package com.twitter.cr\_mixer.candidate\_generation

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

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

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

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

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

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

import com.twitter.cr\_mixer.similarity\_engine.EngineQuery

import com.twitter.cr\_mixer.similarity\_engine.SimClustersANNSimilarityEngine

import com.twitter.cr\_mixer.similarity\_engine.StandardSimilarityEngine

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.frigate.common.base.CandidateSource

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

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

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

import com.twitter.timelines.configapi

import com.twitter.util.Future

import javax.inject.Inject

import javax.inject.Singleton

import javax.inject.Named

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

/\*\*

\* This store looks for similar tweets for a given UserId that generates UserInterestedIn

\* from SimClustersANN. It will be a standalone CandidateGeneration class moving forward.

\*

\* After the abstraction improvement (apply SimilarityEngine trait)

\* these CG will be subjected to change.

\*/

@Singleton

case class SimClustersInterestedInCandidateGeneration @Inject() (

@Named(ModuleNames.SimClustersANNSimilarityEngine)

simClustersANNSimilarityEngine: StandardSimilarityEngine[

SimClustersANNSimilarityEngine.Query,

TweetWithScore

],

statsReceiver: StatsReceiver)

extends CandidateSource[

SimClustersInterestedInCandidateGeneration.Query,

Seq[TweetWithCandidateGenerationInfo]

] {

override def name: String = this.getClass.getSimpleName

private val stats = statsReceiver.scope(name)

private val fetchCandidatesStat = stats.scope("fetchCandidates")

override def get(

query: SimClustersInterestedInCandidateGeneration.Query

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

query.internalId match {

case \_: InternalId.UserId =>

StatsUtil.trackOptionItemsStats(fetchCandidatesStat) {

// UserInterestedIn Queries

val userInterestedInCandidateResultFut =

if (query.enableUserInterestedIn && query.enableProdSimClustersANNSimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.interestedInSimClustersANNQuery,

query.simClustersInterestedInMinScore)

else

Future.None

val userInterestedInExperimentalSANNCandidateResultFut =

if (query.enableUserInterestedIn && query.enableExperimentalSimClustersANNSimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.interestedInExperimentalSimClustersANNQuery,

query.simClustersInterestedInMinScore)

else

Future.None

val userInterestedInSANN1CandidateResultFut =

if (query.enableUserInterestedIn && query.enableSimClustersANN1SimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.interestedInSimClustersANN1Query,

query.simClustersInterestedInMinScore)

else

Future.None

val userInterestedInSANN2CandidateResultFut =

if (query.enableUserInterestedIn && query.enableSimClustersANN2SimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.interestedInSimClustersANN2Query,

query.simClustersInterestedInMinScore)

else

Future.None

val userInterestedInSANN3CandidateResultFut =

if (query.enableUserInterestedIn && query.enableSimClustersANN3SimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.interestedInSimClustersANN3Query,

query.simClustersInterestedInMinScore)

else

Future.None

val userInterestedInSANN5CandidateResultFut =

if (query.enableUserInterestedIn && query.enableSimClustersANN5SimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.interestedInSimClustersANN5Query,

query.simClustersInterestedInMinScore)

else

Future.None

val userInterestedInSANN4CandidateResultFut =

if (query.enableUserInterestedIn && query.enableSimClustersANN4SimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.interestedInSimClustersANN4Query,

query.simClustersInterestedInMinScore)

else

Future.None

// UserNextInterestedIn Queries

val userNextInterestedInCandidateResultFut =

if (query.enableUserNextInterestedIn && query.enableProdSimClustersANNSimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.nextInterestedInSimClustersANNQuery,

query.simClustersInterestedInMinScore)

else

Future.None

val userNextInterestedInExperimentalSANNCandidateResultFut =

if (query.enableUserNextInterestedIn && query.enableExperimentalSimClustersANNSimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.nextInterestedInExperimentalSimClustersANNQuery,

query.simClustersInterestedInMinScore)

else

Future.None

val userNextInterestedInSANN1CandidateResultFut =

if (query.enableUserNextInterestedIn && query.enableSimClustersANN1SimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.nextInterestedInSimClustersANN1Query,

query.simClustersInterestedInMinScore)

else

Future.None

val userNextInterestedInSANN2CandidateResultFut =

if (query.enableUserNextInterestedIn && query.enableSimClustersANN2SimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.nextInterestedInSimClustersANN2Query,

query.simClustersInterestedInMinScore)

else

Future.None

val userNextInterestedInSANN3CandidateResultFut =

if (query.enableUserNextInterestedIn && query.enableSimClustersANN3SimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.nextInterestedInSimClustersANN3Query,

query.simClustersInterestedInMinScore)

else

Future.None

val userNextInterestedInSANN5CandidateResultFut =

if (query.enableUserNextInterestedIn && query.enableSimClustersANN5SimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.nextInterestedInSimClustersANN5Query,

query.simClustersInterestedInMinScore)

else

Future.None

val userNextInterestedInSANN4CandidateResultFut =

if (query.enableUserNextInterestedIn && query.enableSimClustersANN4SimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.nextInterestedInSimClustersANN4Query,

query.simClustersInterestedInMinScore)

else

Future.None

// AddressBookInterestedIn Queries

val userAddressBookInterestedInCandidateResultFut =

if (query.enableAddressBookNextInterestedIn && query.enableProdSimClustersANNSimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.addressbookInterestedInSimClustersANNQuery,

query.simClustersInterestedInMinScore)

else

Future.None

val userAddressBookExperimentalSANNCandidateResultFut =

if (query.enableAddressBookNextInterestedIn && query.enableExperimentalSimClustersANNSimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.addressbookInterestedInExperimentalSimClustersANNQuery,

query.simClustersInterestedInMinScore)

else

Future.None

val userAddressBookSANN1CandidateResultFut =

if (query.enableAddressBookNextInterestedIn && query.enableSimClustersANN1SimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.addressbookInterestedInSimClustersANN1Query,

query.simClustersInterestedInMinScore)

else

Future.None

val userAddressBookSANN2CandidateResultFut =

if (query.enableAddressBookNextInterestedIn && query.enableSimClustersANN2SimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.addressbookInterestedInSimClustersANN2Query,

query.simClustersInterestedInMinScore)

else

Future.None

val userAddressBookSANN3CandidateResultFut =

if (query.enableAddressBookNextInterestedIn && query.enableSimClustersANN3SimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.addressbookInterestedInSimClustersANN3Query,

query.simClustersInterestedInMinScore)

else

Future.None

val userAddressBookSANN5CandidateResultFut =

if (query.enableAddressBookNextInterestedIn && query.enableSimClustersANN5SimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.addressbookInterestedInSimClustersANN5Query,

query.simClustersInterestedInMinScore)

else

Future.None

val userAddressBookSANN4CandidateResultFut =

if (query.enableAddressBookNextInterestedIn && query.enableSimClustersANN4SimilarityEngine)

getInterestedInCandidateResult(

simClustersANNSimilarityEngine,

query.addressbookInterestedInSimClustersANN4Query,

query.simClustersInterestedInMinScore)

else

Future.None

Future

.collect(

Seq(

userInterestedInCandidateResultFut,

userNextInterestedInCandidateResultFut,

userAddressBookInterestedInCandidateResultFut,

userInterestedInExperimentalSANNCandidateResultFut,

userNextInterestedInExperimentalSANNCandidateResultFut,

userAddressBookExperimentalSANNCandidateResultFut,

userInterestedInSANN1CandidateResultFut,

userNextInterestedInSANN1CandidateResultFut,

userAddressBookSANN1CandidateResultFut,

userInterestedInSANN2CandidateResultFut,

userNextInterestedInSANN2CandidateResultFut,

userAddressBookSANN2CandidateResultFut,

userInterestedInSANN3CandidateResultFut,

userNextInterestedInSANN3CandidateResultFut,

userAddressBookSANN3CandidateResultFut,

userInterestedInSANN5CandidateResultFut,

userNextInterestedInSANN5CandidateResultFut,

userAddressBookSANN5CandidateResultFut,

userInterestedInSANN4CandidateResultFut,

userNextInterestedInSANN4CandidateResultFut,

userAddressBookSANN4CandidateResultFut

)

).map { candidateResults =>

Some(

candidateResults.map(candidateResult => candidateResult.getOrElse(Seq.empty))

)

}

}

case \_ =>

stats.counter("sourceId\_is\_not\_userId\_cnt").incr()

Future.None

}

}

private def simClustersCandidateMinScoreFilter(

simClustersAnnCandidates: Seq[TweetWithScore],

simClustersInterestedInMinScore: Double,

simClustersANNConfigId: String

): Seq[TweetWithScore] = {

val filteredCandidates = simClustersAnnCandidates

.filter { candidate =>

candidate.score > simClustersInterestedInMinScore

}

stats.stat(simClustersANNConfigId, "simClustersAnnCandidates\_size").add(filteredCandidates.size)

stats.counter(simClustersANNConfigId, "simClustersAnnRequests").incr()

if (filteredCandidates.isEmpty)

stats.counter(simClustersANNConfigId, "emptyFilteredSimClustersAnnCandidates").incr()

filteredCandidates.map { candidate =>

TweetWithScore(candidate.tweetId, candidate.score)

}

}

private def getInterestedInCandidateResult(

simClustersANNSimilarityEngine: StandardSimilarityEngine[

SimClustersANNSimilarityEngine.Query,

TweetWithScore

],

simClustersANNQuery: EngineQuery[SimClustersANNSimilarityEngine.Query],

simClustersInterestedInMinScore: Double,

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

val interestedInCandidatesFut =

simClustersANNSimilarityEngine.getCandidates(simClustersANNQuery)

val interestedInCandidateResultFut = interestedInCandidatesFut.map { interestedInCandidates =>

stats.stat("candidateSize").add(interestedInCandidates.size)

val embeddingCandidatesStat = stats.scope(

simClustersANNQuery.storeQuery.simClustersANNQuery.sourceEmbeddingId.embeddingType.name)

embeddingCandidatesStat.stat("candidateSize").add(interestedInCandidates.size)

if (interestedInCandidates.isEmpty) {

embeddingCandidatesStat.counter("empty\_results").incr()

}

embeddingCandidatesStat.counter("requests").incr()

val filteredTweets = simClustersCandidateMinScoreFilter(

interestedInCandidates.toSeq.flatten,

simClustersInterestedInMinScore,

simClustersANNQuery.storeQuery.simClustersANNConfigId)

val interestedInTweetsWithCGInfo = filteredTweets.map { tweetWithScore =>

TweetWithCandidateGenerationInfo(

tweetWithScore.tweetId,

CandidateGenerationInfo(

None,

SimClustersANNSimilarityEngine

.toSimilarityEngineInfo(simClustersANNQuery, tweetWithScore.score),

Seq.empty // SANN is an atomic SE, and hence it has no contributing SEs

)

)

}

val interestedInResults = if (interestedInTweetsWithCGInfo.nonEmpty) {

Some(interestedInTweetsWithCGInfo)

} else None

interestedInResults

}

interestedInCandidateResultFut

}

}

object SimClustersInterestedInCandidateGeneration {

case class Query(

internalId: InternalId,

enableUserInterestedIn: Boolean,

enableUserNextInterestedIn: Boolean,

enableAddressBookNextInterestedIn: Boolean,

enableProdSimClustersANNSimilarityEngine: Boolean,

enableExperimentalSimClustersANNSimilarityEngine: Boolean,

enableSimClustersANN1SimilarityEngine: Boolean,

enableSimClustersANN2SimilarityEngine: Boolean,

enableSimClustersANN3SimilarityEngine: Boolean,

enableSimClustersANN5SimilarityEngine: Boolean,

enableSimClustersANN4SimilarityEngine: Boolean,

simClustersInterestedInMinScore: Double,

simClustersNextInterestedInMinScore: Double,

simClustersAddressBookInterestedInMinScore: Double,

interestedInSimClustersANNQuery: EngineQuery[SimClustersANNSimilarityEngine.Query],

nextInterestedInSimClustersANNQuery: EngineQuery[SimClustersANNSimilarityEngine.Query],

addressbookInterestedInSimClustersANNQuery: EngineQuery[SimClustersANNSimilarityEngine.Query],

interestedInExperimentalSimClustersANNQuery: EngineQuery[SimClustersANNSimilarityEngine.Query],

nextInterestedInExperimentalSimClustersANNQuery: EngineQuery[

SimClustersANNSimilarityEngine.Query

],

addressbookInterestedInExperimentalSimClustersANNQuery: EngineQuery[

SimClustersANNSimilarityEngine.Query

],

interestedInSimClustersANN1Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

nextInterestedInSimClustersANN1Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

addressbookInterestedInSimClustersANN1Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

interestedInSimClustersANN2Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

nextInterestedInSimClustersANN2Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

addressbookInterestedInSimClustersANN2Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

interestedInSimClustersANN3Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

nextInterestedInSimClustersANN3Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

addressbookInterestedInSimClustersANN3Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

interestedInSimClustersANN5Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

nextInterestedInSimClustersANN5Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

addressbookInterestedInSimClustersANN5Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

interestedInSimClustersANN4Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

nextInterestedInSimClustersANN4Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

addressbookInterestedInSimClustersANN4Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

)

def fromParams(

internalId: InternalId,

params: configapi.Params,

): Query = {

// SimClusters common configs

val simClustersModelVersion =

ModelVersions.Enum.enumToSimClustersModelVersionMap(params(GlobalParams.ModelVersionParam))

val simClustersANNConfigId = params(SimClustersANNParams.SimClustersANNConfigId)

val experimentalSimClustersANNConfigId = params(

SimClustersANNParams.ExperimentalSimClustersANNConfigId)

val simClustersANN1ConfigId = params(SimClustersANNParams.SimClustersANN1ConfigId)

val simClustersANN2ConfigId = params(SimClustersANNParams.SimClustersANN2ConfigId)

val simClustersANN3ConfigId = params(SimClustersANNParams.SimClustersANN3ConfigId)

val simClustersANN5ConfigId = params(SimClustersANNParams.SimClustersANN5ConfigId)

val simClustersANN4ConfigId = params(SimClustersANNParams.SimClustersANN4ConfigId)

val simClustersInterestedInMinScore = params(InterestedInParams.MinScoreParam)

val simClustersNextInterestedInMinScore = params(

InterestedInParams.MinScoreSequentialModelParam)

val simClustersAddressBookInterestedInMinScore = params(

InterestedInParams.MinScoreAddressBookParam)

// InterestedIn embeddings parameters

val interestedInEmbedding = params(InterestedInParams.InterestedInEmbeddingIdParam)

val nextInterestedInEmbedding = params(InterestedInParams.NextInterestedInEmbeddingIdParam)

val addressbookInterestedInEmbedding = params(

InterestedInParams.AddressBookInterestedInEmbeddingIdParam)

// Prod SimClustersANN Query

val interestedInSimClustersANNQuery =

SimClustersANNSimilarityEngine.fromParams(

internalId,

interestedInEmbedding.embeddingType,

simClustersModelVersion,

simClustersANNConfigId,

params)

val nextInterestedInSimClustersANNQuery =

SimClustersANNSimilarityEngine.fromParams(

internalId,

nextInterestedInEmbedding.embeddingType,

simClustersModelVersion,

simClustersANNConfigId,

params)

val addressbookInterestedInSimClustersANNQuery =

SimClustersANNSimilarityEngine.fromParams(

internalId,

addressbookInterestedInEmbedding.embeddingType,

simClustersModelVersion,

simClustersANNConfigId,

params)

// Experimental SANN cluster Query

val interestedInExperimentalSimClustersANNQuery =

SimClustersANNSimilarityEngine.fromParams(

internalId,

interestedInEmbedding.embeddingType,

simClustersModelVersion,

experimentalSimClustersANNConfigId,

params)

val nextInterestedInExperimentalSimClustersANNQuery =

SimClustersANNSimilarityEngine.fromParams(

internalId,

nextInterestedInEmbedding.embeddingType,

simClustersModelVersion,

experimentalSimClustersANNConfigId,

params)

val addressbookInterestedInExperimentalSimClustersANNQuery =

SimClustersANNSimilarityEngine.fromParams(

internalId,

addressbookInterestedInEmbedding.embeddingType,

simClustersModelVersion,

experimentalSimClustersANNConfigId,

params)

// SimClusters ANN cluster 1 Query

val interestedInSimClustersANN1Query =

SimClustersANNSimilarityEngine.fromParams(

internalId,

interestedInEmbedding.embeddingType,

simClustersModelVersion,

simClustersANN1ConfigId,

params)

val nextInterestedInSimClustersANN1Query =

SimClustersANNSimilarityEngine.fromParams(

internalId,

nextInterestedInEmbedding.embeddingType,

simClustersModelVersion,

simClustersANN1ConfigId,

params)

val addressbookInterestedInSimClustersANN1Query =

SimClustersANNSimilarityEngine.fromParams(

internalId,

addressbookInterestedInEmbedding.embeddingType,

simClustersModelVersion,

simClustersANN1ConfigId,

params)

// SimClusters ANN cluster 2 Query

val interestedInSimClustersANN2Query =

SimClustersANNSimilarityEngine.fromParams(

internalId,

interestedInEmbedding.embeddingType,

simClustersModelVersion,

simClustersANN2ConfigId,

params)

val nextInterestedInSimClustersANN2Query =

SimClustersANNSimilarityEngine.fromParams(

internalId,

nextInterestedInEmbedding.embeddingType,

simClustersModelVersion,

simClustersANN2ConfigId,

params)

val addressbookInterestedInSimClustersANN2Query =

SimClustersANNSimilarityEngine.fromParams(

internalId,

addressbookInterestedInEmbedding.embeddingType,

simClustersModelVersion,

simClustersANN2ConfigId,

params)

// SimClusters ANN cluster 3 Query

val interestedInSimClustersANN3Query =

SimClustersANNSimilarityEngine.fromParams(

internalId,

interestedInEmbedding.embeddingType,

simClustersModelVersion,

simClustersANN3ConfigId,

params)

val nextInterestedInSimClustersANN3Query =

SimClustersANNSimilarityEngine.fromParams(

internalId,

nextInterestedInEmbedding.embeddingType,

simClustersModelVersion,

simClustersANN3ConfigId,

params)

val addressbookInterestedInSimClustersANN3Query =

SimClustersANNSimilarityEngine.fromParams(

internalId,

addressbookInterestedInEmbedding.embeddingType,

simClustersModelVersion,

simClustersANN3ConfigId,

params)

// SimClusters ANN cluster 5 Query

val interestedInSimClustersANN5Query =

SimClustersANNSimilarityEngine.fromParams(

internalId,

interestedInEmbedding.embeddingType,

simClustersModelVersion,

simClustersANN5ConfigId,

params)

// SimClusters ANN cluster 4 Query

val interestedInSimClustersANN4Query =

SimClustersANNSimilarityEngine.fromParams(

internalId,

interestedInEmbedding.embeddingType,

simClustersModelVersion,

simClustersANN4ConfigId,

params)

val nextInterestedInSimClustersANN5Query =

SimClustersANNSimilarityEngine.fromParams(

internalId,

nextInterestedInEmbedding.embeddingType,

simClustersModelVersion,

simClustersANN5ConfigId,

params)

val nextInterestedInSimClustersANN4Query =

SimClustersANNSimilarityEngine.fromParams(

internalId,

nextInterestedInEmbedding.embeddingType,

simClustersModelVersion,

simClustersANN4ConfigId,

params)

val addressbookInterestedInSimClustersANN5Query =

SimClustersANNSimilarityEngine.fromParams(

internalId,

addressbookInterestedInEmbedding.embeddingType,

simClustersModelVersion,

simClustersANN5ConfigId,

params)

val addressbookInterestedInSimClustersANN4Query =

SimClustersANNSimilarityEngine.fromParams(

internalId,

addressbookInterestedInEmbedding.embeddingType,

simClustersModelVersion,

simClustersANN4ConfigId,

params)

Query(

internalId = internalId,

enableUserInterestedIn = params(InterestedInParams.EnableSourceParam),

enableUserNextInterestedIn = params(InterestedInParams.EnableSourceSequentialModelParam),

enableAddressBookNextInterestedIn = params(InterestedInParams.EnableSourceAddressBookParam),

enableProdSimClustersANNSimilarityEngine =

params(InterestedInParams.EnableProdSimClustersANNParam),

enableExperimentalSimClustersANNSimilarityEngine =

params(InterestedInParams.EnableExperimentalSimClustersANNParam),

enableSimClustersANN1SimilarityEngine = params(InterestedInParams.EnableSimClustersANN1Param),

enableSimClustersANN2SimilarityEngine = params(InterestedInParams.EnableSimClustersANN2Param),

enableSimClustersANN3SimilarityEngine = params(InterestedInParams.EnableSimClustersANN3Param),

enableSimClustersANN5SimilarityEngine = params(InterestedInParams.EnableSimClustersANN5Param),

enableSimClustersANN4SimilarityEngine = params(InterestedInParams.EnableSimClustersANN4Param),

simClustersInterestedInMinScore = simClustersInterestedInMinScore,

simClustersNextInterestedInMinScore = simClustersNextInterestedInMinScore,

simClustersAddressBookInterestedInMinScore = simClustersAddressBookInterestedInMinScore,

interestedInSimClustersANNQuery = interestedInSimClustersANNQuery,

nextInterestedInSimClustersANNQuery = nextInterestedInSimClustersANNQuery,

addressbookInterestedInSimClustersANNQuery = addressbookInterestedInSimClustersANNQuery,

interestedInExperimentalSimClustersANNQuery = interestedInExperimentalSimClustersANNQuery,

nextInterestedInExperimentalSimClustersANNQuery =

nextInterestedInExperimentalSimClustersANNQuery,

addressbookInterestedInExperimentalSimClustersANNQuery =

addressbookInterestedInExperimentalSimClustersANNQuery,

interestedInSimClustersANN1Query = interestedInSimClustersANN1Query,

nextInterestedInSimClustersANN1Query = nextInterestedInSimClustersANN1Query,

addressbookInterestedInSimClustersANN1Query = addressbookInterestedInSimClustersANN1Query,

interestedInSimClustersANN2Query = interestedInSimClustersANN2Query,

nextInterestedInSimClustersANN2Query = nextInterestedInSimClustersANN2Query,

addressbookInterestedInSimClustersANN2Query = addressbookInterestedInSimClustersANN2Query,

interestedInSimClustersANN3Query = interestedInSimClustersANN3Query,

nextInterestedInSimClustersANN3Query = nextInterestedInSimClustersANN3Query,

addressbookInterestedInSimClustersANN3Query = addressbookInterestedInSimClustersANN3Query,

interestedInSimClustersANN5Query = interestedInSimClustersANN5Query,

nextInterestedInSimClustersANN5Query = nextInterestedInSimClustersANN5Query,

addressbookInterestedInSimClustersANN5Query = addressbookInterestedInSimClustersANN5Query,

interestedInSimClustersANN4Query = interestedInSimClustersANN4Query,

nextInterestedInSimClustersANN4Query = nextInterestedInSimClustersANN4Query,

addressbookInterestedInSimClustersANN4Query = addressbookInterestedInSimClustersANN4Query,

)

}

}