package com.twitter.cr\_mixer.similarity\_engine

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

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

import com.twitter.cr\_mixer.param.ConsumersBasedUserAdGraphParams

import com.twitter.cr\_mixer.param.GlobalParams

import com.twitter.cr\_mixer.thriftscala.SimilarityEngineType

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.recos.user\_ad\_graph.thriftscala.ConsumersBasedRelatedAdRequest

import com.twitter.recos.user\_ad\_graph.thriftscala.RelatedAdResponse

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

import com.twitter.storehaus.ReadableStore

import com.twitter.timelines.configapi

import com.twitter.util.Future

import javax.inject.Singleton

/\*\*

\* This store uses the graph based input (a list of userIds)

\* to query consumersBasedUserAdGraph and get their top engaged ad tweets

\*/

@Singleton

case class ConsumersBasedUserAdGraphSimilarityEngine(

consumersBasedUserAdGraphStore: ReadableStore[

ConsumersBasedRelatedAdRequest,

RelatedAdResponse

],

statsReceiver: StatsReceiver)

extends ReadableStore[

ConsumersBasedUserAdGraphSimilarityEngine.Query,

Seq[TweetWithScore]

] {

override def get(

query: ConsumersBasedUserAdGraphSimilarityEngine.Query

): Future[Option[Seq[TweetWithScore]]] = {

val consumersBasedRelatedAdRequest =

ConsumersBasedRelatedAdRequest(

query.seedWithScores.keySet.toSeq,

maxResults = Some(query.maxResults),

minCooccurrence = Some(query.minCooccurrence),

minScore = Some(query.minScore),

maxTweetAgeInHours = Some(query.maxTweetAgeInHours)

)

consumersBasedUserAdGraphStore

.get(consumersBasedRelatedAdRequest)

.map { relatedAdResponseOpt =>

relatedAdResponseOpt.map { relatedAdResponse =>

relatedAdResponse.adTweets.map { tweet =>

TweetWithScore(tweet.adTweetId, tweet.score)

}

}

}

}

}

object ConsumersBasedUserAdGraphSimilarityEngine {

case class Query(

seedWithScores: Map[UserId, Double],

maxResults: Int,

minCooccurrence: Int,

minScore: Double,

maxTweetAgeInHours: Int)

def toSimilarityEngineInfo(

score: Double

): SimilarityEngineInfo = {

SimilarityEngineInfo(

similarityEngineType = SimilarityEngineType.ConsumersBasedUserAdGraph,

modelId = None,

score = Some(score))

}

def fromParams(

seedWithScores: Map[UserId, Double],

params: configapi.Params,

): EngineQuery[Query] = {

EngineQuery(

Query(

seedWithScores = seedWithScores,

maxResults = params(GlobalParams.MaxCandidateNumPerSourceKeyParam),

minCooccurrence = params(ConsumersBasedUserAdGraphParams.MinCoOccurrenceParam),

minScore = params(ConsumersBasedUserAdGraphParams.MinScoreParam),

maxTweetAgeInHours = params(GlobalParams.MaxTweetAgeHoursParam).inHours,

),

params

)

}

}