturesAdapter

override def onlyIf(query: PipelineQuery): Boolean = {

val param: BooleanDeciderParam =

ScoredTweetsParam.EnableSimClustersSimilarityFeatureHydrationDeciderParam

query.params.apply(param)

}

override def apply(

query: PipelineQuery,

candidates: Seq[CandidateWithFeatures[TweetCandidate]]

): Stitch[Seq[FeatureMap]] = OffloadFuturePools.offloadFuture {

val tweetToCandidates = candidates.map(candidate => candidate.candidate.id -> candidate).toMap

val tweetIds = tweetToCandidates.keySet.toSeq

val userId = query.getRequiredUserId

val userTweetEdges = tweetIds.map(tweetId => (userId, tweetId))

simClustersEngagementSimilarityClient

.getSimClustersRecentEngagementSimilarityScores(userTweetEdges).map {

simClustersRecentEngagementSimilarityScoresMap =>

candidates.map { candidate =>

val similarityFeatureOpt = simClustersRecentEngagementSimilarityScoresMap

.get(userId -> candidate.candidate.id).flatten

val dataRecordOpt = similarityFeatureOpt.map { similarityFeature =>

simClustersRecentEngagementSimilarityFeaturesAdapter

.adaptToDataRecords(similarityFeature)

.get(0)

}

FeatureMapBuilder()

.add(SimClustersEngagementSimilarityFeature, dataRecordOpt.getOrElse(new DataRecord))

.build()

}

}

}

}