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.ConsumersBasedUserVideoGraphParams

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\_video\_graph.thriftscala.ConsumersBasedRelatedTweetRequest

import com.twitter.recos.user\_video\_graph.thriftscala.RelatedTweetResponse

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 consumersBasedUserVideoGraph and get their top engaged tweets

\*/

@Singleton

case class ConsumersBasedUserVideoGraphSimilarityEngine(

consumersBasedUserVideoGraphStore: ReadableStore[

ConsumersBasedRelatedTweetRequest,

RelatedTweetResponse

],

statsReceiver: StatsReceiver)

extends ReadableStore[

ConsumersBasedUserVideoGraphSimilarityEngine.Query,

Seq[TweetWithScore]

] {

override def get(

query: ConsumersBasedUserVideoGraphSimilarityEngine.Query

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

val consumersBasedRelatedTweetRequest =

ConsumersBasedRelatedTweetRequest(

query.seedWithScores.keySet.toSeq,

maxResults = Some(query.maxResults),

minCooccurrence = Some(query.minCooccurrence),

minScore = Some(query.minScore),

maxTweetAgeInHours = Some(query.maxTweetAgeInHours)

)

consumersBasedUserVideoGraphStore

.get(consumersBasedRelatedTweetRequest)

.map { relatedTweetResponseOpt =>

relatedTweetResponseOpt.map { relatedTweetResponse =>

relatedTweetResponse.tweets.map { tweet =>

TweetWithScore(tweet.tweetId, tweet.score)

}

}

}

}

}

object ConsumersBasedUserVideoGraphSimilarityEngine {

case class Query(

seedWithScores: Map[UserId, Double],

maxResults: Int,

minCooccurrence: Int,

minScore: Double,

maxTweetAgeInHours: Int)

def toSimilarityEngineInfo(

score: Double

): SimilarityEngineInfo = {

SimilarityEngineInfo(

similarityEngineType = SimilarityEngineType.ConsumersBasedUserVideoGraph,

modelId = None,

score = Some(score))

}

def fromParamsForRealGraphIn(

seedWithScores: Map[UserId, Double],

params: configapi.Params,

): EngineQuery[Query] = {

EngineQuery(

Query(

seedWithScores = seedWithScores,

maxResults = params(GlobalParams.MaxCandidateNumPerSourceKeyParam),

minCooccurrence =

params(ConsumersBasedUserVideoGraphParams.RealGraphInMinCoOccurrenceParam),

minScore = params(ConsumersBasedUserVideoGraphParams.RealGraphInMinScoreParam),

maxTweetAgeInHours = params(GlobalParams.MaxTweetAgeHoursParam).inHours

),

params

)

}

}