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

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

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

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

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

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.RelatedTweetTweetBasedParams

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

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

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

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

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

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

import com.twitter.cr\_mixer.util.InterleaveUtil

import com.twitter.finagle.stats.StatsReceiver

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

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

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

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

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.Named

import javax.inject.Singleton

import scala.collection.mutable.ArrayBuffer

/\*\*

\* This store fetches similar tweets from multiple tweet based candidate sources

\* and combines them using different methods obtained from query params

\*/

@Singleton

case class TweetBasedUnifiedSimilarityEngine(

@Named(ModuleNames.TweetBasedUserTweetGraphSimilarityEngine)

tweetBasedUserTweetGraphSimilarityEngine: StandardSimilarityEngine[

TweetBasedUserTweetGraphSimilarityEngine.Query,

TweetWithScore

],

@Named(ModuleNames.TweetBasedUserVideoGraphSimilarityEngine)

tweetBasedUserVideoGraphSimilarityEngine: StandardSimilarityEngine[

TweetBasedUserVideoGraphSimilarityEngine.Query,

TweetWithScore

],

simClustersANNSimilarityEngine: StandardSimilarityEngine[

SimClustersANNSimilarityEngine.Query,

TweetWithScore

],

@Named(ModuleNames.TweetBasedQigSimilarityEngine)

tweetBasedQigSimilarTweetsSimilarityEngine: StandardSimilarityEngine[

TweetBasedQigSimilarityEngine.Query,

TweetWithScore

],

@Named(ModuleNames.TweetBasedTwHINANNSimilarityEngine)

tweetBasedTwHINANNSimilarityEngine: HnswANNSimilarityEngine,

statsReceiver: StatsReceiver)

extends ReadableStore[

TweetBasedUnifiedSimilarityEngine.Query,

Seq[TweetWithCandidateGenerationInfo]

] {

import TweetBasedUnifiedSimilarityEngine.\_

private val stats = statsReceiver.scope(this.getClass.getSimpleName)

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

override def get(

query: Query

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

query.sourceInfo.internalId match {

case \_: InternalId.TweetId =>

StatsUtil.trackOptionItemsStats(fetchCandidatesStat) {

val twhinQuery =

HnswANNEngineQuery(

sourceId = query.sourceInfo.internalId,

modelId = query.twhinModelId,

params = query.params)

val utgCandidatesFut =

if (query.enableUtg)

tweetBasedUserTweetGraphSimilarityEngine.getCandidates(query.utgQuery)

else Future.None

val uvgCandidatesFut =

if (query.enableUvg)

tweetBasedUserVideoGraphSimilarityEngine.getCandidates(query.uvgQuery)

else Future.None

val sannCandidatesFut = if (query.enableSimClustersANN) {

simClustersANNSimilarityEngine.getCandidates(query.simClustersANNQuery)

} else Future.None

val sann1CandidatesFut =

if (query.enableSimClustersANN1) {

simClustersANNSimilarityEngine.getCandidates(query.simClustersANN1Query)

} else Future.None

val sann2CandidatesFut =

if (query.enableSimClustersANN2) {

simClustersANNSimilarityEngine.getCandidates(query.simClustersANN2Query)

} else Future.None

val sann3CandidatesFut =

if (query.enableSimClustersANN3) {

simClustersANNSimilarityEngine.getCandidates(query.simClustersANN3Query)

} else Future.None

val sann5CandidatesFut =

if (query.enableSimClustersANN5) {

simClustersANNSimilarityEngine.getCandidates(query.simClustersANN5Query)

} else Future.None

val sann4CandidatesFut =

if (query.enableSimClustersANN4) {

simClustersANNSimilarityEngine.getCandidates(query.simClustersANN4Query)

} else Future.None

val experimentalSANNCandidatesFut =

if (query.enableExperimentalSimClustersANN) {

simClustersANNSimilarityEngine.getCandidates(query.experimentalSimClustersANNQuery)

} else Future.None

val qigCandidatesFut =

if (query.enableQig)

tweetBasedQigSimilarTweetsSimilarityEngine.getCandidates(query.qigQuery)

else Future.None

val twHINCandidateFut = if (query.enableTwHIN) {

tweetBasedTwHINANNSimilarityEngine.getCandidates(twhinQuery)

} else Future.None

Future

.join(

utgCandidatesFut,

sannCandidatesFut,

sann1CandidatesFut,

sann2CandidatesFut,

sann3CandidatesFut,

sann5CandidatesFut,

sann4CandidatesFut,

experimentalSANNCandidatesFut,

qigCandidatesFut,

twHINCandidateFut,

uvgCandidatesFut

).map {

case (

userTweetGraphCandidates,

simClustersANNCandidates,

simClustersANN1Candidates,

simClustersANN2Candidates,

simClustersANN3Candidates,

simClustersANN5Candidates,

simClustersANN4Candidates,

experimentalSANNCandidates,

qigSimilarTweetsCandidates,

twhinCandidates,

userVideoGraphCandidates) =>

val filteredUTGTweets =

userTweetGraphFilter(userTweetGraphCandidates.toSeq.flatten)

val filteredUVGTweets =

userVideoGraphFilter(userVideoGraphCandidates.toSeq.flatten)

val filteredSANNTweets = simClustersCandidateMinScoreFilter(

simClustersANNCandidates.toSeq.flatten,

query.simClustersMinScore,

query.simClustersANNQuery.storeQuery.simClustersANNConfigId)

val filteredSANN1Tweets = simClustersCandidateMinScoreFilter(

simClustersANN1Candidates.toSeq.flatten,

query.simClustersMinScore,

query.simClustersANN1Query.storeQuery.simClustersANNConfigId)

val filteredSANN2Tweets = simClustersCandidateMinScoreFilter(

simClustersANN2Candidates.toSeq.flatten,

query.simClustersMinScore,

query.simClustersANN2Query.storeQuery.simClustersANNConfigId)

val filteredSANN3Tweets = simClustersCandidateMinScoreFilter(

simClustersANN3Candidates.toSeq.flatten,

query.simClustersMinScore,

query.simClustersANN3Query.storeQuery.simClustersANNConfigId)

val filteredSANN4Tweets = simClustersCandidateMinScoreFilter(

simClustersANN4Candidates.toSeq.flatten,

query.simClustersMinScore,

query.simClustersANN4Query.storeQuery.simClustersANNConfigId)

val filteredSANN5Tweets = simClustersCandidateMinScoreFilter(

simClustersANN5Candidates.toSeq.flatten,

query.simClustersMinScore,

query.simClustersANN5Query.storeQuery.simClustersANNConfigId)

val filteredExperimentalSANNTweets = simClustersCandidateMinScoreFilter(

experimentalSANNCandidates.toSeq.flatten,

query.simClustersVideoBasedMinScore,

query.experimentalSimClustersANNQuery.storeQuery.simClustersANNConfigId)

val filteredQigTweets = qigSimilarTweetsFilter(

qigSimilarTweetsCandidates.toSeq.flatten,

query.qigMaxTweetAgeHours,

query.qigMaxNumSimilarTweets

)

val filteredTwHINTweets = twhinFilter(

twhinCandidates.toSeq.flatten.sortBy(-\_.score),

query.twhinMaxTweetAgeHours,

tweetBasedTwHINANNSimilarityEngine.getScopedStats

)

val utgTweetsWithCGInfo = filteredUTGTweets.map { tweetWithScore =>

val similarityEngineInfo = TweetBasedUserTweetGraphSimilarityEngine

.toSimilarityEngineInfo(tweetWithScore.score)

TweetWithCandidateGenerationInfo(

tweetWithScore.tweetId,

CandidateGenerationInfo(

Some(query.sourceInfo),

similarityEngineInfo,

Seq(similarityEngineInfo)

))

}

val uvgTweetsWithCGInfo = filteredUVGTweets.map { tweetWithScore =>

val similarityEngineInfo = TweetBasedUserVideoGraphSimilarityEngine

.toSimilarityEngineInfo(tweetWithScore.score)

TweetWithCandidateGenerationInfo(

tweetWithScore.tweetId,

CandidateGenerationInfo(

Some(query.sourceInfo),

similarityEngineInfo,

Seq(similarityEngineInfo)

))

}

val sannTweetsWithCGInfo = filteredSANNTweets.map { tweetWithScore =>

val similarityEngineInfo = SimClustersANNSimilarityEngine

.toSimilarityEngineInfo(query.simClustersANNQuery, tweetWithScore.score)

TweetWithCandidateGenerationInfo(

tweetWithScore.tweetId,

CandidateGenerationInfo(

Some(query.sourceInfo),

similarityEngineInfo,

Seq(similarityEngineInfo)

))

}

val sann1TweetsWithCGInfo = filteredSANN1Tweets.map { tweetWithScore =>

val similarityEngineInfo = SimClustersANNSimilarityEngine

.toSimilarityEngineInfo(query.simClustersANN1Query, tweetWithScore.score)

TweetWithCandidateGenerationInfo(

tweetWithScore.tweetId,

CandidateGenerationInfo(

Some(query.sourceInfo),

similarityEngineInfo,

Seq(similarityEngineInfo)

))

}

val sann2TweetsWithCGInfo = filteredSANN2Tweets.map { tweetWithScore =>

val similarityEngineInfo = SimClustersANNSimilarityEngine

.toSimilarityEngineInfo(query.simClustersANN2Query, tweetWithScore.score)

TweetWithCandidateGenerationInfo(

tweetWithScore.tweetId,

CandidateGenerationInfo(

Some(query.sourceInfo),

similarityEngineInfo,

Seq(similarityEngineInfo)

))

}

val sann3TweetsWithCGInfo = filteredSANN3Tweets.map { tweetWithScore =>

val similarityEngineInfo = SimClustersANNSimilarityEngine

.toSimilarityEngineInfo(query.simClustersANN3Query, tweetWithScore.score)

TweetWithCandidateGenerationInfo(

tweetWithScore.tweetId,

CandidateGenerationInfo(

Some(query.sourceInfo),

similarityEngineInfo,

Seq(similarityEngineInfo)

))

}

val sann4TweetsWithCGInfo = filteredSANN4Tweets.map { tweetWithScore =>

val similarityEngineInfo = SimClustersANNSimilarityEngine

.toSimilarityEngineInfo(query.simClustersANN4Query, tweetWithScore.score)

TweetWithCandidateGenerationInfo(

tweetWithScore.tweetId,

CandidateGenerationInfo(

Some(query.sourceInfo),

similarityEngineInfo,

Seq(similarityEngineInfo)

))

}

val sann5TweetsWithCGInfo = filteredSANN5Tweets.map { tweetWithScore =>

val similarityEngineInfo = SimClustersANNSimilarityEngine

.toSimilarityEngineInfo(query.simClustersANN5Query, tweetWithScore.score)

TweetWithCandidateGenerationInfo(

tweetWithScore.tweetId,

CandidateGenerationInfo(

Some(query.sourceInfo),

similarityEngineInfo,

Seq(similarityEngineInfo)

))

}

val experimentalSANNTweetsWithCGInfo = filteredExperimentalSANNTweets.map {

tweetWithScore =>

val similarityEngineInfo = SimClustersANNSimilarityEngine

.toSimilarityEngineInfo(

query.experimentalSimClustersANNQuery,

tweetWithScore.score)

TweetWithCandidateGenerationInfo(

tweetWithScore.tweetId,

CandidateGenerationInfo(

Some(query.sourceInfo),

similarityEngineInfo,

Seq(similarityEngineInfo)

))

}

val qigTweetsWithCGInfo = filteredQigTweets.map { tweetWithScore =>

val similarityEngineInfo = TweetBasedQigSimilarityEngine

.toSimilarityEngineInfo(tweetWithScore.score)

TweetWithCandidateGenerationInfo(

tweetWithScore.tweetId,

CandidateGenerationInfo(

Some(query.sourceInfo),

similarityEngineInfo,

Seq(similarityEngineInfo)

))

}

val twHINTweetsWithCGInfo = filteredTwHINTweets.map { tweetWithScore =>

val similarityEngineInfo = tweetBasedTwHINANNSimilarityEngine

.toSimilarityEngineInfo(twhinQuery, tweetWithScore.score)

TweetWithCandidateGenerationInfo(

tweetWithScore.tweetId,

CandidateGenerationInfo(

Some(query.sourceInfo),

similarityEngineInfo,

Seq(similarityEngineInfo)

))

}

val candidateSourcesToBeInterleaved =

ArrayBuffer[Seq[TweetWithCandidateGenerationInfo]](

sannTweetsWithCGInfo,

experimentalSANNTweetsWithCGInfo,

sann1TweetsWithCGInfo,

sann2TweetsWithCGInfo,

sann3TweetsWithCGInfo,

sann5TweetsWithCGInfo,

sann4TweetsWithCGInfo,

qigTweetsWithCGInfo,

uvgTweetsWithCGInfo,

utgTweetsWithCGInfo,

twHINTweetsWithCGInfo

)

val interleavedCandidates =

InterleaveUtil.interleave(candidateSourcesToBeInterleaved)

val unifiedCandidatesWithUnifiedCGInfo =

interleavedCandidates.map { candidate =>

/\*\*\*

\* when a candidate was made by interleave/keepGivenOrder,

\* then we apply getTweetBasedUnifiedCGInfo() to override with the unified CGInfo

\*

\* we'll not have ALL SEs that generated the tweet

\* in contributingSE list for interleave. We only have the chosen SE available.

\*/

TweetWithCandidateGenerationInfo(

tweetId = candidate.tweetId,

candidateGenerationInfo = getTweetBasedUnifiedCGInfo(

candidate.candidateGenerationInfo.sourceInfoOpt,

candidate.getSimilarityScore,

candidate.candidateGenerationInfo.contributingSimilarityEngines

) // getSimilarityScore comes from either unifiedScore or single score

)

}

stats

.stat("unified\_candidate\_size").add(unifiedCandidatesWithUnifiedCGInfo.size)

val truncatedCandidates =

unifiedCandidatesWithUnifiedCGInfo.take(query.maxCandidateNumPerSourceKey)

stats.stat("truncatedCandidates\_size").add(truncatedCandidates.size)

Some(truncatedCandidates)

}

}

case \_ =>

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

Future.None

}

}

private def simClustersCandidateMinScoreFilter(

simClustersAnnCandidates: Seq[TweetWithScore],

simClustersMinScore: Double,

simClustersANNConfigId: String

): Seq[TweetWithScore] = {

val filteredCandidates = simClustersAnnCandidates

.filter { candidate =>

candidate.score > simClustersMinScore

}

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)

}

}

/\*\* Returns a list of tweets that are generated less than `maxTweetAgeHours` hours ago \*/

private def tweetAgeFilter(

candidates: Seq[TweetWithScore],

maxTweetAgeHours: Duration

): Seq[TweetWithScore] = {

// Tweet IDs are approximately chronological (see http://go/snowflake),

// so we are building the earliest tweet id once

// The per-candidate logic here then be candidate.tweetId > earliestPermittedTweetId, which is far cheaper.

val earliestTweetId = SnowflakeId.firstIdFor(Time.now - maxTweetAgeHours)

candidates.filter { candidate => candidate.tweetId >= earliestTweetId }

}

private def twhinFilter(

twhinCandidates: Seq[TweetWithScore],

twhinMaxTweetAgeHours: Duration,

simEngineStats: StatsReceiver

): Seq[TweetWithScore] = {

simEngineStats.stat("twhinCandidates\_size").add(twhinCandidates.size)

val candidates = twhinCandidates.map { candidate =>

TweetWithScore(candidate.tweetId, candidate.score)

}

val filteredCandidates = tweetAgeFilter(candidates, twhinMaxTweetAgeHours)

simEngineStats.stat("filteredTwhinCandidates\_size").add(filteredCandidates.size)

if (filteredCandidates.isEmpty) simEngineStats.counter("emptyFilteredTwhinCandidates").incr()

filteredCandidates

}

/\*\* A no-op filter as UTG filtering already happens on UTG service side \*/

private def userTweetGraphFilter(

userTweetGraphCandidates: Seq[TweetWithScore]

): Seq[TweetWithScore] = {

val filteredCandidates = userTweetGraphCandidates

stats.stat("userTweetGraphCandidates\_size").add(userTweetGraphCandidates.size)

if (filteredCandidates.isEmpty) stats.counter("emptyFilteredUserTweetGraphCandidates").incr()

filteredCandidates.map { candidate =>

TweetWithScore(candidate.tweetId, candidate.score)

}

}

/\*\* A no-op filter as UVG filtering already happens on UVG service side \*/

private def userVideoGraphFilter(

userVideoGraphCandidates: Seq[TweetWithScore]

): Seq[TweetWithScore] = {

val filteredCandidates = userVideoGraphCandidates

stats.stat("userVideoGraphCandidates\_size").add(userVideoGraphCandidates.size)

if (filteredCandidates.isEmpty) stats.counter("emptyFilteredUserVideoGraphCandidates").incr()

filteredCandidates.map { candidate =>

TweetWithScore(candidate.tweetId, candidate.score)

}

}

private def qigSimilarTweetsFilter(

qigSimilarTweetsCandidates: Seq[TweetWithScore],

qigMaxTweetAgeHours: Duration,

qigMaxNumSimilarTweets: Int

): Seq[TweetWithScore] = {

val ageFilteredCandidates = tweetAgeFilter(qigSimilarTweetsCandidates, qigMaxTweetAgeHours)

stats.stat("ageFilteredQigSimilarTweetsCandidates\_size").add(ageFilteredCandidates.size)

val filteredCandidates = ageFilteredCandidates.take(qigMaxNumSimilarTweets)

if (filteredCandidates.isEmpty) stats.counter("emptyFilteredQigSimilarTweetsCandidates").incr()

filteredCandidates

}

/\*\*\*

\* Every candidate will have the CG Info with TweetBasedUnifiedSimilarityEngine

\* as they are generated by a composite of Similarity Engines.

\* Additionally, we store the contributing SEs (eg., SANN, UTG).

\*/

private def getTweetBasedUnifiedCGInfo(

sourceInfoOpt: Option[SourceInfo],

unifiedScore: Double,

contributingSimilarityEngines: Seq[SimilarityEngineInfo]

): CandidateGenerationInfo = {

CandidateGenerationInfo(

sourceInfoOpt,

SimilarityEngineInfo(

similarityEngineType = SimilarityEngineType.TweetBasedUnifiedSimilarityEngine,

modelId = None, // We do not assign modelId for a unified similarity engine

score = Some(unifiedScore)

),

contributingSimilarityEngines

)

}

}

object TweetBasedUnifiedSimilarityEngine {

case class Query(

sourceInfo: SourceInfo,

maxCandidateNumPerSourceKey: Int,

enableSimClustersANN: Boolean,

simClustersANNQuery: EngineQuery[SimClustersANNSimilarityEngine.Query],

enableExperimentalSimClustersANN: Boolean,

experimentalSimClustersANNQuery: EngineQuery[SimClustersANNSimilarityEngine.Query],

enableSimClustersANN1: Boolean,

simClustersANN1Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

enableSimClustersANN2: Boolean,

simClustersANN2Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

enableSimClustersANN3: Boolean,

simClustersANN3Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

enableSimClustersANN5: Boolean,

simClustersANN5Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

enableSimClustersANN4: Boolean,

simClustersANN4Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

simClustersMinScore: Double,

simClustersVideoBasedMinScore: Double,

twhinModelId: String,

enableTwHIN: Boolean,

twhinMaxTweetAgeHours: Duration,

qigMaxTweetAgeHours: Duration,

qigMaxNumSimilarTweets: Int,

enableUtg: Boolean,

utgQuery: EngineQuery[TweetBasedUserTweetGraphSimilarityEngine.Query],

enableUvg: Boolean,

uvgQuery: EngineQuery[TweetBasedUserVideoGraphSimilarityEngine.Query],

enableQig: Boolean,

qigQuery: EngineQuery[TweetBasedQigSimilarityEngine.Query],

params: configapi.Params)

def fromParams(

sourceInfo: SourceInfo,

params: configapi.Params,

): EngineQuery[Query] = {

// SimClusters

val enableSimClustersANN =

params(TweetBasedCandidateGenerationParams.EnableSimClustersANNParam)

val simClustersModelVersion =

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

val simClustersMinScore = params(TweetBasedCandidateGenerationParams.SimClustersMinScoreParam)

val simClustersVideoBasedMinScore = params(

TweetBasedCandidateGenerationParams.SimClustersVideoBasedMinScoreParam)

val simClustersANNConfigId = params(SimClustersANNParams.SimClustersANNConfigId)

// SimClusters - Experimental SANN Similarity Engine (Video based SE)

val enableExperimentalSimClustersANN =

params(TweetBasedCandidateGenerationParams.EnableExperimentalSimClustersANNParam)

val experimentalSimClustersANNConfigId = params(

SimClustersANNParams.ExperimentalSimClustersANNConfigId)

// SimClusters - SANN cluster 1 Similarity Engine

val enableSimClustersANN1 =

params(TweetBasedCandidateGenerationParams.EnableSimClustersANN1Param)

val simClustersANN1ConfigId = params(SimClustersANNParams.SimClustersANN1ConfigId)

// SimClusters - SANN cluster 2 Similarity Engine

val enableSimClustersANN2 =

params(TweetBasedCandidateGenerationParams.EnableSimClustersANN2Param)

val simClustersANN2ConfigId = params(SimClustersANNParams.SimClustersANN2ConfigId)

// SimClusters - SANN cluster 3 Similarity Engine

val enableSimClustersANN3 =

params(TweetBasedCandidateGenerationParams.EnableSimClustersANN3Param)

val simClustersANN3ConfigId = params(SimClustersANNParams.SimClustersANN3ConfigId)

// SimClusters - SANN cluster 5 Similarity Engine

val enableSimClustersANN5 =

params(TweetBasedCandidateGenerationParams.EnableSimClustersANN5Param)

val simClustersANN5ConfigId = params(SimClustersANNParams.SimClustersANN5ConfigId)

// SimClusters - SANN cluster 4 Similarity Engine

val enableSimClustersANN4 =

params(TweetBasedCandidateGenerationParams.EnableSimClustersANN4Param)

val simClustersANN4ConfigId = params(SimClustersANNParams.SimClustersANN4ConfigId)

// SimClusters ANN Queries for different SANN clusters

val simClustersANNQuery = SimClustersANNSimilarityEngine.fromParams(

sourceInfo.internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

simClustersANNConfigId,

params

)

val experimentalSimClustersANNQuery = SimClustersANNSimilarityEngine.fromParams(

sourceInfo.internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

experimentalSimClustersANNConfigId,

params

)

val simClustersANN1Query = SimClustersANNSimilarityEngine.fromParams(

sourceInfo.internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

simClustersANN1ConfigId,

params

)

val simClustersANN2Query = SimClustersANNSimilarityEngine.fromParams(

sourceInfo.internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

simClustersANN2ConfigId,

params

)

val simClustersANN3Query = SimClustersANNSimilarityEngine.fromParams(

sourceInfo.internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

simClustersANN3ConfigId,

params

)

val simClustersANN5Query = SimClustersANNSimilarityEngine.fromParams(

sourceInfo.internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

simClustersANN5ConfigId,

params

)

val simClustersANN4Query = SimClustersANNSimilarityEngine.fromParams(

sourceInfo.internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

simClustersANN4ConfigId,

params

)

// TweetBasedCandidateGeneration

val maxCandidateNumPerSourceKey = params(GlobalParams.MaxCandidateNumPerSourceKeyParam)

// TwHIN

val twhinModelId = params(TweetBasedTwHINParams.ModelIdParam)

val enableTwHIN =

params(TweetBasedCandidateGenerationParams.EnableTwHINParam)

val twhinMaxTweetAgeHours = params(GlobalParams.MaxTweetAgeHoursParam)

// QIG

val enableQig =

params(TweetBasedCandidateGenerationParams.EnableQigSimilarTweetsParam)

val qigMaxTweetAgeHours = params(GlobalParams.MaxTweetAgeHoursParam)

val qigMaxNumSimilarTweets = params(

TweetBasedCandidateGenerationParams.QigMaxNumSimilarTweetsParam)

// UTG

val enableUtg =

params(TweetBasedCandidateGenerationParams.EnableUTGParam)

// UVG

val enableUvg =

params(TweetBasedCandidateGenerationParams.EnableUVGParam)

EngineQuery(

Query(

sourceInfo = sourceInfo,

maxCandidateNumPerSourceKey = maxCandidateNumPerSourceKey,

enableSimClustersANN = enableSimClustersANN,

simClustersANNQuery = simClustersANNQuery,

enableExperimentalSimClustersANN = enableExperimentalSimClustersANN,

experimentalSimClustersANNQuery = experimentalSimClustersANNQuery,

enableSimClustersANN1 = enableSimClustersANN1,

simClustersANN1Query = simClustersANN1Query,

enableSimClustersANN2 = enableSimClustersANN2,

simClustersANN2Query = simClustersANN2Query,

enableSimClustersANN3 = enableSimClustersANN3,

simClustersANN3Query = simClustersANN3Query,

enableSimClustersANN5 = enableSimClustersANN5,

simClustersANN5Query = simClustersANN5Query,

enableSimClustersANN4 = enableSimClustersANN4,

simClustersANN4Query = simClustersANN4Query,

simClustersMinScore = simClustersMinScore,

simClustersVideoBasedMinScore = simClustersVideoBasedMinScore,

twhinModelId = twhinModelId,

enableTwHIN = enableTwHIN,

twhinMaxTweetAgeHours = twhinMaxTweetAgeHours,

qigMaxTweetAgeHours = qigMaxTweetAgeHours,

qigMaxNumSimilarTweets = qigMaxNumSimilarTweets,

enableUtg = enableUtg,

utgQuery = TweetBasedUserTweetGraphSimilarityEngine

.fromParams(sourceInfo.internalId, params),

enableQig = enableQig,

qigQuery = TweetBasedQigSimilarityEngine.fromParams(sourceInfo.internalId, params),

enableUvg = enableUvg,

uvgQuery =

TweetBasedUserVideoGraphSimilarityEngine.fromParams(sourceInfo.internalId, params),

params = params

),

params

)

}

def fromParamsForRelatedTweet(

internalId: InternalId,

params: configapi.Params,

): EngineQuery[Query] = {

// SimClusters

val enableSimClustersANN = params(RelatedTweetTweetBasedParams.EnableSimClustersANNParam)

val simClustersModelVersion =

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

val simClustersMinScore = params(RelatedTweetTweetBasedParams.SimClustersMinScoreParam)

val simClustersANNConfigId = params(SimClustersANNParams.SimClustersANNConfigId)

val enableExperimentalSimClustersANN =

params(RelatedTweetTweetBasedParams.EnableExperimentalSimClustersANNParam)

val experimentalSimClustersANNConfigId = params(

SimClustersANNParams.ExperimentalSimClustersANNConfigId)

// SimClusters - SANN cluster 1 Similarity Engine

val enableSimClustersANN1 = params(RelatedTweetTweetBasedParams.EnableSimClustersANN1Param)

val simClustersANN1ConfigId = params(SimClustersANNParams.SimClustersANN1ConfigId)

// SimClusters - SANN cluster 2 Similarity Engine

val enableSimClustersANN2 = params(RelatedTweetTweetBasedParams.EnableSimClustersANN2Param)

val simClustersANN2ConfigId = params(SimClustersANNParams.SimClustersANN2ConfigId)

// SimClusters - SANN cluster 3 Similarity Engine

val enableSimClustersANN3 = params(RelatedTweetTweetBasedParams.EnableSimClustersANN3Param)

val simClustersANN3ConfigId = params(SimClustersANNParams.SimClustersANN3ConfigId)

// SimClusters - SANN cluster 5 Similarity Engine

val enableSimClustersANN5 = params(RelatedTweetTweetBasedParams.EnableSimClustersANN5Param)

val simClustersANN5ConfigId = params(SimClustersANNParams.SimClustersANN5ConfigId)

// SimClusters - SANN cluster 4 Similarity Engine

val enableSimClustersANN4 = params(RelatedTweetTweetBasedParams.EnableSimClustersANN4Param)

val simClustersANN4ConfigId = params(SimClustersANNParams.SimClustersANN4ConfigId)

// SimClusters ANN Queries for different SANN clusters

val simClustersANNQuery = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

simClustersANNConfigId,

params

)

val experimentalSimClustersANNQuery = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

experimentalSimClustersANNConfigId,

params

)

val simClustersANN1Query = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

simClustersANN1ConfigId,

params

)

val simClustersANN2Query = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

simClustersANN2ConfigId,

params

)

val simClustersANN3Query = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

simClustersANN3ConfigId,

params

)

val simClustersANN5Query = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

simClustersANN5ConfigId,

params

)

val simClustersANN4Query = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

simClustersANN4ConfigId,

params

)

// TweetBasedCandidateGeneration

val maxCandidateNumPerSourceKey = params(GlobalParams.MaxCandidateNumPerSourceKeyParam)

// TwHIN

val twhinModelId = params(TweetBasedTwHINParams.ModelIdParam)

val enableTwHIN = params(RelatedTweetTweetBasedParams.EnableTwHINParam)

val twhinMaxTweetAgeHours = params(GlobalParams.MaxTweetAgeHoursParam)

// QIG

val enableQig = params(RelatedTweetTweetBasedParams.EnableQigSimilarTweetsParam)

val qigMaxTweetAgeHours = params(GlobalParams.MaxTweetAgeHoursParam)

val qigMaxNumSimilarTweets = params(

TweetBasedCandidateGenerationParams.QigMaxNumSimilarTweetsParam)

// UTG

val enableUtg = params(RelatedTweetTweetBasedParams.EnableUTGParam)

// UVG

val enableUvg = params(RelatedTweetTweetBasedParams.EnableUVGParam)

// SourceType.RequestTweetId is a placeholder.

val sourceInfo = SourceInfo(SourceType.RequestTweetId, internalId, None)

EngineQuery(

Query(

sourceInfo = sourceInfo,

maxCandidateNumPerSourceKey = maxCandidateNumPerSourceKey,

enableSimClustersANN = enableSimClustersANN,

simClustersMinScore = simClustersMinScore,

simClustersVideoBasedMinScore = simClustersMinScore,

simClustersANNQuery = simClustersANNQuery,

enableExperimentalSimClustersANN = enableExperimentalSimClustersANN,

experimentalSimClustersANNQuery = experimentalSimClustersANNQuery,

enableSimClustersANN1 = enableSimClustersANN1,

simClustersANN1Query = simClustersANN1Query,

enableSimClustersANN2 = enableSimClustersANN2,

simClustersANN2Query = simClustersANN2Query,

enableSimClustersANN3 = enableSimClustersANN3,

simClustersANN3Query = simClustersANN3Query,

enableSimClustersANN5 = enableSimClustersANN5,

simClustersANN5Query = simClustersANN5Query,

enableSimClustersANN4 = enableSimClustersANN4,

simClustersANN4Query = simClustersANN4Query,

twhinModelId = twhinModelId,

enableTwHIN = enableTwHIN,

twhinMaxTweetAgeHours = twhinMaxTweetAgeHours,

qigMaxTweetAgeHours = qigMaxTweetAgeHours,

qigMaxNumSimilarTweets = qigMaxNumSimilarTweets,

enableUtg = enableUtg,

utgQuery = TweetBasedUserTweetGraphSimilarityEngine

.fromParams(sourceInfo.internalId, params),

enableQig = enableQig,

qigQuery = TweetBasedQigSimilarityEngine.fromParams(sourceInfo.internalId, params),

enableUvg = enableUvg,

uvgQuery =

TweetBasedUserVideoGraphSimilarityEngine.fromParams(sourceInfo.internalId, params),

params = params,

),

params

)

}

def fromParamsForRelatedVideoTweet(

internalId: InternalId,

params: configapi.Params,

): EngineQuery[Query] = {

// SimClusters

val enableSimClustersANN = params(RelatedVideoTweetTweetBasedParams.EnableSimClustersANNParam)

val simClustersModelVersion =

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

val simClustersMinScore = params(RelatedVideoTweetTweetBasedParams.SimClustersMinScoreParam)

val simClustersANNConfigId = params(SimClustersANNParams.SimClustersANNConfigId)

val enableExperimentalSimClustersANN = params(

RelatedVideoTweetTweetBasedParams.EnableExperimentalSimClustersANNParam)

val experimentalSimClustersANNConfigId = params(

SimClustersANNParams.ExperimentalSimClustersANNConfigId)

// SimClusters - SANN cluster 1 Similarity Engine

val enableSimClustersANN1 = params(RelatedVideoTweetTweetBasedParams.EnableSimClustersANN1Param)

val simClustersANN1ConfigId = params(SimClustersANNParams.SimClustersANN1ConfigId)

// SimClusters - SANN cluster 2 Similarity Engine

val enableSimClustersANN2 = params(RelatedVideoTweetTweetBasedParams.EnableSimClustersANN2Param)

val simClustersANN2ConfigId = params(SimClustersANNParams.SimClustersANN2ConfigId)

// SimClusters - SANN cluster 3 Similarity Engine

val enableSimClustersANN3 = params(RelatedVideoTweetTweetBasedParams.EnableSimClustersANN3Param)

val simClustersANN3ConfigId = params(SimClustersANNParams.SimClustersANN3ConfigId)

// SimClusters - SANN cluster 5 Similarity Engine

val enableSimClustersANN5 = params(RelatedVideoTweetTweetBasedParams.EnableSimClustersANN5Param)

val simClustersANN5ConfigId = params(SimClustersANNParams.SimClustersANN5ConfigId)

// SimClusters - SANN cluster 4 Similarity Engine

val enableSimClustersANN4 = params(RelatedVideoTweetTweetBasedParams.EnableSimClustersANN4Param)

val simClustersANN4ConfigId = params(SimClustersANNParams.SimClustersANN4ConfigId)

// SimClusters ANN Queries for different SANN clusters

val simClustersANNQuery = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

simClustersANNConfigId,

params

)

val experimentalSimClustersANNQuery = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

experimentalSimClustersANNConfigId,

params

)

val simClustersANN1Query = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

simClustersANN1ConfigId,

params

)

val simClustersANN2Query = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

simClustersANN2ConfigId,

params

)

val simClustersANN3Query = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

simClustersANN3ConfigId,

params

)

val simClustersANN5Query = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

simClustersANN5ConfigId,

params

)

val simClustersANN4Query = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.LogFavLongestL2EmbeddingTweet,

simClustersModelVersion,

simClustersANN4ConfigId,

params

)

// TweetBasedCandidateGeneration

val maxCandidateNumPerSourceKey = params(GlobalParams.MaxCandidateNumPerSourceKeyParam)

// TwHIN

val twhinModelId = params(TweetBasedTwHINParams.ModelIdParam)

val enableTwHIN = params(RelatedVideoTweetTweetBasedParams.EnableTwHINParam)

val twhinMaxTweetAgeHours = params(GlobalParams.MaxTweetAgeHoursParam)

// QIG

val enableQig = params(RelatedVideoTweetTweetBasedParams.EnableQigSimilarTweetsParam)

val qigMaxTweetAgeHours = params(GlobalParams.MaxTweetAgeHoursParam)

val qigMaxNumSimilarTweets = params(

TweetBasedCandidateGenerationParams.QigMaxNumSimilarTweetsParam)

// UTG

val enableUtg = params(RelatedVideoTweetTweetBasedParams.EnableUTGParam)

// SourceType.RequestTweetId is a placeholder.

val sourceInfo = SourceInfo(SourceType.RequestTweetId, internalId, None)

val enableUvg = params(RelatedVideoTweetTweetBasedParams.EnableUVGParam)

EngineQuery(

Query(

sourceInfo = sourceInfo,

maxCandidateNumPerSourceKey = maxCandidateNumPerSourceKey,

enableSimClustersANN = enableSimClustersANN,

simClustersMinScore = simClustersMinScore,

simClustersVideoBasedMinScore = simClustersMinScore,

simClustersANNQuery = simClustersANNQuery,

enableExperimentalSimClustersANN = enableExperimentalSimClustersANN,

experimentalSimClustersANNQuery = experimentalSimClustersANNQuery,

enableSimClustersANN1 = enableSimClustersANN1,

simClustersANN1Query = simClustersANN1Query,

enableSimClustersANN2 = enableSimClustersANN2,

simClustersANN2Query = simClustersANN2Query,

enableSimClustersANN3 = enableSimClustersANN3,

simClustersANN3Query = simClustersANN3Query,

enableSimClustersANN5 = enableSimClustersANN5,

simClustersANN5Query = simClustersANN5Query,

enableSimClustersANN4 = enableSimClustersANN4,

simClustersANN4Query = simClustersANN4Query,

twhinModelId = twhinModelId,

enableTwHIN = enableTwHIN,

twhinMaxTweetAgeHours = twhinMaxTweetAgeHours,

qigMaxTweetAgeHours = qigMaxTweetAgeHours,

qigMaxNumSimilarTweets = qigMaxNumSimilarTweets,

enableUtg = enableUtg,

utgQuery = TweetBasedUserTweetGraphSimilarityEngine

.fromParams(sourceInfo.internalId, params),

enableUvg = enableUvg,

uvgQuery =

TweetBasedUserVideoGraphSimilarityEngine.fromParams(sourceInfo.internalId, params),

enableQig = enableQig,

qigQuery = TweetBasedQigSimilarityEngine.fromParams(sourceInfo.internalId, params),

params = params

),

params

)

}

}