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

import com.twitter.recos.recos\_common.thriftscala.SocialProofType

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

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

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

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

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.recos.user\_tweet\_entity\_graph.thriftscala.TweetEntityDisplayLocation

import com.twitter.recos.user\_tweet\_entity\_graph.thriftscala.UserTweetEntityGraph

import com.twitter.recos.user\_tweet\_entity\_graph.thriftscala.RecommendTweetEntityRequest

import com.twitter.recos.user\_tweet\_entity\_graph.thriftscala.RecommendationType

import com.twitter.recos.user\_tweet\_entity\_graph.thriftscala.UserTweetEntityRecommendationUnion.TweetRec

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

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

import com.twitter.storehaus.ReadableStore

import com.twitter.timelines.configapi

import com.twitter.util.Duration

import com.twitter.util.Future

import javax.inject.Singleton

@Singleton

case class UserTweetEntityGraphSimilarityEngine(

userTweetEntityGraph: UserTweetEntityGraph.MethodPerEndpoint,

statsReceiver: StatsReceiver)

extends ReadableStore[

UserTweetEntityGraphSimilarityEngine.Query,

Seq[TweetWithScoreAndSocialProof]

] {

override def get(

query: UserTweetEntityGraphSimilarityEngine.Query

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

val recommendTweetEntityRequest =

RecommendTweetEntityRequest(

requesterId = query.userId,

displayLocation = TweetEntityDisplayLocation.HomeTimeline,

recommendationTypes = Seq(RecommendationType.Tweet),

seedsWithWeights = query.seedsWithWeights,

maxResultsByType = Some(Map(RecommendationType.Tweet -> query.maxUtegCandidates)),

maxTweetAgeInMillis = Some(query.maxTweetAge.inMilliseconds),

excludedTweetIds = query.excludedTweetIds,

maxUserSocialProofSize = Some(UserTweetEntityGraphSimilarityEngine.MaxUserSocialProofSize),

maxTweetSocialProofSize =

Some(UserTweetEntityGraphSimilarityEngine.MaxTweetSocialProofSize),

minUserSocialProofSizes = Some(Map(RecommendationType.Tweet -> 1)),

tweetTypes = None,

socialProofTypes = query.socialProofTypes,

socialProofTypeUnions = None,

tweetAuthors = None,

maxEngagementAgeInMillis = None,

excludedTweetAuthors = None,

)

userTweetEntityGraph

.recommendTweets(recommendTweetEntityRequest)

.map { recommendTweetsResponse =>

val candidates = recommendTweetsResponse.recommendations.flatMap {

case TweetRec(recommendation) =>

Some(

TweetWithScoreAndSocialProof(

recommendation.tweetId,

recommendation.score,

recommendation.socialProofByType.toMap))

case \_ => None

}

Some(candidates)

}

}

}

object UserTweetEntityGraphSimilarityEngine {

private val MaxUserSocialProofSize = 10

private val MaxTweetSocialProofSize = 10

def toSimilarityEngineInfo(score: Double): SimilarityEngineInfo = {

SimilarityEngineInfo(

similarityEngineType = SimilarityEngineType.Uteg,

modelId = None,

score = Some(score))

}

case class Query(

userId: UserId,

seedsWithWeights: Map[UserId, Double],

excludedTweetIds: Option[Seq[Long]] = None,

maxUtegCandidates: Int,

maxTweetAge: Duration,

socialProofTypes: Option[Seq[SocialProofType]])

def fromParams(

userId: UserId,

seedsWithWeights: Map[UserId, Double],

excludedTweetIds: Option[Seq[TweetId]] = None,

params: configapi.Params,

): EngineQuery[Query] = {

EngineQuery(

Query(

userId = userId,

seedsWithWeights = seedsWithWeights,

excludedTweetIds = excludedTweetIds,

maxUtegCandidates = params(UtegTweetGlobalParams.MaxUtegCandidatesToRequestParam),

maxTweetAge = params(UtegTweetGlobalParams.CandidateRefreshSinceTimeOffsetHoursParam),

socialProofTypes = Some(Seq(SocialProofType.Favorite))

),

params

)

}

}