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

import com.google.inject.Inject

import com.google.inject.Singleton

import com.google.inject.name.Named

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

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

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

import com.twitter.cr\_mixer.similarity\_engine.CertoTopicTweetSimilarityEngine.\_

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

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.util.StatsUtil

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

import com.twitter.storehaus.ReadableStore

import com.twitter.timelines.configapi

import com.twitter.topic\_recos.thriftscala.\_

import com.twitter.util.Future

@Singleton

case class CertoTopicTweetSimilarityEngine @Inject() (

@Named(ModuleNames.CertoStratoStoreName) certoStratoStore: ReadableStore[

TopicId,

Seq[TweetWithScores]

],

statsReceiver: StatsReceiver)

extends ReadableStore[EngineQuery[Query], Seq[TopicTweetWithScore]] {

private val name: String = this.getClass.getSimpleName

private val stats = statsReceiver.scope(name)

override def get(query: EngineQuery[Query]): Future[Option[Seq[TopicTweetWithScore]]] = {

StatsUtil.trackOptionItemsStats(stats) {

topTweetsByFollowerL2NormalizedScore.get(query).map {

\_.map { topicTopTweets =>

topicTopTweets.map { topicTweet =>

TopicTweetWithScore(

tweetId = topicTweet.tweetId,

score = topicTweet.scores.followerL2NormalizedCosineSimilarity8HrHalfLife,

similarityEngineType = SimilarityEngineType.CertoTopicTweet

)

}

}

}

}

}

private val topTweetsByFollowerL2NormalizedScore: ReadableStore[EngineQuery[Query], Seq[

TweetWithScores

]] = {

ReadableStore.fromFnFuture { query: EngineQuery[Query] =>

StatsUtil.trackOptionItemsStats(stats) {

for {

topKTweetsWithScores <- certoStratoStore.get(query.storeQuery.topicId)

} yield {

topKTweetsWithScores.map(

\_.filter(

\_.scores.followerL2NormalizedCosineSimilarity8HrHalfLife >= query.storeQuery.certoScoreTheshold)

.take(query.storeQuery.maxCandidates))

}

}

}

}

}

object CertoTopicTweetSimilarityEngine {

// Query is used as a cache key. Do not add any user level information in this.

case class Query(

topicId: TopicId,

maxCandidates: Int,

certoScoreTheshold: Double)

def fromParams(

topicId: TopicId,

isVideoOnly: Boolean,

params: configapi.Params,

): EngineQuery[Query] = {

val maxCandidates = if (isVideoOnly) {

params(TopicTweetParams.MaxCertoCandidatesParam) \* 2

} else {

params(TopicTweetParams.MaxCertoCandidatesParam)

}

EngineQuery(

Query(

topicId = topicId,

maxCandidates = maxCandidates,

certoScoreTheshold = params(TopicTweetParams.CertoScoreThresholdParam)

),

params

)

}

}