package com.twitter.simclusters\_v2.hdfs\_sources

import com.twitter.scalding.DateRange

import com.twitter.scalding.TypedPipe

import com.twitter.scalding\_internal.dalv2.DAL

import com.twitter.scalding\_internal.dalv2.remote\_access.AllowCrossClusterSameDC

import com.twitter.scalding\_internal.dalv2.remote\_access.ExplicitLocation

import com.twitter.scalding\_internal.dalv2.remote\_access.Proc3Atla

import com.twitter.scalding\_internal.multiformat.format.keyval.KeyVal

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

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

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

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

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

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

object ProducerEmbeddingSources {

/\*\*

\* Helper function to retrieve producer SimClusters embeddings with the legacy `TopSimClustersWithScore`

\* value type.

\*/

def producerEmbeddingSourceLegacy(

embeddingType: EmbeddingType,

modelVersion: ModelVersion

)(

implicit dateRange: DateRange

): TypedPipe[(Long, TopSimClustersWithScore)] = {

val producerEmbeddingDataset = (embeddingType, modelVersion) match {

case (EmbeddingType.ProducerFollowBasedSemanticCoreEntity, ModelVersion.Model20m145kDec11) =>

ProducerTopKSimclusterEmbeddingsByFollowScoreScalaDataset

case (EmbeddingType.ProducerFavBasedSemanticCoreEntity, ModelVersion.Model20m145kDec11) =>

ProducerTopKSimclusterEmbeddingsByFavScoreScalaDataset

case (

EmbeddingType.ProducerFollowBasedSemanticCoreEntity,

ModelVersion.Model20m145kUpdated) =>

ProducerTopKSimclusterEmbeddingsByFollowScoreUpdatedScalaDataset

case (EmbeddingType.ProducerFavBasedSemanticCoreEntity, ModelVersion.Model20m145kUpdated) =>

ProducerTopKSimclusterEmbeddingsByFavScoreUpdatedScalaDataset

case (\_, \_) =>

throw new ClassNotFoundException(

"Unsupported embedding type: " + embeddingType + " and model version: " + modelVersion)

}

DAL

.readMostRecentSnapshot(producerEmbeddingDataset).withRemoteReadPolicy(

AllowCrossClusterSameDC)

.toTypedPipe.map {

case KeyVal(producerId, topSimClustersWithScore) =>

(producerId, topSimClustersWithScore)

}

}

def producerEmbeddingSource(

embeddingType: EmbeddingType,

modelVersion: ModelVersion

)(

implicit dateRange: DateRange

): TypedPipe[(Long, SimClustersEmbedding)] = {

val producerEmbeddingDataset = (embeddingType, modelVersion) match {

case (EmbeddingType.AggregatableLogFavBasedProducer, ModelVersion.Model20m145k2020) =>

AggregatableProducerSimclustersEmbeddingsByLogFavScore2020ScalaDataset

case (EmbeddingType.AggregatableFollowBasedProducer, ModelVersion.Model20m145k2020) =>

AggregatableProducerSimclustersEmbeddingsByFollowScore2020ScalaDataset

case (EmbeddingType.RelaxedAggregatableLogFavBasedProducer, ModelVersion.Model20m145k2020) =>

AggregatableProducerSimclustersEmbeddingsByLogFavScoreRelaxedFavEngagementThreshold2020ScalaDataset

case (\_, \_) =>

throw new ClassNotFoundException(

"Unsupported embedding type: " + embeddingType + " and model version: " + modelVersion)

}

DAL

.readMostRecentSnapshot(

producerEmbeddingDataset

)

.withRemoteReadPolicy(ExplicitLocation(Proc3Atla))

.toTypedPipe

.map {

case KeyVal(

SimClustersEmbeddingId(\_, \_, InternalId.UserId(producerId: Long)),

embedding: SimClustersEmbedding) =>

(producerId, embedding)

}

}

}