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

import com.twitter.cr\_mixer.config.TimeoutConfig

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

import com.twitter.cr\_mixer.model.EarlybirdSimilarityEngineType\_ModelBased

import com.twitter.cr\_mixer.model.EarlybirdSimilarityEngineType\_RecencyBased

import com.twitter.cr\_mixer.model.EarlybirdSimilarityEngineType\_TensorflowBased

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

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

import com.twitter.cr\_mixer.param.EarlybirdFrsBasedCandidateGenerationParams.FrsBasedCandidateGenerationEarlybirdSimilarityEngineTypeParam

import com.twitter.cr\_mixer.param.FrsParams.FrsBasedCandidateGenerationMaxCandidatesNumParam

import com.twitter.finagle.stats.StatsReceiver

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

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

import com.twitter.snowflake.id.SnowflakeId

import com.twitter.storehaus.ReadableStore

import com.twitter.timelines.configapi

import com.twitter.util.Duration

import com.twitter.util.Future

import com.twitter.util.Time

import javax.inject.Inject

import javax.inject.Singleton

@Singleton

case class EarlybirdSimilarityEngineRouter @Inject() (

earlybirdRecencyBasedSimilarityEngine: EarlybirdSimilarityEngine[

EarlybirdRecencyBasedSimilarityEngine.EarlybirdRecencyBasedSearchQuery,

EarlybirdRecencyBasedSimilarityEngine

],

earlybirdModelBasedSimilarityEngine: EarlybirdSimilarityEngine[

EarlybirdModelBasedSimilarityEngine.EarlybirdModelBasedSearchQuery,

EarlybirdModelBasedSimilarityEngine

],

earlybirdTensorflowBasedSimilarityEngine: EarlybirdSimilarityEngine[

EarlybirdTensorflowBasedSimilarityEngine.EarlybirdTensorflowBasedSearchQuery,

EarlybirdTensorflowBasedSimilarityEngine

],

timeoutConfig: TimeoutConfig,

statsReceiver: StatsReceiver)

extends ReadableStore[EarlybirdSimilarityEngineRouter.Query, Seq[TweetWithAuthor]] {

import EarlybirdSimilarityEngineRouter.\_

override def get(

k: EarlybirdSimilarityEngineRouter.Query

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

k.rankingMode match {

case EarlybirdSimilarityEngineType\_RecencyBased =>

earlybirdRecencyBasedSimilarityEngine.getCandidates(recencyBasedQueryFromParams(k))

case EarlybirdSimilarityEngineType\_ModelBased =>

earlybirdModelBasedSimilarityEngine.getCandidates(modelBasedQueryFromParams(k))

case EarlybirdSimilarityEngineType\_TensorflowBased =>

earlybirdTensorflowBasedSimilarityEngine.getCandidates(tensorflowBasedQueryFromParams(k))

}

}

}

object EarlybirdSimilarityEngineRouter {

case class Query(

searcherUserId: Option[UserId],

seedUserIds: Seq[UserId],

maxNumTweets: Int,

excludedTweetIds: Set[TweetId],

rankingMode: EarlybirdSimilarityEngineType,

frsUserToScoresForScoreAdjustment: Option[Map[UserId, Double]],

maxTweetAge: Duration,

filterOutRetweetsAndReplies: Boolean,

params: configapi.Params)

def queryFromParams(

searcherUserId: Option[UserId],

seedUserIds: Seq[UserId],

excludedTweetIds: Set[TweetId],

frsUserToScoresForScoreAdjustment: Option[Map[UserId, Double]],

params: configapi.Params

): Query =

Query(

searcherUserId,

seedUserIds,

maxNumTweets = params(FrsBasedCandidateGenerationMaxCandidatesNumParam),

excludedTweetIds,

rankingMode =

params(FrsBasedCandidateGenerationEarlybirdSimilarityEngineTypeParam).rankingMode,

frsUserToScoresForScoreAdjustment,

maxTweetAge = params(

EarlybirdFrsBasedCandidateGenerationParams.FrsBasedCandidateGenerationEarlybirdMaxTweetAge),

filterOutRetweetsAndReplies = params(

EarlybirdFrsBasedCandidateGenerationParams.FrsBasedCandidateGenerationEarlybirdFilterOutRetweetsAndReplies),

params

)

private def recencyBasedQueryFromParams(

query: Query

): EngineQuery[EarlybirdRecencyBasedSimilarityEngine.EarlybirdRecencyBasedSearchQuery] =

EngineQuery(

EarlybirdRecencyBasedSimilarityEngine.EarlybirdRecencyBasedSearchQuery(

seedUserIds = query.seedUserIds,

maxNumTweets = query.maxNumTweets,

excludedTweetIds = query.excludedTweetIds,

maxTweetAge = query.maxTweetAge,

filterOutRetweetsAndReplies = query.filterOutRetweetsAndReplies

),

query.params

)

private def tensorflowBasedQueryFromParams(

query: Query,

): EngineQuery[EarlybirdTensorflowBasedSimilarityEngine.EarlybirdTensorflowBasedSearchQuery] =

EngineQuery(

EarlybirdTensorflowBasedSimilarityEngine.EarlybirdTensorflowBasedSearchQuery(

searcherUserId = query.searcherUserId,

seedUserIds = query.seedUserIds,

maxNumTweets = query.maxNumTweets,

// hard code the params below for now. Will move to FS after shipping the ddg

beforeTweetIdExclusive = None,

afterTweetIdExclusive =

Some(SnowflakeId.firstIdFor((Time.now - query.maxTweetAge).inMilliseconds)),

filterOutRetweetsAndReplies = query.filterOutRetweetsAndReplies,

useTensorflowRanking = true,

excludedTweetIds = query.excludedTweetIds,

maxNumHitsPerShard = 1000

),

query.params

)

private def modelBasedQueryFromParams(

query: Query,

): EngineQuery[EarlybirdModelBasedSimilarityEngine.EarlybirdModelBasedSearchQuery] =

EngineQuery(

EarlybirdModelBasedSimilarityEngine.EarlybirdModelBasedSearchQuery(

seedUserIds = query.seedUserIds,

maxNumTweets = query.maxNumTweets,

oldestTweetTimestampInSec = Some(query.maxTweetAge.ago.inSeconds),

frsUserToScoresForScoreAdjustment = query.frsUserToScoresForScoreAdjustment

),

query.params

)

}