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.ProducerBasedCandidateGenerationParams

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

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

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

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.storehaus.ReadableStore

import com.twitter.timelines.configapi

import com.twitter.util.Duration

import com.twitter.util.Future

import javax.inject.Named

import javax.inject.Singleton

import scala.collection.mutable.ArrayBuffer

/\*\*

\* This store looks for similar tweets from UserTweetGraph for a Source ProducerId

\* For a query producerId,User Tweet Graph (UTG),

\* lets us find out which tweets the query producer's followers co-engaged

\*/

@Singleton

case class ProducerBasedUnifiedSimilarityEngine(

@Named(ModuleNames.ProducerBasedUserTweetGraphSimilarityEngine)

producerBasedUserTweetGraphSimilarityEngine: StandardSimilarityEngine[

ProducerBasedUserTweetGraphSimilarityEngine.Query,

TweetWithScore

],

simClustersANNSimilarityEngine: StandardSimilarityEngine[

SimClustersANNSimilarityEngine.Query,

TweetWithScore

],

statsReceiver: StatsReceiver)

extends ReadableStore[ProducerBasedUnifiedSimilarityEngine.Query, Seq[

TweetWithCandidateGenerationInfo

]] {

import ProducerBasedUnifiedSimilarityEngine.\_

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.UserId =>

StatsUtil.trackOptionItemsStats(fetchCandidatesStat) {

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 sann4CandidatesFut =

if (query.enableSimClustersANN4) {

simClustersANNSimilarityEngine.getCandidates(query.simClustersANN4Query)

} else Future.None

val sann5CandidatesFut =

if (query.enableSimClustersANN5) {

simClustersANNSimilarityEngine.getCandidates(query.simClustersANN5Query)

} else Future.None

val experimentalSANNCandidatesFut =

if (query.enableExperimentalSimClustersANN) {

simClustersANNSimilarityEngine.getCandidates(query.experimentalSimClustersANNQuery)

} else Future.None

val utgCandidatesFut = if (query.enableUtg) {

producerBasedUserTweetGraphSimilarityEngine.getCandidates(query.utgQuery)

} else Future.None

Future

.join(

sannCandidatesFut,

sann1CandidatesFut,

sann2CandidatesFut,

sann3CandidatesFut,

sann4CandidatesFut,

sann5CandidatesFut,

experimentalSANNCandidatesFut,

utgCandidatesFut

).map {

case (

simClustersAnnCandidates,

simClustersAnn1Candidates,

simClustersAnn2Candidates,

simClustersAnn3Candidates,

simClustersAnn4Candidates,

simClustersAnn5Candidates,

experimentalSANNCandidates,

userTweetGraphCandidates) =>

val filteredSANNTweets = simClustersCandidateMinScoreFilter(

simClustersAnnCandidates.toSeq.flatten,

query.simClustersMinScore,

query.simClustersANNQuery.storeQuery.simClustersANNConfigId)

val filteredExperimentalSANNTweets = simClustersCandidateMinScoreFilter(

experimentalSANNCandidates.toSeq.flatten,

query.simClustersMinScore,

query.experimentalSimClustersANNQuery.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 filteredUTGTweets =

userTweetGraphFilter(userTweetGraphCandidates.toSeq.flatten)

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 utgTweetsWithCGInfo = filteredUTGTweets.map { tweetWithScore =>

val similarityEngineInfo =

ProducerBasedUserTweetGraphSimilarityEngine

.toSimilarityEngineInfo(tweetWithScore.score)

TweetWithCandidateGenerationInfo(

tweetWithScore.tweetId,

CandidateGenerationInfo(

Some(query.sourceInfo),

similarityEngineInfo,

Seq(similarityEngineInfo)

))

}

val candidateSourcesToBeInterleaved =

ArrayBuffer[Seq[TweetWithCandidateGenerationInfo]](

sannTweetsWithCGInfo,

sann1TweetsWithCGInfo,

sann2TweetsWithCGInfo,

sann3TweetsWithCGInfo,

sann4TweetsWithCGInfo,

sann5TweetsWithCGInfo,

experimentalSANNTweetsWithCGInfo,

)

if (query.utgCombinationMethod == UnifiedSETweetCombinationMethod.Interleave) {

candidateSourcesToBeInterleaved += utgTweetsWithCGInfo

}

val interleavedCandidates =

InterleaveUtil.interleave(candidateSourcesToBeInterleaved)

val candidateSourcesToBeOrdered =

ArrayBuffer[Seq[TweetWithCandidateGenerationInfo]](interleavedCandidates)

if (query.utgCombinationMethod == UnifiedSETweetCombinationMethod.Frontload)

candidateSourcesToBeOrdered.prepend(utgTweetsWithCGInfo)

val candidatesFromGivenOrderCombination =

SimilaritySourceOrderingUtil.keepGivenOrder(candidateSourcesToBeOrdered)

val unifiedCandidatesWithUnifiedCGInfo = candidatesFromGivenOrderCombination.map {

candidate =>

/\*\*\*

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

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

\*

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

\* This is hard to add for interleave, and we plan to add it later after abstraction improvement.

\*/

TweetWithCandidateGenerationInfo(

tweetId = candidate.tweetId,

candidateGenerationInfo = getProducerBasedUnifiedCGInfo(

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\_userId\_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)

}

}

/\*\* A no-op filter as UTG filter already happened at 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)

}

}

}

object ProducerBasedUnifiedSimilarityEngine {

/\*\*\*

\* Every candidate will have the CG Info with ProducerBasedUnifiedSimilarityEngine

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

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

\*/

private def getProducerBasedUnifiedCGInfo(

sourceInfoOpt: Option[SourceInfo],

unifiedScore: Double,

contributingSimilarityEngines: Seq[SimilarityEngineInfo]

): CandidateGenerationInfo = {

CandidateGenerationInfo(

sourceInfoOpt,

SimilarityEngineInfo(

similarityEngineType = SimilarityEngineType.ProducerBasedUnifiedSimilarityEngine,

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

score = Some(unifiedScore)

),

contributingSimilarityEngines

)

}

case class Query(

sourceInfo: SourceInfo,

maxCandidateNumPerSourceKey: Int,

maxTweetAgeHours: Duration,

// SimClusters

enableSimClustersANN: Boolean,

simClustersANNQuery: EngineQuery[SimClustersANNSimilarityEngine.Query],

enableExperimentalSimClustersANN: Boolean,

experimentalSimClustersANNQuery: EngineQuery[SimClustersANNSimilarityEngine.Query],

enableSimClustersANN1: Boolean,

simClustersANN1Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

enableSimClustersANN2: Boolean,

simClustersANN2Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

enableSimClustersANN4: Boolean,

simClustersANN4Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

enableSimClustersANN3: Boolean,

simClustersANN3Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

enableSimClustersANN5: Boolean,

simClustersANN5Query: EngineQuery[SimClustersANNSimilarityEngine.Query],

simClustersMinScore: Double,

// UTG

enableUtg: Boolean,

utgCombinationMethod: UnifiedSETweetCombinationMethod.Value,

utgQuery: EngineQuery[ProducerBasedUserTweetGraphSimilarityEngine.Query])

def fromParams(

sourceInfo: SourceInfo,

params: configapi.Params,

): EngineQuery[Query] = {

val maxCandidateNumPerSourceKey = params(GlobalParams.MaxCandidateNumPerSourceKeyParam)

val maxTweetAgeHours = params(GlobalParams.MaxTweetAgeHoursParam)

// SimClusters

val enableSimClustersANN = params(

ProducerBasedCandidateGenerationParams.EnableSimClustersANNParam)

val simClustersModelVersion =

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

val simClustersANNConfigId = params(SimClustersANNParams.SimClustersANNConfigId)

// SimClusters - Experimental SANN Similarity Engine

val enableExperimentalSimClustersANN = params(

ProducerBasedCandidateGenerationParams.EnableExperimentalSimClustersANNParam)

val experimentalSimClustersANNConfigId = params(

SimClustersANNParams.ExperimentalSimClustersANNConfigId)

// SimClusters - SANN cluster 1 Similarity Engine

val enableSimClustersANN1 = params(

ProducerBasedCandidateGenerationParams.EnableSimClustersANN1Param)

val simClustersANN1ConfigId = params(SimClustersANNParams.SimClustersANN1ConfigId)

// SimClusters - SANN cluster 2 Similarity Engine

val enableSimClustersANN2 = params(

ProducerBasedCandidateGenerationParams.EnableSimClustersANN2Param)

val simClustersANN2ConfigId = params(SimClustersANNParams.SimClustersANN2ConfigId)

// SimClusters - SANN cluster 3 Similarity Engine

val enableSimClustersANN3 = params(

ProducerBasedCandidateGenerationParams.EnableSimClustersANN3Param)

val simClustersANN3ConfigId = params(SimClustersANNParams.SimClustersANN3ConfigId)

// SimClusters - SANN cluster 5 Similarity Engine

val enableSimClustersANN5 = params(

ProducerBasedCandidateGenerationParams.EnableSimClustersANN5Param)

val simClustersANN5ConfigId = params(SimClustersANNParams.SimClustersANN5ConfigId)

val enableSimClustersANN4 = params(

ProducerBasedCandidateGenerationParams.EnableSimClustersANN4Param)

val simClustersANN4ConfigId = params(SimClustersANNParams.SimClustersANN4ConfigId)

val simClustersMinScore = params(

ProducerBasedCandidateGenerationParams.SimClustersMinScoreParam)

// SimClusters ANN Query

val simClustersANNQuery = SimClustersANNSimilarityEngine.fromParams(

sourceInfo.internalId,

EmbeddingType.FavBasedProducer,

simClustersModelVersion,

simClustersANNConfigId,

params

)

val experimentalSimClustersANNQuery = SimClustersANNSimilarityEngine.fromParams(

sourceInfo.internalId,

EmbeddingType.FavBasedProducer,

simClustersModelVersion,

experimentalSimClustersANNConfigId,

params

)

val simClustersANN1Query = SimClustersANNSimilarityEngine.fromParams(

sourceInfo.internalId,

EmbeddingType.FavBasedProducer,

simClustersModelVersion,

simClustersANN1ConfigId,

params

)

val simClustersANN2Query = SimClustersANNSimilarityEngine.fromParams(

sourceInfo.internalId,

EmbeddingType.FavBasedProducer,

simClustersModelVersion,

simClustersANN2ConfigId,

params

)

val simClustersANN3Query = SimClustersANNSimilarityEngine.fromParams(

sourceInfo.internalId,

EmbeddingType.FavBasedProducer,

simClustersModelVersion,

simClustersANN3ConfigId,

params

)

val simClustersANN5Query = SimClustersANNSimilarityEngine.fromParams(

sourceInfo.internalId,

EmbeddingType.FavBasedProducer,

simClustersModelVersion,

simClustersANN5ConfigId,

params

)

val simClustersANN4Query = SimClustersANNSimilarityEngine.fromParams(

sourceInfo.internalId,

EmbeddingType.FavBasedProducer,

simClustersModelVersion,

simClustersANN4ConfigId,

params

)

// UTG

val enableUtg = params(ProducerBasedCandidateGenerationParams.EnableUTGParam)

val utgCombinationMethod = params(

ProducerBasedCandidateGenerationParams.UtgCombinationMethodParam)

EngineQuery(

Query(

sourceInfo = sourceInfo,

maxCandidateNumPerSourceKey = maxCandidateNumPerSourceKey,

maxTweetAgeHours = maxTweetAgeHours,

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,

enableUtg = enableUtg,

utgCombinationMethod = utgCombinationMethod,

utgQuery = ProducerBasedUserTweetGraphSimilarityEngine

.fromParams(sourceInfo.internalId, params)

),

params

)

}

def fromParamsForRelatedTweet(

internalId: InternalId,

params: configapi.Params

): EngineQuery[Query] = {

val maxCandidateNumPerSourceKey = params(GlobalParams.MaxCandidateNumPerSourceKeyParam)

val maxTweetAgeHours = params(GlobalParams.MaxTweetAgeHoursParam)

// SimClusters

val enableSimClustersANN = params(RelatedTweetProducerBasedParams.EnableSimClustersANNParam)

val simClustersModelVersion =

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

val simClustersANNConfigId = params(SimClustersANNParams.SimClustersANNConfigId)

val simClustersMinScore =

params(RelatedTweetProducerBasedParams.SimClustersMinScoreParam)

// SimClusters - Experimental SANN Similarity Engine

val enableExperimentalSimClustersANN = params(

RelatedTweetProducerBasedParams.EnableExperimentalSimClustersANNParam)

val experimentalSimClustersANNConfigId = params(

SimClustersANNParams.ExperimentalSimClustersANNConfigId)

// SimClusters - SANN cluster 1 Similarity Engine

val enableSimClustersANN1 = params(RelatedTweetProducerBasedParams.EnableSimClustersANN1Param)

val simClustersANN1ConfigId = params(SimClustersANNParams.SimClustersANN1ConfigId)

// SimClusters - SANN cluster 2 Similarity Engine

val enableSimClustersANN2 = params(RelatedTweetProducerBasedParams.EnableSimClustersANN2Param)

val simClustersANN2ConfigId = params(SimClustersANNParams.SimClustersANN2ConfigId)

// SimClusters - SANN cluster 3 Similarity Engine

val enableSimClustersANN3 = params(RelatedTweetProducerBasedParams.EnableSimClustersANN3Param)

val simClustersANN3ConfigId = params(SimClustersANNParams.SimClustersANN3ConfigId)

// SimClusters - SANN cluster 5 Similarity Engine

val enableSimClustersANN5 = params(RelatedTweetProducerBasedParams.EnableSimClustersANN5Param)

val simClustersANN5ConfigId = params(SimClustersANNParams.SimClustersANN5ConfigId)

val enableSimClustersANN4 = params(RelatedTweetProducerBasedParams.EnableSimClustersANN4Param)

val simClustersANN4ConfigId = params(SimClustersANNParams.SimClustersANN4ConfigId)

// Build SANN Query

val simClustersANNQuery = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.FavBasedProducer,

simClustersModelVersion,

simClustersANNConfigId,

params

)

val experimentalSimClustersANNQuery = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.FavBasedProducer,

simClustersModelVersion,

experimentalSimClustersANNConfigId,

params

)

val simClustersANN1Query = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.FavBasedProducer,

simClustersModelVersion,

simClustersANN1ConfigId,

params

)

val simClustersANN2Query = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.FavBasedProducer,

simClustersModelVersion,

simClustersANN2ConfigId,

params

)

val simClustersANN3Query = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.FavBasedProducer,

simClustersModelVersion,

simClustersANN3ConfigId,

params

)

val simClustersANN5Query = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.FavBasedProducer,

simClustersModelVersion,

simClustersANN5ConfigId,

params

)

val simClustersANN4Query = SimClustersANNSimilarityEngine.fromParams(

internalId,

EmbeddingType.FavBasedProducer,

simClustersModelVersion,

simClustersANN4ConfigId,

params

)

// UTG

val enableUtg = params(RelatedTweetProducerBasedParams.EnableUTGParam)

val utgCombinationMethod = params(

ProducerBasedCandidateGenerationParams.UtgCombinationMethodParam)

// SourceType.RequestUserId is a placeholder.

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

EngineQuery(

Query(

sourceInfo = sourceInfo,

maxCandidateNumPerSourceKey = maxCandidateNumPerSourceKey,

maxTweetAgeHours = maxTweetAgeHours,

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,

enableUtg = enableUtg,

utgQuery = ProducerBasedUserTweetGraphSimilarityEngine.fromParams(internalId, params),

utgCombinationMethod = utgCombinationMethod

),

params

)

}

}