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

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

import com.twitter.simclusters\_v2.thriftscala.TweetsWithScore

import com.twitter.simclusters\_v2.thriftscala.InternalId

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

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

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.simclusters\_v2.thriftscala.InternalId

import com.twitter.storehaus.ReadableStore

import com.twitter.timelines.configapi

import com.twitter.util.Future

import javax.inject.Singleton

@Singleton

case class DiffusionBasedSimilarityEngine(

retweetBasedDiffusionRecsMhStore: ReadableStore[Long, TweetsWithScore],

statsReceiver: StatsReceiver)

extends ReadableStore[

DiffusionBasedSimilarityEngine.Query,

Seq[TweetWithScore]

] {

override def get(

query: DiffusionBasedSimilarityEngine.Query

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

query.sourceId match {

case InternalId.UserId(userId) =>

retweetBasedDiffusionRecsMhStore.get(userId).map {

\_.map { tweetsWithScore =>

{

tweetsWithScore.tweets

.map(tweet => TweetWithScore(tweet.tweetId, tweet.score))

}

}

}

case \_ =>

Future.None

}

}

}

object DiffusionBasedSimilarityEngine {

val defaultScore: Double = 0.0

case class Query(

sourceId: InternalId,

)

def toSimilarityEngineInfo(

query: LookupEngineQuery[Query],

score: Double

): SimilarityEngineInfo = {

SimilarityEngineInfo(

similarityEngineType = SimilarityEngineType.DiffusionBasedTweet,

modelId = Some(query.lookupKey),

score = Some(score))

}

def fromParams(

sourceId: InternalId,

modelId: String,

params: configapi.Params,

): LookupEngineQuery[Query] = {

LookupEngineQuery(

Query(sourceId = sourceId),

modelId,

params

)

}

}